IntechOpen

Machine Learning

Edited by Abdelhamid Mellouk and Abdennacer Chebira

Machine Learning

Edited by

Abdelhamid Mellouk
and
Abdennacer Chebira

Machine Learning
http://dx.doi.org/10.5772/intechopen.85684
Edited by Abdelhamid Mellouk and Abdennacer Chebira

First published in Croatia, 2009 by INTECH d.o.o.
eBook (PDF) Published by IN TECH d.o.o.
Place and year of publication of eBook (PDF): Rijeka, 2019.
IntechOpen is the global imprint of IN TECH d.o.o.
Printed in Croatia

Legal deposit, Croatia: National and University Library in Zagreb

Additional hard and PDF copies can be obtained from orders@intechopen.com

Machine Learning
Edited by Abdelhamid Mellouk and Abdennacer Chebira

p. cm.

ISBN 978-3-902613-56-1

eBook (PDF) ISBN 978-953-51-5838-7

Meet the editors

Abdelhamid Mellouk , full Professor of Networks and Telecommunications Department (IUT C/V) at the University of Paris-Est (UPEC), Abdelhamid Mellouk is currently the Technical Program Chair of the IEEE Technical Committee on Communications Software in Communications Society and IEEE Senior Member. Founder of TINC (Transport Infrastructure and Network Control) research activity, his general area is in high-speed new generation wired/wireless networking, end to end quality of service and quality of experience. Currently, he is working on routing and switching optimization in dynamic traffic networks; human and bio-inspired artificial intelligence approaches. He investigates particularly the use of artificial neuronal intelligence together with biologically inspired techniques such as reinforcement learning, to control network decision in real-time.

Dr. Abdennasser Chebira received his Ph.D. degree in Electrical Engineering and Computer Sciences from PARIS XI University, Orsay, France, in 1994. Since September 1994 he works as Professor Assistant at Sénart Institute of Technology of PARIS XII – Val de Marne University. He is a staff researcher at Images, Signal and Intelligent Systems Laboratory (LISSI / EA 3956) of this University. His current research works concern selforganizing neural network based multi-modeling, hybrid neural based information processing systems; Neural based data fusion and complexity estimation.

Preface

Machine Learning is often referred to as a branch of artificial intelligence which deals with the design and the development of algorithms and techniques that help machines to "learn". Hence, it is closely related to various scientific domains as Optimization, Vision, Robotic and Control, Theoretical Computer Science, etc.

Based on this, Machine Learning can be defined in various ways related to a scientific domain concerned with the design and development of theoretical and implementation tools that allow building systems with some Human Like intelligent behavior. Machine learning addresses more specifically the ability to improve automatically through experience.

This book brings together many different aspects of the current research on several fields associated to Machine Learning. The selection of the chapters for this book was done in respect to the fact that it comprises a cross-edition of topics that reflect variety of perspectives and disciplinary backgrounds.

Four main parts have been defined and allow gathering the 21 chapters around the following topics: machine learning approaches, Human-like behavior and machine Human interaction, supervised and unsupervised learning approaches, reinforcement learning approaches and their applications.

This book starts with a first set of chapters which addresses general approaches in Machine Learning fields. One can find discussion about various issues: how to use the paradigm divide and conquer to build a hybrid self organized neural network tree structure, how to move from automation to autonomy, how to take experience to a whole new level, how to design very large scale networks based on Hamiltonian neural networks, how to design classifiers generative with similarity based abilities, and how information-theoretic competitive learning can force networks to increase knowledge.

In addition, the second part addresses the problem of Human-like behavior and machine Human interaction. It contains five chapters that deal with the following scope: Human-Knowledge poor-process of ontological information extraction, Machine learning for spoken dialogue system optimization, Bayesian additive regression trees applied to mail phishing detection, Composition of web services under multiple criteria and supervised learning problems under the ranking framework.

Another set of chapters presents an overview and challenges in several areas of supervised and unsupervised learning approaches. Subjects deal with generation method for a person specific facial expression map, linear subspace methods in the context of automatic facial expression analysis, nearest neighbor re-sampling method for prognostic gene expression patterns of tumor patients, 3D shape classification and retrieval algorithm and classification of faults in electrical power systems using a hybrid model based on neural networks.

The last part of the book deals with reinforcement learning approaches used in Machine Learning area. Various techniques are developed: Genetic learning programming and Sarsa learning allow the selection of appropriate stock trading rules in financial area, convergence of the online value-iteration in dynamic programming techniques is given in the case of the optimal control problem for general nonlinear discrete-time systems, modular reinforcement learning with situation-sensitive ability is used for intention estimation, experience replay technique is applied to real-words application and finally sequential modeling and prediction allow an adaptive intrusion detection in computer system.

This book shows that Machine Learning is a very dynamic area in terms of theory and application. The field of Machine Learning has been growing rapidly, producing a wide variety of learning algorithms for different applications. The ultimate value of those algorithms is to a great extent judged by their success in solving real-world problems. There is also a very extensive literature on Machine Learning, and to give a complete bibliography and a historical account of the research that led to the present form would have been impossible. It is thus inevitable that some topics have been treated in less detail than others. The choices made reflect on one hand personal taste and expertise and on second hand a preference for a very promising research and recent developments in Machine Learning fields.

Finally, we would to thank all contributors to this book for their research and effort.We hope you enjoy reading this book and get many helpful ideas and overviews for your own study.

Editors

Abdelhamid Mellouk
IUT Creteil/Vitry,
LiSSi Laboratory,
University of Paris 12
France

Abdennacer Chebira
IUT Senart/Fontainebleau,
LiSSi Laboratory,
University of Paris 12
France

Contents

Neural Machine Learning Approaches: Q-Learning and Complexity Estimation Based Information Processing System

Abdennasser Chebira, Abdelhamid Mellouk,
Kurosh Madani and Said Hoceini
LISSI laboratory, University Paris 12-Val de Marne
France

1. Introduction

Real world dilemmas, and especially industry related ones, are set apart from academic ones from several basic points of views. The difference appears since definition of the "problem's solution" notion. In fact, academic (called also sometime theoretical) approach often begins by problem's constraints simplification in order to obtain a "solvable" model (here, solvable model means a set of mathematically solvable relations or equations describing a behavior, phenomena, etc…) (Madani, 2008). If the theoretical consideration is a mandatory step to study a given problem's solvability, for a very large number of real world dilemmas, it doesn't lead to a solvable or realistic solution. Difficulty could be related to several issues among which:

- large number of parameters to be taken into account (influencing the behavior) making conventional mathematical tools inefficient,
- strong nonlinearity of the system (or behavior), leading to unsolvable equations,
- partial or total inaccessibility of system's relevant features, making the model insignificant,
- subjective nature of relevant features, parameters or data, making the processing of such data or parameters difficult in the frame of conventional quantification,
- necessity of expert's knowledge, or heuristic information consideration,
- imprecise information or data leakage.

Examples illustrating the above-mentioned difficulties are numerous and may concern various areas of real world or industrial applications. As first example, one can emphasize difficulties related to economical and financial modeling and prediction, where the large number of parameters, on the one hand, and human related factors, on the other hand, make related real world problems among the most difficult to solve. Another illustrative example concerns the delicate class of dilemmas dealing with complex data's and multifaceted information's processing, especially when processed information (representing patterns, signals, images, etc.) are strongly noisy or involve deficient data. In fact, real world and industrial applications, comprising system identification, industrial processes control, systems and plants safety, manufacturing regulation and optimization, pattern recognition, communication networks (complex routing, large communication networks management

and optimization, etc.) (Mellouk, 2008a), are often those belonging to such class of dilemmas.
If much is still to discover about how the animal's brain trains and self-organizes itself in order to process so various and so complex information, a number of recent advances in "neurobiology" allow already highlighting some of key mechanisms of this marvels machine. Among them one can emphasizes brain's "modular" structure and its "self-organizing" capabilities. In fact, if our simple and inappropriate binary technology remains too primitive to achieve the processing ability of these marvels mechanisms, a number of those highlighted points could already be sources of inspiration for designing new machine learning approaches leading to higher levels of artificial systems' intelligence (Madani, 2007).
In this chapter, we deal with machine learning based modular approaches which could offer powerful solutions to overcome processing difficulties in the aforementioned frame. If the machine learning capability provides processing system's adaptability and offers an appealing alternative for fashioning the processing technique adequacy, the modularity may result on a substantial reduction of treatment's complexity. In fact, the modularity issued complexity reduction may be obtained from several instances: it may result from distribution of computational effort on several modules; it can emerge from cooperative or concurrent contribution of several processing modules in handling a same task; it may drop from the modules' complementary contribution (e.g. specialization of a module on treating a given task to be performed).
A number of works dealing with modular computing and issued architectures have been proposed since 1990. Most of them associate a set of Artificial Neural Networks (ANN) in a modular structure in order to process a complex task by dividing it into several simpler sub-tasks. One can mention active learning approaches (Fahlman & Lebiere, 1990), neural networks ensemble concept proposed by (Hanibal, 1993), intelligent hybrid systems (Krogh & Vedelsby, 1995), Mixture of experts concept proposed by (Bruske & Sommer, 1995) and (Sung & Niyogi, 1995) or structures based on dynamic cells (Lang & Witbrock, 1998). In the same years, a number of authors proposed multi-modeling concept for nonlinear systems modeling, where a set of simple models is used to sculpt a complex behaviour (Goonnatilake & Khebbal, 1996), (Mayoubi et al., 1995), (Murray-Smith & Johansen, 1997), (Ernst, 1998)) in order to avoid difficulties (modeling complexity). However, it is important to remind that the most of proposed works (except those described in the four latest references) remain essentially theoretical and if a relatively consequent number of different structures have been proposed, a very few of them have been applied to real-world dilemmas solution.
The present chapter focuses those machine learning based modular approaches which take advantage either from modules' independence (multi-agent approach) or from self-organizing multi-modeling ("divide and conquer" paradigm). In other words, we will expound online and self-organizing approaches which are used when no a priori learning information is available. Within this frame, we will present, detail and discuss two challenging applicative aspects: the first one dealing with routing optimization in high speed communication networks and the other with complex information processing. Concerning the network routing optimization problem, we will describe and evaluate an adaptive online machine learning based approach, combining multi-agent based modularity and neural network based reinforcement learning ((Mellouk, 2007), (Mellouk, 2008b)). On the side of complex information processing, we will describe and evaluate a self-organizing

modular machine learning approach, combining "divide and conquer" paradigm and “complexity estimation” techniques that we called self-organizing “Tree-like Divide To Simplify” (T-DTS) approach ((Madani et al., 2003), (Madani et al., 2005), (Bouyoucef et al., 2005), (Chebira et al., 2006)).
This chapter is composed by four sections. The second section presents the state of the art of modular approaches over three modular paradigms: "divide and conquer" paradigm, Committee Machines and Multi Agent systems. In section 3, a neural network based reinforcement learning approach dealing with adaptive routing in communication networks is presented. In the last section, dealing with complex information processing, we will detail the self-organizing Tree divide to simplify approach, including methods and strategies for building the modular structure, decomposition of databases and finally processing. A sub-section will present a number of aspects relating “complexity estimation” that is used in T-DTS in order to self-organize such modular structure. Evaluating the universality of T-DTS approach, by showing its applicability to different classes of problems will concern other sub-sections of this fourth section. Global conclusions end this chapter and give further perspectives for the future development of proposed approaches.

2. Modular approaches

Apart from specialized "one-piece" algorithm as explicit solution of a problem, there exist a number of alternative solutions, which promote modular structure. In modular structure, units (computational unit or model) could either have some defined and regularized connectivity or be more or less randomly linked, ending up at completely independent and individual units. The units can communicate with each others. The units’ communication may take various forms. It may consist of data exchange. It may consist of orders exchange, resulting either on module’s features modification or on its structure. Units may espouse cooperative or competitive interaction. A modular structure composed of Artificial Neural Networks is called Multi Neural Network (MNN).
We will present here three modular paradigms that are of particular interest: "Divide and Conquer" paradigm, Committee Machines and Multi Agent Systems. "Divide and conquer" paradigm is certainly a leading idea for the tree structure described in this section. Committee machines are in large part incorporation of this paradigm. For multi-agent approach the stress is put on the modules independence.

2.1 “Divide and Conquer" paradigms

This approach is based on the principle "Divide et Impera" (Julius Caesar). The main frame of the principle can be expressed as:
- Break up problem into two (or more) smaller sub-problems;
- Solve sub-problems;
- Combine results to produce a solution to original problem.

The ways in which the original problem is split differ as well as the algorithms of solving sub-problems and combining the partial solutions. The splitting of the problem can be done in recursive way. Very known algorithm using this paradigm is Quicksort (Hoare, 1962), which splits recursively data in order to sort them in a defined order. In the Artificial Neural Networks area the most known algorithm of similar structure is Mixture of Experts (Bruske & Sommer, 1995).

Algorithmic paradigms evaluation could be made on the basis of running time. This is useful in that it allows computational effort comparisons between the performances of two algorithms to be made. For Divide-and-Conquer algorithms the running time is mainly affected by:

- The number of sub-instances into which a problem is split;
- The ratio of initial problem size to sub-problem size;
- The number of steps required to divide the initial instance and to combine sub-solutions;
- Task complexity;
- Database size.

2.2 Committee machines

The committee machines are based on engineering principle divide and conquer. According to that rule, a complex computational task is solved by dividing it into a number of computationally simple sub-tasks and then combining the solutions of these sub-tasks. In supervised learning, the task is distributed among a number of experts. The combination of experts is called committee machine. Committee machine fuses knowledge of experts to achieve an overall task, which may be more efficient than that achieved by any of the experts alone (Tresp, 2001).
The taxonomy of committee machines could be as follows:

- Static structures: Ensemble Averaging and Boosting;
- Dynamic structures: Mixture of Experts and Hierarchical Mixture of Experts.

Next several subsections will present the types of committee machines in detail.

2.2.1 Ensemble averaging

In ensemble averaging technique (Haykin, 1999), (Arbib, 1989), a number of differently trained experts (i.e. neural networks) share a common input and their outputs are combined to produce an overall output value *y*.

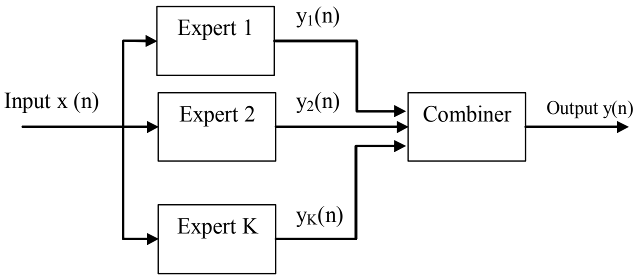


Fig. 1. Ensemble averaging structure

The advantage of such structure over a single expert is that the variance of the average function is smaller than the variance of single expert. Simultaneously both average functions have the same bias. These two facts lead to a training strategy for reducing the overall error produced by a committee machine due to varying initial conditions (Naftaly et al., 1997): the experts are purposely over-trained, what results in reducing the bias at the variance cost. The variance is subsequently reduced by averaging the experts, leaving the bias unchanged.

2.2.2 Boosting

In boosting approach (Schapire, 1999) the experts are trained on data sets with entirely different distributions; it is a general method which can improve the performance of any learning algorithm. Boosting can be implemented in three different ways: Boosting by filtering, Boosting by sub-sampling and Boosting by re-weighing. A well known example is AdaBoost (Schapire, 1999) algorithm, which runs a given weak learner several times on slightly altered training data, and combining the hypotheses to one final hypothesis, in order to achieve higher accuracy than the weak learner's hypothesis would have.

2.2.3 Mixture of experts

Mixture of experts consists of *K* supervised models called expert networks and a gating network, which performs a function of mediator among expert networks. The output is a weighted sum of experts' outputs (Jordan & Jacobs, 2002).
A typical Mixture of Experts structure is presented by figure 2. One can notice the *K* experts and a gating network that filters the solutions of experts. Finally the weighted outputs are combined to produce overall structure output. The gating network consists of *K* neurons, each one is assigned to a specific expert.
The neurons in gating network are nonlinear with activation function that is a differentiable version of "winner-takes-all" operation of picking the maximum value. It is referred as "softmax" transfer function (Bridle, 1990). The mixture of experts is an associative Gaussian mixture model, which is a generalization of traditional Gaussian mixture model (Titterington et al., 1985), (MacLachlan & Basford, 1988).

2.2.4 Hierarchical mixture of experts

Hierarchical mixture of experts (Jordan & Jacobs, 1993) works similarly to ordinary mixture of experts, except that multiple levels of gating networks exist. So the outputs of mixture of experts are gated in order to produce combined output of several mixtures of expert structures. In figure 3 one can see two separate mixture of experts blocks (marked with dashed rectangles). The additional gating network is gating the outputs of these two blocks in order to produce the global structure output.

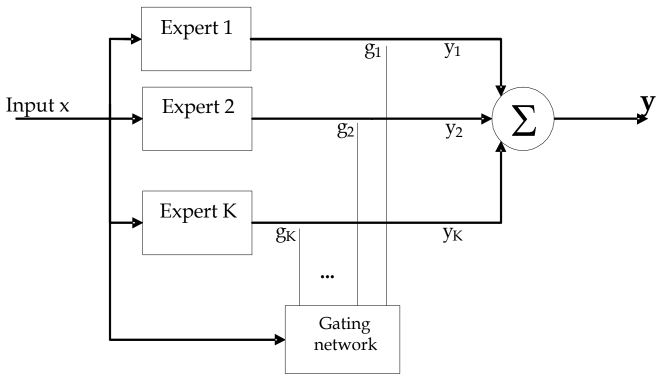


Fig. 2. Mixture of Experts

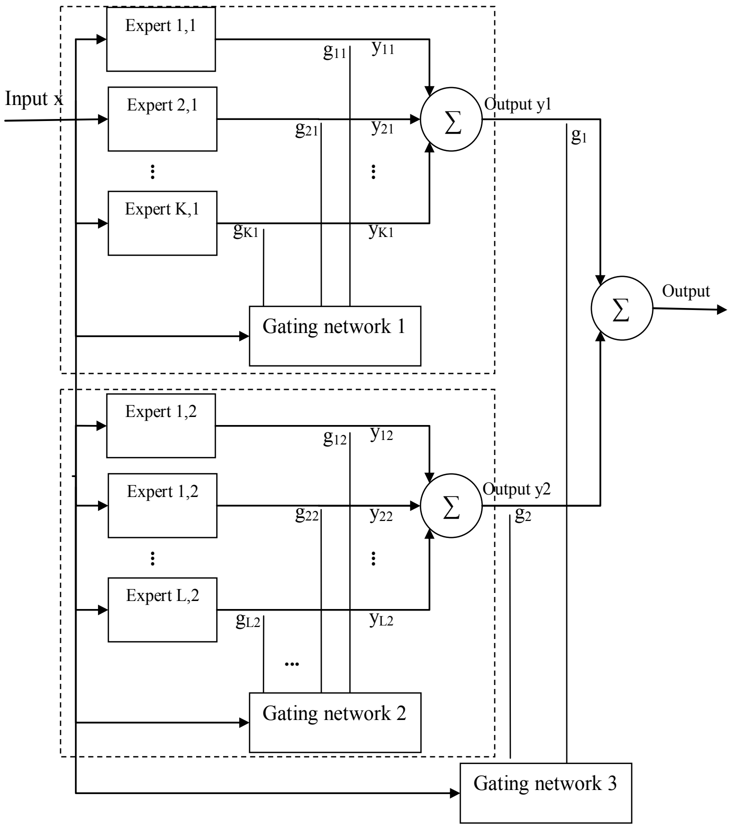


Fig. 3. Example of hierarchical mixture of experts

2.3 Multi agent systems

Multi agent system is a system that compounds of independent modules called "agents". There is no single control structure (designer) which controls all agents. Each of these agents can work on different goals, sometimes in cooperative and sometimes in competitive modes. Both cooperation and competition modes are possible among agents (Decker et al., 1997). There is a great variety of intelligent software agents and structures. The characteristics of Multi Agent Systems (Ferber, 1998) are:

- Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;
- There is no system global control;
- Data are decentralized;
- Computation is asynchronous.

In Multi Agent Systems many intelligent agents interact with each other. The agents can share a common goal (e.g. an ant colony), or they can pursue their own interests (as in the free market economy). Figure 4 gives the classification of intelligent artificial agents considering their origin.

Agents may also be classified according to the tasks they perform:

- **Interface Agents** - Computer programs using artificial intelligence techniques in order to provide assistance to a user dealing with a particular application. The metaphor is that of a personal assistant who is collaborating with the user in the same work environment (Maes, 1994).
- **Information Agents** - An information agent is an agent that has access to at least one, and potentially many information sources, and is able to collect and manipulate information obtained from these sources to answer to users and other information agent's queries (Wooldridge & Jennings, 1995).

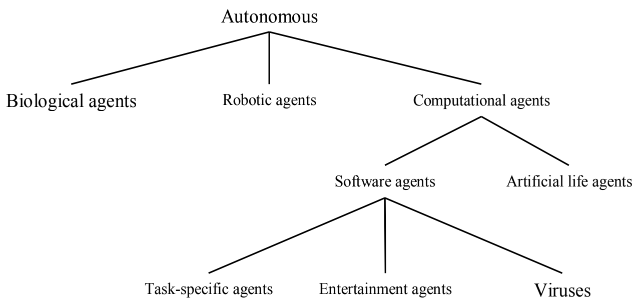


Fig. 4. Classification of intelligent artificial agents considering origin

- **Commerce Agents**- A commerce agent is an agent that provides commercial services (e.g., selling, buying and prices' advice) for a human user or for another agent.
- **Entertainment Agents** - Artistically interesting, highly interactive, simulated worlds to give users the experience of living in (not merely watching) dramatically rich worlds that include moderately competent, emotional agents (Bate et al., 1992).

Agents can communicate, cooperate and negotiate with other agents. The basic idea behind Multi Agent systems is to build many agents with small areas of action and link them together to create a structure which is much more powerful than the single agent itself.

2.4 Discussion

If over past decade wide studies have been devoted to theoretical aspects of modular structures (and algorithms), very few works have concerned their effective implementation and their application to real-world dilemmas. Presenting appealing potential advantages over single structures, this kind of processing systems may avoid difficulties inherent to large and complicated processing systems by splitting the initial complex task into a set of simpler task requiring simpler processing algorithms. The other main advantage is the customized nature of the modular design regarding the task under hand. Among the above-presented structures, the "Divide and Conquer" class of algorithms presents engaging faultlessness. Three variants could be distinguished:

- Each module works with full database aiming a "global" processing. This variant uses a combination of the results issued from individual modules to construct the final system's response.
- Modules work with a part of database (sub-database) aiming a "local" but "not exclusive" processing. In this variant, some of the processing data could be shared by several modules. However, depending on the amount of shared data this variant could be more or less similar to the two others cases.

- Modules work with a part of database (sub-database) aiming a “local” and “exclusive” processing. In this option, sub-databases are exclusive by meaning that no data is shared by modules. The final system’s result could either be a set of responses corresponding to different parts of the initial treated problem or be the output of the most appropriated module among the available ones.

Tree-like Divide To Simplify Approach (described later in this chapter) could be classified as belonging to "Divide and Conquer" class of algorithms as it breaks up an initially complex problem into a set of sub-problems. However, regarding the three aforementioned variants, its actually implemented version solves the sub-problems issued from the decomposition process according to the last variant. In the next section, we present a first modular algorithms which hybridize multi-agents techniques and Q-Neural learning.

3. Multi-agents approach and Q-neural reinforcement learning hybridization: application to QoS complex routing problem

This section present in detail a Q-routing algorithm optimizing the average packet delivery time, based on Neural Network (NN) ensuring the prediction of parameters depending on traffic variations. Compared to the approaches based on Q-tables, the Q-value is approximated by a reinforcement learning based neural network of a fixed size, allowing the learner to incorporate various parameters such as local queue size and time of day, into its distance estimation. Indeed, a Neural Network allows the modeling of complex functions with a good precision along with a discriminating training and network context consideration. Moreover, it can be used to predict non-stationary or irregular traffics. The Q-Neural Routing algorithm is presented in detail in section 3.2. The performance of Q-Routing and Q-Neural Routing algorithms are evaluated experimentally in section 3.3 and compared to the standard shortest path routing algorithms.

3.1 Routing problem in communication networks

Network, such as Internet, has become the most important communication infrastructure of today's human society. It enables the world-wide users (individual, group and organizational) to access and exchange remote information scattered over the world. Currently, due to the growing needs in telecommunications (VoD, Video-Conference, VoIP, etc.) and the diversity of transported flows, Internet network does not meet the requirements of the future integrated-service networks that carry multimedia data traffic with a high Quality of Service (QoS). The main drivers of this evolution are the continuous growth of the bandwidth requests, the promise of cost improvements and finally the possibility of increasing profits by offering new services. First, it does not support resource reservation which is primordial to guarantee an end-to-end Qos (bounded delay, bounded delay jitter, and/or bounded loss ratio). Second, data packets may be subjected to unpredictable delays and thus may arrive at their destination after the expiration time, which is undesirable for continuous real-time media. In this Context, for optimizing the financial investment on their networks, operators must use the same support for transporting all the flows. Therefore, it is necessary to develop a high quality control mechanism to check the network traffic load and ensure QoS requirements.

A lot of different definitions and parameters for this concept of quality of service can be found. For ITU-T E.800 recommendation, QoS is described as “the collective effect of service performance which determines the degree of satisfaction of a user of the service”. This

definition is completed by the I.350 ITU-T recommendation which defines more precisely the differences between QoS and Network Performance. Relating QoS concepts in the Internet are focused on a packet-based end-to-end, edge-to-edge or end-to-edge communication. QoS parameters which refer to this packet transport at different layers are: availability, bandwidth, delay, jitter and loss ratio. It's clear that the integration of these QoS parameters increases the complexity of the used algorithms. Anyway, there will be QoS relevant technological challenges in the emerging hybrid networks which mixes several networks topologies and technologies (wireless, broadcast, mobile, fixed, etc.).
In the literature, we can find the usage of QoS in three ways:

- *Deterministic QoS* consists in sufficiently resources reserved for a particular flow in order to respect the strict temporal constraints for all the packages of flow. No loss of package or going beyond of expiries is considered in this type of guarantee. This model makes it possible to provide an absolute terminal on the time according to the reserved resources.
- *Probabilistic QoS* consists in providing a long-term guarantee of the level of service required by a flow. For time-reality applications tolerating the loss of a few packages or the going beyond of some expiries, the temporal requirements as well as the rates of loss are evaluated on average. The probabilistic guarantee makes it possible to provide a temporal terminal with a certain probability which is given according to the conditions of load of the network.
- *Stochastic QoS* which is fixed before by a stochastic distribution.

Various techniques have been proposed to take into account QoS requirements (Strassner, 2003). By using in-band or out-band specific control protocols, these techniques may be classified as follows: the congestion control (Slow Start (Welzl, 2003), Weighted Random Early Detection (Jacobson, 1988)), the traffic shaping (Leaky Bucket (Feng et al., 1997), Token Bucket (Turner, 1986)), integrated services architecture, (RSVP (Shenker et al., 1997), (Zhang et al., 1993)), the differentiated services (DiffServ (Zhang et al., 1993), (Bernet, 1998)) and QoS based routing. In this section, we focus on QoS routing policies.
A routing algorithm is based on the hop-by-hop shortest-path paradigm. The source of a packet specifies the address of the destination, and each router along the route forwards the packet to a neighbour located "closest" to the destination. The best optimal path is selected according to given criteria. When the network is heavily loaded, some of the routers introduce an excessive delay while others are ignored (not expoited). In some cases, this non-optimized usage of the network resources may introduce not only excessive delays but also high packet loss rate. Among routing algorithms extensively employed in routers, one can note: distance vector algorithm such as RIP (Malkin, 1993) and the link state algorithm such as OSPF (Moy, 1998). These kinds of algorithms take into account variations of load leading to limited performances.
A lot of study has been conducted in a search for an alternative routing paradigm that would address the integration of dynamic criteria. The most popular formulation of the optimal distributed routing problem in a data network is based on a multi-commodity flow optimization whereby a separable objective function is minimized with respect to the types of flow subject to multi-commodity flow constraints (Gallager, 1977), (Ozdalgar et al., 2003). However, due their complexity, increased processing burden, a few proposed routing schemes could be accepted for the internet. We listed here some QoS based routing algorithms proposed in the literature: QOSPF (Quality Of Service Path First) (Crawley et al.,

1998), MPLS (Multiprotocol label switching) (Rosen et al., 1999), (Stallings, 2001), (Partridge, 1992), Traffic Engineering (Strasnner, 2003), (Welzl, 2003), Wang-Crowcroft algorithm (Wang & Crowcroft, 1996), Ants routing approach (Subramanian et al., 1997), Cognitive Packet Networks based on random neural networks (Gelenbe et al., 2002).

For a network node to be able to make an optimal routing decision, according to relevant performance criteria, it requires not only up-to-date and complete knowledge of the state of the entire network but also an accurate prediction of the network dynamics during propagation of the message through the network. This, however, is impossible unless the routing algorithm is capable of adapting to network state changes in almost real time. So, it is necessary to develop a new intelligent and adaptive optimizing routing algorithm. This problem is naturally formulated as a dynamic programming problem, which, however, is too complex to be solved exactly.

In our approach, we use the methodology of reinforcement learning (RL) introduced by Sutton (Sutton & Barto, 1997) to approximate the value function of dynamic programming. One of pioneering works related to this kind of approaches concerns Q-Routing algorithm (Boyan & Littman, 1994) based on Q-learning technique (Watkins & Dayan, 1989). In this approach, each node makes its routing decision based on the local routing information, represented as a table of Q values which estimate the quality of the alternative routes. These values are updated each time the node sends a packet to one of its neighbors. However, when a Q value is not updated for a long time, it does not necessarily reflect the current state of the network and hence a routing decision based on such an unreliable Q value will not be accurate. The update rule in Q-Routing does not take into account the reliability of the estimated or updated Q value because it's depending on the traffic pattern, and load levels, only a few Q values are current while most of the Q values in the network are unreliable. For this purpose, other algorithms have been proposed like Confidence based Q-Routing (CQ-Routing) (Kumar & Miikkualainen, 1998) or Dual Reinforcement Q-Routing (DRQ-Routing) (Kumar & Miikkualainen, 1997), (Goetz et al., 1996). All these routing algorithms use a table to estimate Q values. However, the size of the table depends on the number of destination nodes existing in the network. Thus, this approach is not well suited when we are concerned with a state-space of high dimensionality.

3.2 Q-neural routing approach

In this section, we present an adaptive routing algorithm based on the Q-learning approach, the Q-function is approximated by a reinforcement learning based neural network. First, we formulate the reinforcement learning process.

3.2.1 Reinforcement learning

Algorithms for reinforcement learning face the same issues as traditional distributed algorithms, with some additional peculiarities. First, the environment is modelled as stochastic (especially links, link costs, traffic, and congestion), so routing algorithms can take into account the dynamics of the network. However no model of dynamics is assumed to be given. This means that RL algorithms have to sample, estimate, and perhaps build models of pertinent aspect of the environment. Second, RL algorithms, unlike other machine learning algorithms, do not have an explicit learning phase followed by evaluation. Since there is no training signal for a direct evaluation of the policy's performance before the packet has reached its final destination, it is difficult to apply supervised learning techniques to this

problem (Haykin, 1998). In addition, it is difficult to determine to what extent a routing decision that has been made on a single node may influence the network's overall performance. This fact fits into the temporal credit assignment problem (Watkins, 1989).
The RL algorithm, called reactive approach, consists of endowing an autonomous agent with a correctness behavior guaranteeing the fulfillment of the desired task in the dynamics environment. The behavior must be specified in terms of Perception - Decision – Action loop (Fig. 5). Each variation of the environment induces stimuli received by the agent, leading to the determination of the appropriate action. The reaction is then considered as a punishment or a performance function, also called, reinforcement signal.

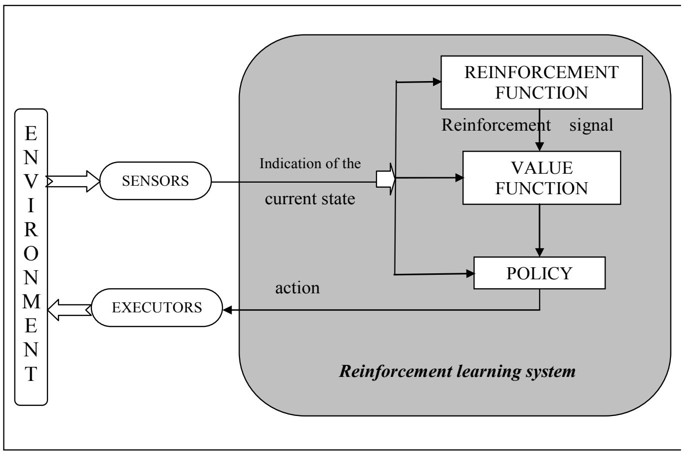


Fig. 5. Reinforcement learning model

Thus, the agent must integrate this function to modify its future actions in order to reach an optimal performance. In other words, a RL Algorithm is a finite-state machine that interacts with a stochastic environment, trying to learn the optimal action the environment offers through a learning process. At any iteration the automaton's agent chooses an action, according to a probability vector, using an output function. This function stimulates the environment, which responds with an answer (reward or penalty). The automaton's agent takes into account this answer and jumps, if necessary, to a new state using a transition function.

Network Elements	**RL System**	
Network	Environment	-
Each network node	Agent	-
Delay in links and nodes	Reinforcement	T
Estimate of total delay	Function	$Q(s, y, d)$
Action of sending a packet	Value Function	$a(s_t)$
Node through which the packet passes	Action	s_t
in time t	State in time t	
Local routing decision	Policy	π

Table. 1. Correspondences between a RL system and network elements

It is necessary for the agent to gather useful experience about the possible system states, actions, transitions and rewards actively to act optimally. Another difference from

supervised learning is that on-line performance is important: the evaluation of the system is often concurrent with learning.
A Reinforcement Learning system thus involves the following elements: an Agent, an Environment, a Reinforcement Function, an Action, a State, a Value Function, which is obtained from the reinforcement function, and a Policy. In order to obtain a network routing useful model, it is possible to associate the network's elements to the basic elements of a RL system, as shown in Table 1.

3.2.2 Q-learning algorithm for routing

In our routing algorithm (Mellouk, 2006), the objective is to minimize the average packet delivery time. Consequently, the reinforcement signal which is chosen corresponds to the estimated time to transfer a packet to its destination. Typically, the packet delivery time includes three variables: The packet transmission time, the packet treatment time in the router and the latency in the waiting queue. In our case, the packet transmission time is not taken into account. In fact, this parameter can be neglected in comparison to the other ones and has no effect on the routing process.
The reinforcement signal *T* employed in the Q-learning algorithm can be defined as the minimum of the sum of the estimated *Q (y, x, d)* sent by the router *x* neighbor of router *y* and the latency in waiting queue q_y corresponding to router *y*.

$$T = \min_{x \in \text{neighbor of y}} \left\{ q_y + Q(y,x,d) \right\} \tag{1}$$

Q(s, y, d) denote the estimated time by the router *s* so that the packet *p* reaches its destination *d* through the router *y*. This parameter does not include the latency in the waiting queue of the router *s*. The packet is sent to the router *y* which determines the optimal path to send this packet (Watkins, 1989).

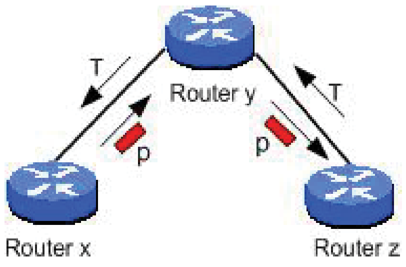


Fig. 6. Updating the reinforcement signal

Once the choice of the next router made, the router *y* puts the packet in the waiting queue, and sends back the value *T* as a reinforcement signal to the router *s*. It can therefore update its reinforcement function as:

$$\Delta Q(s,y,d) = \eta(\alpha + T - Q(s,y,d)) \tag{2}$$

So, the new estimation $Q'(s,y,d)$ can be written as follows (fig.6):

$$Q'(s,y,d) = Q(s,y,d)\left(1-\eta\right) + \eta(T+\alpha) \tag{3}$$

α and η are respectively, the packet transmission time between *s* and *y*, and the learning rate.

3.2.2 Q-learning neural net architecture

The neural network proposed in our study is a Recurrent Multi-Layers Perceptron (MLP) with an input, one hidden and an output layer.

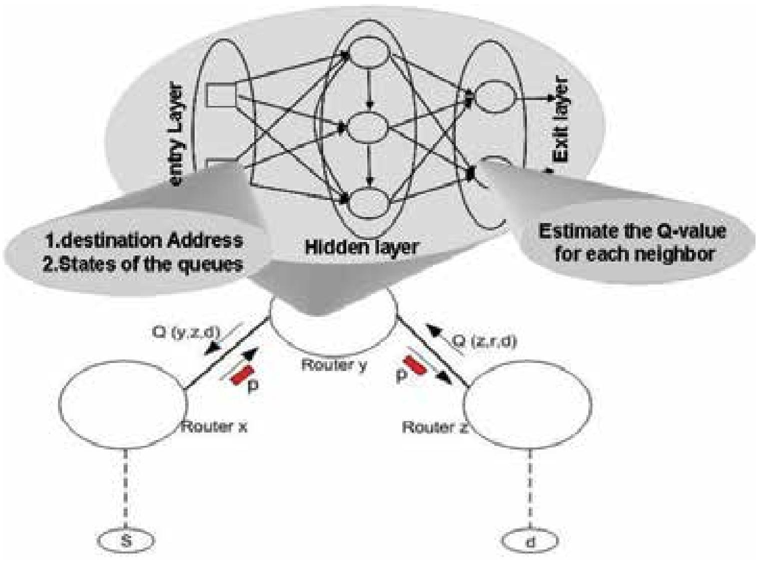


Fig. 7. Artificial Neural Network Architecture

The input cells correspond to the destination addresses *d* and the waiting queue states. The outputs are the estimated packet transfer times passing through the neighbors of the considered router. The algorithm derived from this architecture is called Q-Neural Routing and can be described according to the following algorithm:

```
Etiq1 :
{While (not packet receive)
    Begin
    End
}
If (packet = "packet of reinforcement")
    Begin
        1. Neural Network updating using a retro-propagation algorithm based on gradient
           method,
        2. Destroy the reinforcement packet.
    End
Else
    Begin
        1.  Calculate Neural Network outputs,
        2.  Select the smallest output value and get an IP address of the associated router,
        3.  Send the packet to this router,
        4.  Get an IP address of the precedent router,
        5.  Create and send the packet as a reinforcement signal.
    End
End
Goto Etiq1
```

3.3 Implementation and simulation results

To show the efficiency and evaluate the performances of our approach, an implementation has been performed on OPNET software of MIL3 Company. The proposed approach has been compared to that based on standard Q-routing (Boyan & Littman, 1994) and shortest path routing policy. OPNET constitutes for telecommunication networks an appropriate modeling, scheduling and simulation tool. It allows the visualization of a physical topology of a local, metropolitan, distant or on board network. The protocol specification language is based on a formal description of a finite state automaton.

The proposed approaches have been compared to that based on standard Q-routing and shortest paths routing policies (such as Routing Internet Protocol RIP). The topology of the network used for simulations purpuse, which used in many papers, includes 33 interconnected nodes, as shown in figure 8. Two kinds of traffic have been studied: low load and high load of the network. In the first case, a low rate flow is sent to node destination-1, from nodes source-1 and source-4. From the previous case, we have created conditions of congestion of the network. Thus, a high rate flow is generated by nodes source-2 and source-3. Two possible ways R-1 (router-29 and router-30) and R-2 (router-21 and router-22) to route the packets between the left part and the right part of the network.

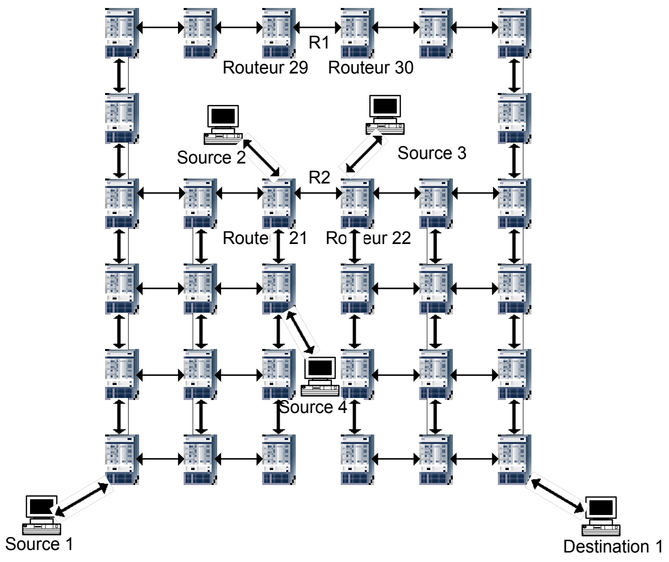


Fig. 8. Network topology for simulation

Performances of algorithms are evaluated in terms of average packet delivery time. Figure 9 and figure 10 illustrates the obtained results when source-2 and source-3 send information packets during 10 minutes. From figure 10, one can see clearly, that after an initialization period, the Q-routing and Q-Neural routing algorithms, exhibit better performances than RIP. Thus, packet average delivery time obtained by Q-routing algorithm and Q-Neural routing algorithm is reduced of respectively 23.6% and 27.3% compared to RIP routing policy (table 2). These results confirm that classical shortest path routing algorithm like RIP lead to weak performances due to packets delayed in the waiting queues of the routers. Moreover, this policy does not take into account the load of the network. On the other hand, when a way of destination is saturated, Q-routing and Q-Neural routing algorithms allow the selection of a new one to avoid this congestion. In the case of a low load (figure 10), one can note that after a period of initialization, performances of these algorithms are approximately the same as those obtained with RIP routing policy.

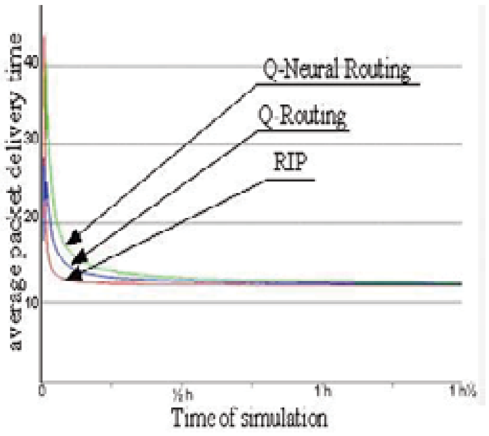


Fig. 9. Network with a low load

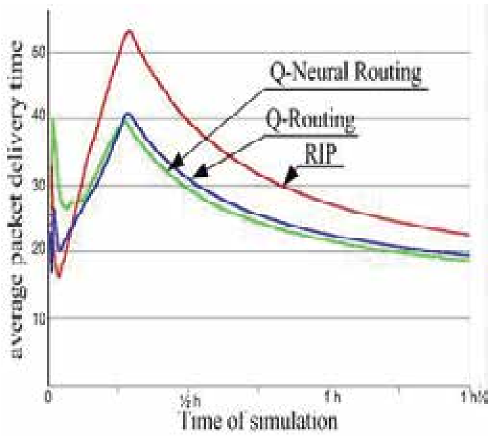


Fig. 10. Network with a high load

Computed Algorithms	*MAPTT*
Q-routing	42
Q-neural routing	40
RIP	55

Table 2. Maximum average packet delivery time

Figure 11 illustrates the average packet delivery time obtained when a congestion of the network is generated during 60 minutes. Thus, in the case where the number of packets is more important, the Q-Neural routing algorithm gives better results compared to Q-routing algorithm. For example, after 2 hours of simulation, Q-Neural routing exhibits a performance of 20% higher than that of Q-routing. Indeed, the use of waiting queue state of the neighboring routers in the routing decision, allows anticipation of routers congestion.

In general, the topology of the neural network must be fixed before the training process. The only variables being able to be modified are the values of the weights of connections. The specification of this architecture, the number of cells of each layer and of connections, remains a crucial problem. If this number is insufficient, the model will not be able to take into account all data. A contrario, if it is too significant, the training will be perfect but the network generalization ability will be poor (overfitting problem). However, we are concerned here by online training, for which the number of examples is not defined a priori. For that, we propose an empirical study based on pruning technique to find a compromise between a satisfactory estimate of the function Q and an acceptable computing time.

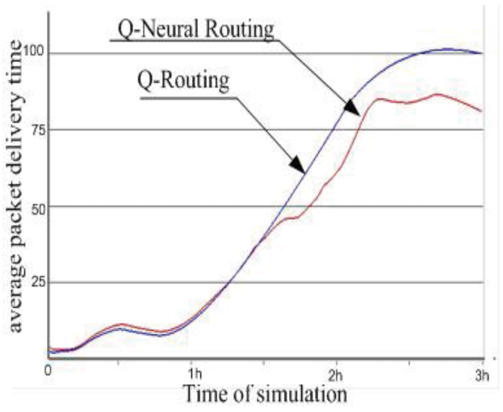


Fig. 11. Very High load Network

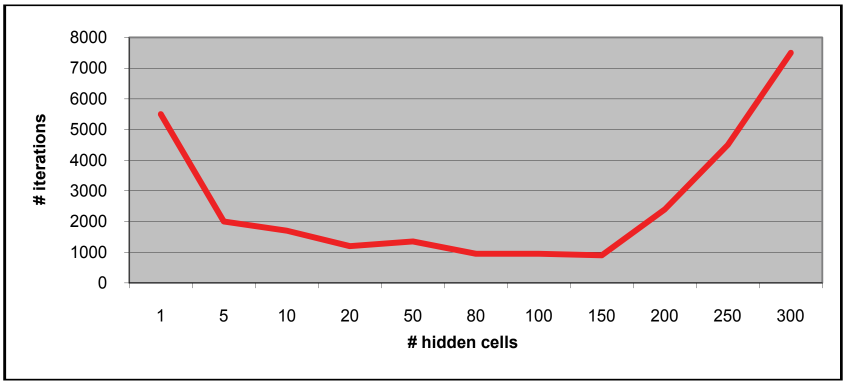


Fig. 12. Empirical pruning study for choosing the number of hidden cells over time

The Neural network structure has been fixed using an empirical pruning strategy (figure 12). A self-organizing approach is useful for automatic adjustment of the neural network parameters as the number of neuron per layer and the hidden layers numbers for example. Next section introduces such a concept and present complexity estimation based self organizing structure.

4. Self-organizing modular information processing through the Tree-like Divide To Simplify approach

This section presents in detail the "Tree-like-Divide To Simplify" (T-DTS) approach, define its structure, and describe the types of modules that are used in the structure. T-DTS is based on modular tree-like decomposition structure, which is used amongst others for task decomposition. This section will present also in detail procedures and algorithms that are used for the creation, execution and modification of the modules. It will discuss also advantages and disadvantages of T-DTS approach and compare it with other approaches.
T-DTS is a self-organizing modular structure including two types of modules: Decomposition Unit (DU) and Processing Unit (PU). The purpose is based on the use of a set of specialized mapping neural networks (PU), supervised by a set of DU. DU could be a prototype based neural network, Markovian decision process, etc. The T-DTS paradigm

allows us to build a modular tree structure. In such structure, DU could be seen as "nodes" and PU as leaves. At the nodes level, the input space is decomposed into a set of subspaces of smaller sizes. At the leaves level, the aim is to learn the relations between inputs and outputs of sub-spaces, obtained from splitting. As the modules are based on Artificial Neural Networks, they inherit the ANN's approximation universality as well as their learning and generalization abilities.

4.1 Hybrid Multiple Neural Networks framework - T-DTS

As it has been mentioned above, in essence, T-DTS is a self-organizing modular structure (Madani et Al., 2003). T-DTS paradigm builds a tree-like structure of models (DU and PU). Decomposition Units are prototypes based ANNs and Processing Units are specialized mapping ANNs. However, in a general frame, PU could be any kind of processing model (conventional algorithm or model, ANN based model, etc…). At the nodes level(s) - the input space is decomposed into a set of optimal sub-spaces of the smaller size. At the leaves level(s) - the aim is to learn the relation between inputs and outputs of sub-spaces obtained from splitting. T-DTS acts in two main operational phases:

Learning: recursive decomposition under DU supervision of the database into sub-sets: tree structure building phase;

Operational: Activation of the tree structure to compute system output (provided by PU at tree leaf's level).

General block diagram of T-DTS is described on Figure 13. The proposed schema is open software architecture. It can be adapted to specific problem using the appropriate modeling paradigm at PU level: we use mainly Artificial Neural Network computing model in this work. In our case the tree structure construction is based on a complexity estimation module. This module introduces a feedback in the learning process and control the tree building process. The reliability of tree model to sculpt the problem behavior is associated to the complexity estimation module. The whole decomposing process is built on the paradigm "*splitting database into sub-databases - decreasing task complexity*". It means that the decomposition process is activated until a low satisfactory complexity ratio is reached. T- DTS

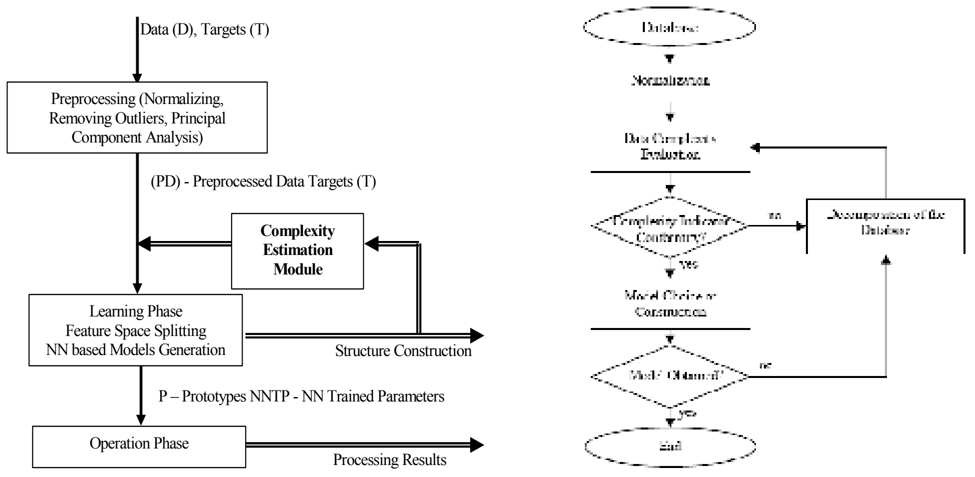

Data (D), Targets (T)
Preprocessing (Normalizing, Removing Outliers, Principal Component Analysis)
(PD) - Preprocessed Data Targets (T)
Complexity Estimation Module
Learning Phase Feature Space Splitting NN based Models Generation
Structure Construction
P – Prototypes NNTP - NN Trained Parameters
Operation Phase
Processing Results
Database
Normalization
Data Complexity Evaluation
Complexity Indicator Conformity?
no
Decomposition of the Database
yes
Model Choice or Construction
Model Obtained?
no
yes
End

Fig. 13. Bloc scheme of T-DTS: *Left* – Modular concept, *Right* – Algorithmic concept

software architecture is depicted on Figure 14. T-DTS software incorporates three databases: decomposition methods, ANN models and complexity estimation modules databases.

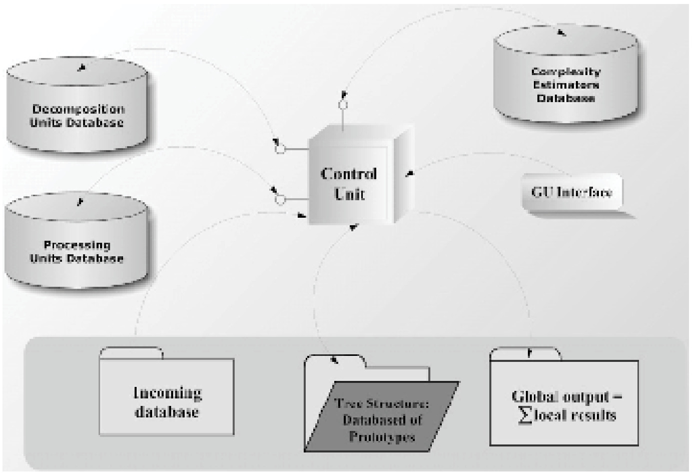

Decomposition Units Database
Complexity Estimators Database
Control Unit
GU Interface
Processing Units Database
Incoming database
Tree Structure: Databased of Prototypes
Global output = ∑local results

Fig. 14. T-DTS software architecture

T-DTS software engine is the *Control Unit.* This core-module controls and activates several software packages: normalization of incoming database (if it's required), splitting and building a tree of prototypes using selected decomposition method, sculpting the set of local results and generating global result (learning and generalization rates). T-DTS software can be seen as a *Lego system* of decomposition methods, processing methods powered by a control engine an accessible by operator thought Graphic User Interface.
The three databases can be independently developed out of the main frame and more important, they can be easily incorporated into T-DTS framework.
For example, SOM-LSVMDT (Mehmet et al., 2003) algorithm; which is based on the same idea of decomposition, can be implement by T-DTS by mean of LSVMDT (Chi & Ersoy, 2002) (Linear Support Vector Machine Decision Tree) processing method incorporation into PU database.

- The current T-DTS software (version 2.02) includes the following units and methods:
  - ✓ Decomposition Units:
  - ✓ CN (Competitive Network)
  - ✓ SOM (Self Organized Map)
  - ✓ LVQ (Learning Vector Quantization)
- Processing Units:
  - ✓ LVQ (Learning Vector Quantization)
  - ✓ Perceptrons
  - ✓ MLP (Multilayer Perceptron)
  - ✓ GRNN (General Regression Neural Network)
  - ✓ RBF (Radial basis function network)
  - ✓ PNN (Probabilistic Neural Network)
  - ✓ LN
- Complexity estimators (Bouyoucef, 2007), presented in sub-section 4.2.5, are based on the following criteria:
  - ✓ MaxStd (Sum of the maximal standard deviations)

- ✓ Fisher measure.
- ✓ Purity measure
- ✓ Normalized mean distance
- ✓ Divergence measure
- ✓ Jeffries-Matusita distance
- ✓ Bhattacharyya bound
- ✓ Mahalanobis distance
- ✓ Scattered-matrix method based on inter-intra matrix-criteria (Fukunaga, 1972).
- ✓ ZISC© IBM ® based complexity indicator (Budnyk & al. 2007).

4.2 T-DTS learning and decomposition mechanism

The decomposition mechanism in T-DTS approach builds a tree structure. The creation of decomposition tree is data-driven. It means that the decision to-split-or-not and how-to-split is made depending on the properties of the current sub-database. For each database the decision to-split-or-not should be made. After a positive decision a Decomposition Unit (DU) is created which divides the data and distributes the resulting sub-databases creating children in the tree. If the decision is negative the decomposition of this sub-database (and tree branch) is over and a Processing Unit should be built for the sub-database. The type of the new tree module depends on the result of decomposition decision made for the current sub-database (and in some cases also on other parameters, as described later). The tree is built beginning from the root which achieves the complete learning database. The process results in a tree which has DUs at nodes and Processing Unit models in tree leaves.
Figure 15 shows decomposition tree structure (in case of binary tree) and its recurrent construction in time, while question marks mean decomposition decisions.
For any database B (including the initial) a splitting decision (if to split and how to split) is taken. When the decision is positive then a Decomposition Unit is created, and the database is decomposed (clustered) by the new Decomposition Unit. When the decomposition decision is negative, a Processing Unit is created in order to process the database (for example to create a model).
The database B incoming to some Decomposition Unit will be split into several sub-databases $b_1, b_2 ... b_k$, depending on the properties of the database B and parameters τ obtained from controlling structure. The function $S(\psi_i,\tau)$ assigns any vector ψ_i from database B to an appropriate sub-database *j*. The procedure is repeated in recursive way i.e. for each resulting sub-database a decomposition decision is taken and the process repeats. One chain of the process is depicted in figure 16.

$$S(\Psi_i,\tau,\xi) = (s_1 ... s_k ... s_M)^T \quad \textbf{with} \quad \begin{array}{ll} s_k = 1 & \text{if } \tau = \tau_k \text{and} \xi = \xi_k \\ s_k = 0 & \text{else} \end{array} \tag{4}$$

The scheduling vector $S(\psi_i,\tau_k)$ will activate (select) the *K*-th Processing Unit, and so the processing of an unlearned input data conform to parameter τ_k and condition ξ_k will be given by the output of the selected Processing Unit:

$$Y(\Psi_i) = Y_k(i) = F_k(\Psi_i) \tag{5}$$

Complexity indicators are used in our approach in order to reach one of the following goals:

- Global decomposition control - estimator which evaluates the difficulty of classification of the whole dataset and chooses decomposition strategy and parameters before any decomposition has started,
- Local decomposition control - estimator which evaluates the difficulty of classification of the current sub-database during decomposition of dataset, in particular:
  - ✓ Estimator which evaluates the difficulty of classification of the current sub-database, to produce decomposition decision (if to divide the current sub-database or not);
  - ✓ Estimator which can be used to determine the type of used classifier or its properties and parameters.
- Mixed approach - use of many techniques mentioned above at once, for example: usage of Global decomposition control to determine the parameters of local decomposition control.

One should mention also that estimation of sub-database complexity occurs for each sub-database dividing decision thus computational complexity of the algorithm should rather be small. Thus it doesn't require advanced complexity estimation methods. Considering these features, the second option - estimation during decomposition - has been chosen in our experiments in order to achieve self adaptation feature of T-DTS structure.

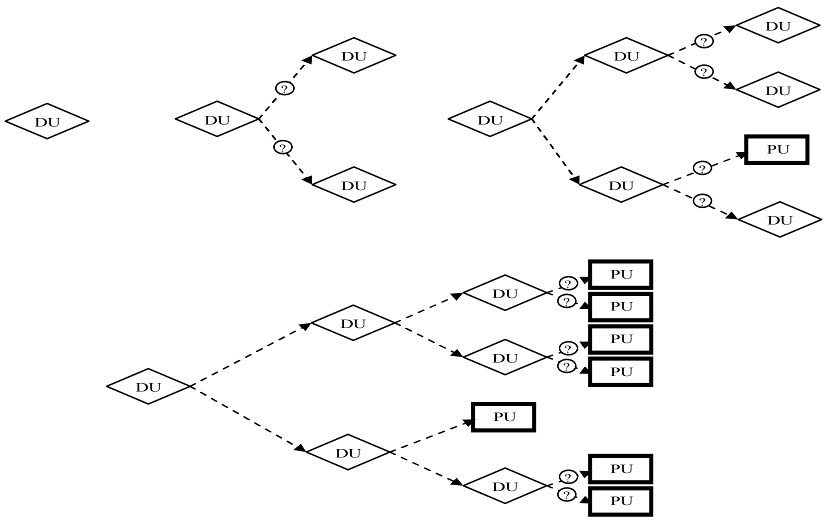


Fig. 15. T-DTS decomposition tree creation in time

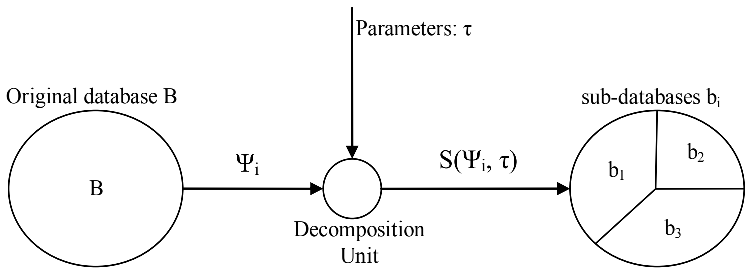


Fig. 16. Decomposition Unit activities

4.2.1 Decomposition Unit (DU)

The purpose of Decomposition Unit is to divide the database into several sub-databases. This task is referred in the literature as clustering (Hartgan, 1975). To accomplish this task a plenty of methods are known. We are using Vector Quantization unsupervised methods, in particular: competitive Neural Networks and Kohonen Self-Organizing Maps (Kohonen, 1984). These methods are based on prototype, that represent the centre of cluster (cluster = group of vectors). In our approach cluster is referred to as sub-database.

4.2.2 Decomposition of learning database

The learning database is split into M learning sub-databases by DUs during building of the decomposition tree. The learning database decomposition is equivalent to "following the decomposition tree" decomposition strategy. The resulting learning sub-databases could be used for Processing Unit learning. Each sub-database has then Processing Unit attached. The Processing Unit models are trained using the corresponding learning sub-database.

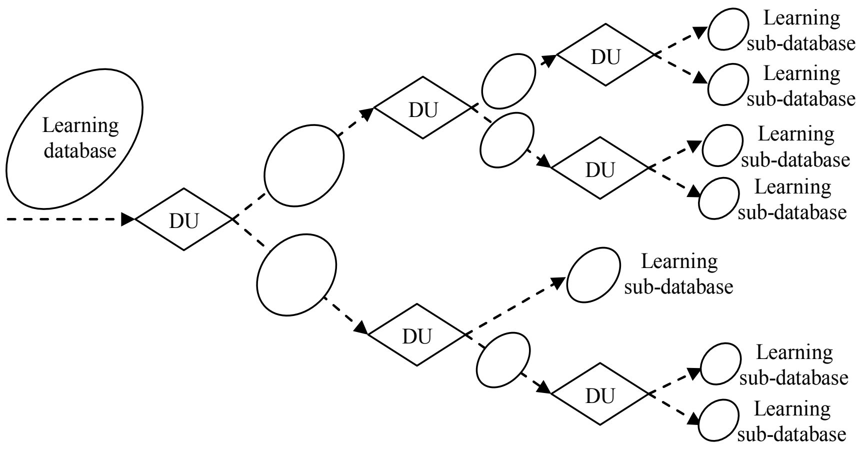


Fig. 17. Decomposition of learning database "following the decomposition tree" strategy

4.2.3 Training of Processing Units (models)

For each sub-database T-DTS constructs a neural based model describing the relations between inputs and outputs. Training of Processing Unit models is performed using standard supervised training techniques, possibly most appropriate for the learning task required. In this work only Artificial Neural Networks are used, however there should be no difficulty to use other modelling techniques.

Processing Unit is provided with a sub-database and target data. It is expected to model the input/output mapping underlying the subspace as reflected by the sub-database provided. The trained model is used later to process data patterns assigned to the Processing Unit by assignment rules.

4.2.4 Processing Units

Processing Unit models used in our approach can be of any origin. In fact they could be also not based on Artificial Neural Networks at all. The structure used depend on the type of learning task, we use:

- for classification - MLP, LVQ, Probabilistic Networks (Haykin, 1999), RBF, Linear Networks;
- for regression - MLP, RBF;
- for model identification - MLP.

Processing Unit models are created and trained in the learning phase of T-DTS algorithm, using learning sub-databases assigned by decomposition structure. In the generalization phase, they are provided with generalization vectors assigned to them by pattern assignment rules. The vectors form generalization sub-databases are processed by Processing Unit models. Each Processing Unit produce some set of approximated output vectors, and the ensemble of them will compose whole generalization database.

4.2.5 Complexity estimation techniques

The goal of complexity estimation techniques is to estimate the processing task's difficulty. The information provided by these techniques is mainly used in a splitting process according to a divide and conquer approach. It act's at three levels:

- The task decomposition process up to some degree dependant on task or data complexity.
- The choice of appropriate processing structure (i.e. appropriated model) for each subset of decomposed data.
- The choice of processing architecture (i.e. models parameters).

The techniques usually used for complexity estimation are sorted out in three main categories: those based on Bayes error estimation, those based on space partitioning methods and others based on intuitive paradigms. Bayes error estimation may involve two classes of approaches, known as: *indirect* and *non-parametric* Bayes error estimation methods, respectively. This sub-section of the chapter will present a detailed summery of these main complexity estimation methods which are used in the T-DTS self-organizing system core, focusing mainly on measurements supporting task decomposition aspect.

4.2.5.1 Indirect Bayes error estimation

To avoid the difficulties related to direct estimation of the Bayes error, an alternative approach is to estimate a measure directly related to the Bayes error, but easier to compute. Usually one assumes that the data distribution is normal (Gaussian). Statistical methods grounded in the estimation of probability distributions are most frequently used. The drawback of these is that they assume data normality. A number of limitations have been documented in literature (Vapnik, 1998):

- model construction could be time consuming;
- model checking could be difficult;
- as data dimensionality increases, a much larger number of samples is needed to estimate accurately class conditional probabilities;
- if sample does not sufficiently represent the problem, the probability distribution function cannot be reliably approximated;
- with a large number of classes, estimating a priori probabilities is quite difficult. This can be only partially overcome by assuming equal class probabilities (Fukunaga, 1990), (Ho & Basu, 2002).
- we normally do not know the density form (distribution function);
- most distributions in practice are multimodal, while models are unimodal;
- approximating a multimodal distributions as a product of univariate distributions do not work well in practice.

4.2.5.1.1 Normalized mean distance

Normalized mean distance is a very simple complexity measure for Gaussian unimodal distribution. It raises when the distributions are distant and not overlapping.

$$d_{norm} = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{6}$$

The main drawback of that estimator is that it is inadequate (as a measure of separability) when both classes have the same mean values.

4.2.5.1.2 Chernoff bound

The Bayes error for the two class case can be expressed as:

$$\varepsilon = \int \min_i \left[P(c_k) p\left(x \mid c_k\right) \right] dx \tag{7}$$

Through modifications, we can obtain a Chernoff bound ε_u, which is an upper bound on ε for the two class case:

$$\varepsilon_u = P(c_1)^s P(c_2)^{1-s} \int p(x^s \mid c_1) p(x^{1-s} \mid c_2) dx \text{ for } 0 \le s \le 1 \tag{8}$$

The tightness of bound varies with s.

4.2.5.1.3 Bhattacharyya bound

The Bhattacharyya bound is a special case of Chernoff bound for s = 1/2. Empirical evidence indicates that optimal value for Chernoff bound is close to 1/2 when the majority of separation comes from the difference in class means. Under a Gaussian assumption, the expression of the Bhattacharyya bound is:

$$\varepsilon_b = \sqrt{P(c_1)P(c_2)} e^{-\mu(1/2)} \tag{9}$$

where:

$$\mu(1/2) = \frac{1}{8}(\mu_2 - \mu_1)^T \left[\frac{\Sigma_1 + \Sigma_2}{2} \right]^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \ln \frac{\left| \frac{\Sigma_1 + \Sigma_2}{2} \right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \tag{10}$$

and μ_i and Σ_i are respectively the means and classes covariance's (i in {1,2}).

4.2.5.1.4 Mahalanobis distance

Mahalanobis distance (Takeshita et al., 1987) is defined as follows:

$$M_D = \left(\mu_2 - \mu_1\right)^T \Sigma^{-1} \left(\mu_2 - \mu_1\right) \tag{11}$$

M_D is the Mahalanobis distance between two classes. The classes' means are μ_1 and μ_2 and Σ is the covariance matrix. Mahalanobis distance is used in statistics to measure the similarity of two data distributions. It is sensitive to distribution of points in both samples. The Mahalanobis distance is measured in units of standard deviation, so it is possible to assign statistical probabilities (that the data comes from the same class) to the specific measure values. Mahalanobis distance greater than 3 is considered as a signal that data are not homogenous (does not come from the same distribution).

4.2.5.1.5 Jeffries-Matusita distance

Jeffries-Matusita (Matusita ,1967) distance between class's c_1 and c_2 is defined as:

$$JM_D = \int_x \left\{ \sqrt{p(X \mid c_2)} - \sqrt{p(X \mid c_1)} \right\}^2 dx \tag{12}$$

If class's distributions are normal Jeffries-Matusita distance reduces to:

$$JM_D = 2\left(1 - e^{-\alpha}\right), \text{ where} \tag{13}$$

$$\alpha = \frac{1}{8}\left(\mu_2 - \mu_1\right)^T \left(\frac{\Sigma_2 + \Sigma_1}{2}\right)^{-1} \left(\mu_2 - \mu_1\right) + \frac{1}{2}\log_e\left(\frac{\det \Sigma}{\det \Sigma_1 - \det \Sigma_2}\right) \tag{14}$$

Matusita distance is bounded within range [0, 2] where high values signify high separation between c_1 and c_2 classes.

4.2.5.2 Non-Parametric Bayes error estimation and bounds

Non-parametric Bayes error estimation methods make no assumptions about the specific distributions involved. They use some intuitive methods and then prove the relation to Bayes error. Non-parametric techniques do not suffer from problems with parametric techniques.

4.2.5.2.1 Error of the classifier itself

This is the most intuitive measure. However it varies much depending on the type of classifier used and, as such, it is not very reliable unless one uses many classification methods and averages the results. The last solution is certainly not computationally efficient.

4.2.5.2.2 k-Nearest Neighbours, (k-NN)

K- Nearest Neighbours (Cove & Hart, 1967) technique relays on the concept of setting a local region *Γ(x)* around each sample *x* and examining the ratio of the number of samples enclosed *k* to the total number of samples *N*, normalized with respect to region volume *v*:

$$\widehat{p}(x) = \frac{k(x)}{vN} \tag{15}$$

K-NN technique fixes the number of samples enclosed by the local region (*k* becomes constant). The density estimation Equation for k-NN becomes:

$$\widehat{p}(x) = \frac{k\text{ - }1}{v(x)N} \tag{16}$$

where *p(x)* represent probability of specific class appearance and *v(x)* represent local region volume. K-NN is used to estimate Bayes error by either providing an asymptotic bound or through direct estimation. K-NN estimation is computationally complex.

4.2.5.2.3 Paren Estimation

Parzen techniques relay on the same concept as k-NN: setting a local region *Γ(x)* around each sample *x* and examining the ratio of the samples enclosed *k*, to the total number of samples *N*, normalized with respect to region volume *v*:

$$\widehat{p}(x) = \frac{k}{vN} \tag{17}$$

The difference according to k-NN is that Parzen fixes the volume of local region *v*. Then the density estimation equation becomes:

$$\widehat{p}(x) = \frac{k(x)}{vN} \tag{18}$$

where *p(x)* represents density and *k(x)* represents number of samples enclosed in volume. Estimating the Bayes error using the Parzen estimate is done by forming the log likelihood ratio functions based upon the Parzen density estimates and then using resubstitution and leave-one-out methodologies to find an optimistic and pessimistic value for error estimate. Parzen estimates are however not known to bound the Bayes error. Parzen estimation is computationally complex.

4.2.5.2.4 Boundary methods

The boundary methods are described in the work of Pierson (Pierson, 1998). Data from each class is enclosed within a boundary of specified shape according to some criteria. The boundaries can be generated using general shapes like: ellipses, convex hulls, splines and others. The boundary method often uses ellipsoidal boundaries for Gaussian data, since it is a natural representation of those. The boundaries may be made compact by excluding outlying observations. Since most decision boundaries pass through overlap regions, a count of these samples may give a measure related to misclassification rate. Collapsing boundaries iteratively in a structured manner and counting the measure again lead to a series of decreasing values related to misclassification error. The rate of overlap region decay provides information about the separability of classes. Pierson discuses in his work a way in which the process from two classes in two dimensions can be expanded to higher dimension with more classes. Pierson has demonstrated that the measure of separability called the Overlap Sum is directly related to Bayes error with a much more simple computational complexity. It does not require any exact knowledge of the *a posteriori* distributions. Overlap Sum is the arithmetical mean of overlapped points with respect to progressive collapsing iterations:

$$O_S(mt_0) = \frac{1}{N}\sum_{k=1}^{m}(kt_0)\Delta s(kt_0) \tag{19}$$

where t_0 is the step size, m is the maximum number of iteration (collapsing boundaries), N is the number of data points in whole dataset and *Δ*s(kt$_0$) is the number of points in the differential overlap.

4.2.5.3 Measures related to space partitioning

Measures related to space partitioning are connected to space partitioning algorithms. Space partitioning algorithms divide the feature space into sub-spaces. That allows obtaining some advantages, like information about the distribution of class instances in the sub-spaces. Then the local information is globalized in some manner to obtain information about the whole database, not only the parts of it.

4.2.5.3.1 Class Discriminability Measures

Class Discriminability Measure (CDM) (Kohn et al., 1996) is based on the idea of inhomogeneous buckets. The idea here is to divide the feature space into a number of hypercuboids. Each of those hypercuboids is called a "box". The dividing process stops when any of following criteria is fulfilled:

- box contains data from only one class;
- box is non-homogenous but linearly separable;
- number of samples in a box is lower that $N^{0.375}$, where N is the total number of samples in dataset.

If the stopping criteria are not satisfied, the box is partitioned into two boxes along the axis that has the highest range in terms of samples, as a point of division using among others median of the data.
Final result will be a number of boxes which can be:

- homogenous terminal boxes (HTB);
- non-linearly separable terminal boxes (NLSTB);
- non-homogenous non-linearly separable terminal boxes (NNLSTB).

In order to measure complexity, CDM uses only Not Linearly Separable Terminal Boxes, as, according to author (Kohn et al., 1996), only these contribute to Bayes error. That is however not true, because Bayes error of the set of boxes can be greater than the sum of Bayes errors of the boxes - partitioning (and in fact nothing) cannot by itself diminish the Bayes error of the whole dataset; however it can help classifiers in approaching the Bayes error optimum. So given enough partitions we arrive to have only homogenous terminal boxes, so the Bayes error is supposed to be zero, that is not true.
The formula for CDM is:

$$CDM = \frac{1}{N}\sum_{i=1}^{M}\{k(i) - \max[k(j\,|\,i)\} \tag{20}$$

where k(i) is the total number of samples in the *i*-th NNLSTB, k(j | i) is the number of samples from class j in the *i*-th NNLSTB and N is the total number of samples. For task that lead to only non-homogenous but linearly separable boxes, this measure equals zero.

4.2.5.3.2 Purity measure

Purity measure (Sing, 2003) is developed by Singh and it is presented with connection to his idea based on feature space partitioning called PRISM (Pattern Recognition using Information Slicing Method). PRISM divides the space into cells within defined resolution B. Then for each cell probability of class *i* in cell *l* is:

$$p_{il} = \frac{n_{il}}{\sum_{j=1}^{K_l} n_{jl}} \tag{21}$$

where n_{jl} is the number of points of class *j* in cell *l*, n_{il} is the number of points of class *i* in cell *l* and K_l is the total number of classes.

Degree of separability in cell *l* is given by:

$$S_{H(l)} = \sqrt{\left(\frac{k}{k\quad 1}\right)\sum_{i=1}^{k}\left(p_{il}\quad {}^{1}\!/_{k}\right)^{2}} \tag{22}$$

These values are averaged for all classes, obtaining overall degree of separability:

$$S_H = \sum_{l=1}^{H_{total}} S_{H(l)} \frac{N^l}{N} \tag{23}$$

where N^l signifies the number of points in the *l*-th cell, and *N* signifies total number of points. If this value was computed at resolution B then it is weighted by factor $w = \frac{1}{2^B}$ for B=(0,1,...31). Considering the curve (S_H versus normalized resolution) as a closed polygon with vertices (x_i,y_i), the area under the curve called **purity** for a total of *n* vertices is given as:

$$AS_H = \frac{1}{2}\left(\sum_{i=1}^{n-1}\left(x_i y_{i+1} - y_i x_{i+1}\right)\right) \tag{24}$$

The *x* axis is scaled to achieve values bounded within range [0, 1]. After the weighing process maximum possible value is 0.702, thus the value is rescaled once again to be between [0, 1] range.
The main drawback of purity measure is that if in a given cell, the number of points from each class is equal, then the purity measure is zero despite that in fact the distribution may be linearly separable. Purity measure does not depend on the distribution of data in space of single cell, but the distribution of data into the cells is obviously associated with data distribution.

4.2.5.3.3 Neighborhood Separability

Neighborhood Separability (Singh, 2003) measure is developed by Singh. Similarly to purity, it also depends on the PRISM partitioning results. In each cell, up to *k* nearest neighbors are found. Then one measure a proportion p_k of nearest neighbors that come from the same class to total number of nearest neighbors. For each number of neighbors *k*, $1<=k<=\lambda_{il}$ calculate the area under the curve that plots p_k against *k* as φ_j. Then compute the average proportion for cell H_l as:

$$p_l = \frac{1}{N^l}\sum_{j=1}^{N_l}\phi_j \tag{25}$$

Overall separability of data is given as:

$$S_{NN} = \sum_{l=1}^{H_{total}} p_l \frac{N^l}{N} \tag{26}$$

One compute the S_{NN} measure for each resolution B=(0, 1, … ,31). Finally, the area AS_{NN} under the curve S_{NN} versus resolution gives the measure of **neighborhood separability** for a given data set.

4.2.5.3.4 Collective entropy

Collective entropy (Singh & Galton, 2002), (Singh, 2003) measure the degree of uncertainty. High values of entropy represent disordered systems. The measure is connected to data partitioning algorithm called PRISM.
Calculate the entropy measure for each cell H_l:

$$E_l = \sum_{i=1}^{K_l} \left(p_{il} \quad \log(p_{il}) \right) \tag{27}$$

Estimate overall entropy of data as weighted by the number of data in each cell:

$$E = \sum_{l=1}^{H_{total}} E_l \cdot \frac{N^l}{N} \tag{28}$$

Collective entropy for data at given partition resolution is defined as:

$$E_C = 1 - E \tag{29}$$

This is to keep consistency with other measures: maximal value of 1 signifies complete certainty and minimum value of 0 uncertainty and disorder.
Collective entropy is measured at multiple partition resolutions B=(0,…31) and scaled by factor $w = 1/2^B$ to promote lower resolution. Area under the curve of Collective Entropy versus resolution gives a measure of uncertainty for a given data set. That measure should be scaled as $AS_E = {AS_E}/{0.702}$ to keep the values in [0,1] range.

4.2.5.4 Other Measures

The measures described here are difficult to classify as they are very different in idea and it's difficult to distinguish common properties.

4.2.5.4.1 Correlation-based approach

Correlation-based approach (Rahman & Fairhurst, 1998) is described by Rahman and Fairhust. In their work, databases are ranked by the complexity of images within them. The degree of similarity in database is measured as the correlation between a given image and the remaining images in database. It indicates how homogenous the database is. For separable data, the correlation between data of different classes should be low.

4.2.5.4.2 Fisher discriminant ratio

Fisher discriminant ratio (Fisher, 2000) originates from Linear Discriminant Analysis (LDA). The idea of linear discriminant approach is to seek a linear combination of the variables which separates two classes in best way. The Fisher discriminant ratio is given as:

$$f_1 = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{30}$$

where μ_1, μ_2, σ_1, σ_2 are the means and variances of two classes respectively. The measure is calculated in each dimension separately and afterwards the maximum of the values is taken. It takes values from $[0,+\infty]$; high value signifies high class separability. To use it for multi class problem it is necessary however to compute Fisher discriminant ratios for each two-element combination of classes and later average the values.

Important feature of the measurement is that it is strongly related to data structure. The main drawback is that it acts more like a detector of linearly separable classes than complexity measure. The advantage is very low computational complexity.

4.2.5.4.3 Interclass distance measures

The ***interclass distance measures*** (Fukunaga, 1990) are founded upon the idea that class separability increases as class means separate and class covariance's become tighter. We define:
Within-class scatter matrix:

$$S_w = \sum_{i=1}^{L} P(\omega_i)\Sigma_i \tag{31}$$

Between-class scatter matrix:

$$S_b = \sum_{i=1}^{L} P(\omega_i)(\mu_i - \mu_0)(\mu_i - \mu_0)^T \tag{32}$$

Mixture (total) scatter matrix:

$$S_m = S_w + S_b \tag{33}$$

where μ_i are class means, $P(c_i)$ are the class probabilities, Σ_i are class covariance matrices, and $\mu_0 = \sum_{i=1}^{L} P(\omega_i)\mu_i$ is the mean of all classes.

Many intuitive measures of class separability come from manipulating these matrices which are formulated to capture the separation of class means and class covariance compactness. Some of the popular measures are:

$$J_1 = tr(S_2{}^{-1}S_1),\ J_2 = \ln\left|S_2{}^{-1}S_1\right|,\ J_3 = \frac{tr(S_1)}{tr(S_2)} \tag{34}$$

where S_1, S_2 are a tuple from among { S_b, S_w, S_m}, and *tr* signifies matrix trace. Frequently many of these combinations and criteria result in the same optimal features.

4.2.5.4.4 Volume of the overlap region

We can find volume of the overlap region (Ho & Baird, 1998) by finding the lengths of overlapping of two classes' combination across all dimensions. The lengths are then divided by overall range of values in the dimension (normalized), where d_o represents length of overlapping region, d_{max} and d_{min} represent consequently maximum and minimum feature values in specified dimension:

$$r_d = \frac{d_o}{d_{\max} - d_{\min}} \tag{35}$$

Resulting ratios are multiplied across all dimensions *dim* to achieve volume of overlapping ratio for the 2-class case (normalized with respect to feature space)

$$v_o = \prod_{i=1}^{\dim} r_d \tag{36}$$

It should be noted that the value is zero as long as there is at least one dimension in which the classes don't overlap.

Technique	**Relation to Bayes error**	**Computing cost**	**Probability density functions**	**Number of classes**
Chernoff bound	Yes	High	needed	2
Bhattacharyya bound	Yes	Medium	needed	2
Divergence	Yes	High	needed	2
Mahalanobis distance	Yes	Medium	not needed	2
Matusita distance	Yes	High	needed	2
Entropy measures	No	High	needed	>2
Classifier error	Potential	Depends on the classifier used		
k-Nearest Neighbours	Yes	High	not needed	>2
Parzen estimation	No	High	not needed	>2
Boundary methods	Yes	Medium	not needed	2
Class Discriminability Measures	No	High	not needed	2
Purity	No	High	not needed	>2
Neighbourhood separability	No	High	not needed	>2
Collective entropy	No	High	not needed	2
Correlation based approach	No	High	not needed	>2
Fisher discriminant ratio	No	very low	not needed	2
Interclass distance measures	No	Low	not needed	>2
Volume of the overlap region	No	Low	not needed	2
Feature efficiency	No	Medium	not needed	2
Minimum Spanning Tree	No	High	not needed	>2
Inter-intra cluster distance	No	High	not needed	2
Space covered by epsilon neighbourhoods	No	High	not needed	>2
Ensemble of estimators	Potential	High	depends	Depends

Table 3. Comparison of Classification Complexity Techniques

4.2.6 Discussion

Classification complexity estimation methods present great variability. The methods which are derived from Bayes error are most reliable in terms of performance, as they are theoretically stated. The most obvious drawback is that they have to do assumptions about a priori probability distributions. If the advantage of the methods designed using experimental (empirical) basis is that they are based uniquely on experimental data and do not need probability density estimates of distributions, these methods are as various as those relating the Bayes error's estimation and their performance are difficult to predict. Some methods are designed only for two-class problems, and as such they need special procedures to accommodate them to multi-class problem (like counting the average of complexities of all two-class combinations). The table 3, comparing complexity estimation methods, is aimed at several specific aspects which are:

- **Relation with Bayes error** which could be seen as a proof of estimator's accuracy up to some point;
- **Computational Cost**, this is especially important when the measurements are taken many times during the processing of problem, as in T-DTS case;

- **Need for probability density function estimates**;
- **Number of classes** in classification problem for which the method can be applied directly.

Recently, a number of investigations pushed forward the idea to combine several complexity estimation methods: for example by using a weighted average of them (Bouyoucef, 2007). It is possible that a single measure of complexity be not suitable for practical applications; instead, a hierarchy of estimators may be more appropriate (Maddox, 1990).

Using complexity estimation techniques based splitting regulation, T-DTS is able to reduce complexity on both data and processing chain levels (Madani et Al., 2003). It constructs a treelike evolutionary architecture of models, where nodes (DU) are decision units and leaves correspond to Neural Network - based Models (Processing Unit). That results in splitting the learning database into set of sub-databases. For each sub-database a separate model is built.

This approach presents numerous advantages among which are:

- simplification of the treated problem - by using a set of simpler local models;
- parallel processing capability - after decomposition, the sub-databases can be processed independently and joined together after processing;
- task decomposition is useful in cases when information about system is distributed locally and the models used are limited in strength in terms of computational difficulty or processing (modeling) power;
- modular structure gives universality: it allows using of specialized processing structures as well as replacing Decomposition Units with another clustering techniques;
- classification complexity estimation and other statistical techniques may influence the parameters to automate processing, i.e., decompose automatically;
- automatic learning.

However, our approach is not free of some disadvantages:

- if the problem doesn't require simplification (problem is solved efficiently with single model) then Task Decomposition may decrease the time performance, as learning or executing of some problems divided into sub-problems is slower than learning or executing of not split problem; especially if using sequential processing (in opposition to parallel processing);
- some problems may be naturally suited to solve by one-piece model - in this case splitting process should detect that and do not divide the problem;
- too much decomposition leads to very small learning sub-databases. Then they may loss of generalization properties. In extreme case leading to problem solution based only on distance to learning examples, so equal to nearest-neighbor classification method.

In the following section, we study the efficiency of T-DTS approach when dealing with classification problems.

4.2.7 Implementation and validation results

In order to validate the T-DTS self-organizing approach, we present in this section the application of such a paradigm to three complex problems. The first one concerns a pattern recognition problem. The second and third one are picked from the well know UCI repository: a toy problem (Tic-Tac-Toe) for validation purpose and a DNA classification one.

4.2.7.1 Application to UCI Reprository

Complexity estimating plays key-role in decomposition and tree-building process. In order to evaluate and validate T-DTS approach, we use two benchmarks from the UCI Machine Learning Repository (Bouyoucef, 2007). These two benchmarks are:

1. Tic-tac-toe end-game problem. The problem is to predict whether each of 958 legal endgame boards for tic-tac-toe is won for `x'. The 958 instances encode the complete set of possible board configurations at the end of tic-tac-toe. This problem is hard for the covering family algorithm, because of multi-overlapping.
2. Splice-junction DNA Sequences classification problem. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as EI sites), and recognizing intron/exon boundaries (IE sites). There are 3190 numbers of instances from Genbank 64.1, each of them compound 62 attributes which defines DNA sequences (ftp-site: ftp://ftp.genbank.bio.net) problem.

Next subsections include description of experimental protocol.

4.2.7.2 Experimental protocol

In the first case, Tic-tac-toe end game, we have used 50% of database for learning purpose and 50% for generalization purpose. At the node level (DU), competitive networks perform the decomposition. The following complexity estimation methods have been used: Mahalanobis, ZISC and Normalized mean. At T-DTS leaf level we have applied PU - LVQ.

Method type	*Max G_r (± Std. Dev.) (%)*
IB3-CI	99.1
CN2 standard	98.33 (± 0.08)
IB1	98.1
Decision Tree (DT)+FICUS	96.45 (± 1.68)
3-Nearest neighbor algorithm+FICUS	96.14 (± 2.03)
MBRTalk	88.4
Decision Tree (DT) Learning Concept	85.38 (± 2.18)
T-DTS&Mahalanobis com. est.	**84.551 (± 4.592)**
NewID	84.0
CN2-SD (add. weight.)	83.92 (± 0.39)
T-DTS&ZISC based com. est.	**82.087 (± 2.455)**
IB3	82.0
Back propagation +FICUS	**81.66 (± 14.46)**
T-DTS&Normalized mean com. est.	**81.002 (±1.753)**
7-Nearest neighbor	76.36 (± 1.87)
CN2-WRAcc	70.56 (± 0.42)
3-Nearest neighbor	67.95 (± 1.82)
Back propagation	**62.90 (± 3.88)**
Perceptron+FICUS	37.69 (± 3.98)
Perceptron	34.66 (± 1.84)

Table 4. Tic-tac-toe endgame problem

Method type	*Max G_r (± Std. Dev.) (%)*
3-Nearest neighbor algorithm+**FICUS**	86.30 (± 4.96)
Perceptron+**FICUS**	83.96 (± 6.22)
Decision Tree (DT)+**FICUS**	83.78 (± 4.61)
Back propagation algorithm+**FICUS**	83.42 (± 7.73)
T-DTS&ZISC based com. Est	**80.084 (± 3.176)**
3-Nearest neighbor algorithm	79.18 (± 6.32)
T-DTS&Mahalanobis based com. Est	78.672 (± 4.998)
Perceptron	76.34 (± 6.71)
T-DTS & Jeffries-Matusita based c.e.	75.647 (±8.665)
Decision Tree (DT) Learning Concept	73.55 (± 5.88)

Table 5. Splice-junction DNA sequences classification test

For DNA Benchmark, we have used 20% of database for learning purpose and 80% for generalization purpose. At the node level competitive networks perform the decomposition. The following complexity estimation methods have been used: Mahalanobis, Bhattacharya, ZISC, Purity and Fisher Measure. At T-DTS leaf level we applied PU - MLP.

For DNA Benchmark, we have used 20% of database for learning purpose and 80% for generalization purpose. At the node level competitive networks perform the decomposition. The following complexity estimation methods have been used: Mahalanobis, Bhattacharya, ZISC, Purity and Fisher Measure. At T-DTS leaf level we applied PU - MLP.

For both cases, a manual optimization has been performed. We have selected the decomposition units, the complexity estimation methods and the processing units that allow us to reach the highest performances in terms of generalization rate. In the next subsection, we present the results and compare them to those obtained by other approaches, mainly based on decision tree algorithms.

4.2.7.3 Results presentation and discussion

Various experiments have been conducted according to the experimental protocol described previously. Table 4 and Table 5 consolidate the results of our experiments and the results obtained by other classification approaches (Lavrac et al,. 2002), (Aha, 1991), (Markovitch & Rosenstein, 2002). As it is shown, we have resolved Tic-tac-toe endgame classification task with respectively 84.55%, 82.09% and 81.00% of generalization rates using *Mahalanobis,* ZISC and Normalized mean complexity estimators with a standard deviation of 4.59%, 2.46% and 1.75%. Taking into account standard deviation ratio, we can state that these results are equivalent as they are in the same range.

IB3-CI, CN2, IB1, DT and MBRTalk algorithms are rely on the instances extracting and their extrapolation. So, they are well adapted to board game problems. They also use domain knowledge to reach very high generalization rates (around 95%). Methods associated to FICUS use hypothesis driven construction strategies and especially FICUS algorithms allows to enhance the learning data base size.

In our case, T-DTS uses only data driven strategy. So, as we can see in Table 5, for Splice-junction DNA Sequences benchmark, taking into account the generalization rate standard deviation, the leading algorithms exhibit the same performances (3-Nearest neighbor+**FICUS,** Perceptron+**FICUS,** DT+**FICUS,** Back propagation **+FICUS and T-DTS&ZISC**). So, without using specific domain knowledge, T-DTS reaches a high generalization rate. The T-DTS strength is its ability to solve hard classification problems without need of domain specific knowledge. In the experiments described in this paper, T-DTS structure optimization has been conducted manually (by the user). This is the main drawback.

5. Conclusion

Due the complexity of the actual systems based on heterogeneous methods, artificial neural networks approaches can reduce this complexity by modeling the environment as stochastic. Algorithms based on Neural Networks can take into account the dynamics of these environments with no model of dynamics to be given. Main idea of the approaches developed in this chapter is to take advantage from distributed processing and task simplification by dividing an initially complex processing task into a set of simpler subtasks using complexity estimation based loop to control the splitting process. An appealing consequence of combining complexity estimation based splitting and artificial neural networks based processing techniques is decreasing of user's intervention in specifying processing parameters. A first modular structure is proposed. We have focused our attention in some special kind of Constrained Based Routing in wired networks which we called QoS self-optimization Routing. In a second part, we study the use of T-DTS self-organizing and task adaptive abilities. Beside complexity estimation based self-organization and adaptation abilities of our approach, the neural nature of generated models leads to additional attractive features which are modularity and some universality of the issued processing system, opening new dimensions in bio-inspired artificial intelligence. Moreover, the distributed nature of T-DTS makes the processing phase potentially realizable using either parallel machine or network of sequential machines. Very promising results, obtained from experimental validation, involving either the presented set of classification benchmarks (problems) or the reported pattern recognition dilemma, show efficiency of such self-organizing multiple models' generator to enhance global and local processing capabilities by reducing complexity on both processing and data levels.

6. Acknowlegments

The present work has been partially supported by French Ministry of High Education and Research. A part of this project has also benefit from the French Eiffel Excellence Program of EGIDE.

7. References

Aha. D. W. (1991). Incremental Constructive Induction: An Instance-Based Approach, *Proceedings of the Eight International Workshop on Machine Learning, Morgan Kaufmann.*

Arbib (1989). *The Metaphorical Brain*, 2nd Edition, New York: Wiley.

Bates. J., Bryan Loyall A. & Scott Reilly W. (1989). Integrating reactivity, goals and emotion in a broad agent. *Technical Report CMU-CS-92-142*, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Bernet Y. (1998). Requirements of Diff-serv Boundary Routers, IETF Internet Draft.

Bouyoucef E. (2007). Contribution à l'étude et la mise en œuvre d'indicateurs quantitatifs et qualitatifs d'estimation de la complexité pour la régulation du processus d'auto organisation d'une structure neuronale modulaire de traitement d'information, *PhD Thesis*, LISSI, University Paris XII.

Bouyoucef E., Chebira A., Rybnik M., Madani K. (2005). Multiple Neural Network Model Generator with Complexity Estimation and self Organization Abilities", *International Scientific Journal of Computing*, vol.4, issue 3, pp.20-29.

Boyan J. A. and Littman M. L., Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach, *Advances in Neural Information Processing Systems 6*, Cowan, Tesauro and Alspector (eds).

Bridle, J.S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, *Neurocomputing: Algorithms, architectures and applications*, F.Foulgelman-Soulie and J Hérault, eds., New York: Springer-Verlag.

Bruske J., Sommer G. (1995). Dynamic Cell Structure, *Advances in Neural Information Processing Systems 7*, The MIT Press, Ed by G. Tesauro, pp.497-504.

Budnyk I., Chebira A., Madani K. (2008). Estimating Complexity of Classification Tasks Using Neurocomputers Technology, *International Scientific Journal of Computing*, under press.

Chebira A., Bouyoucef E., Rybnik M., Madani K. (2006). ATNS: An Adaptive Tree Neural Structure, *International Journal of Information Technology and Intelligent Computing*, IEEE Computational Intelligence Society, Vol. 1, N°3, pp.463-476.

Chi H., Ersoy O.K. (2002). Support Vector Machine Decision Trees with Rare Event Detection, *International Journal for Smart Engineering System Design*, Vol. 4, 225-242.

Cover T. M., Hart P. E. (1967). Nearest neighbour pattern classification. *IEEE Transactions on information theory*, Vol IT-13, pp 21-27.

Crawley E., Nair R., Rajagopalan B., Sandick H. (1998). A Framework for QoS-based Routing in the Internet, *RFC2386*, IETF, August.

Decker, K., Sycara, K., Williamson, M. (1997). Middle-Agents for the Internet ", *Proceedings of the 15th International Joint Conference on Artificial Intelligence,* Nagoya, Japan.

Ernst S. (1998). "Hinging hyper-plane trees for approximation and identification, *37th IEEE Conf. on Decision and Control*, Tampa, Florida, USA.

Fahlman S. E., Lebiere C. (1990). The Cascaded-Correlation Learning Architecture, *Advances in Neural Information Processing Systems 2*, Morgan Kauffman, San Mateo, pp.524-534.

Feng W., Kandlur D., Saha D., Shin K. (1997). Understanding TCP Dynamics in an Integrated Services Internet", *Proceedings of NOSSDAV*.

Ferber J. (1998). Multi-Agent Systems: Towards a Collective Intelligence, Reading, MA: Addison-Wesley.

Fisher A. (2000). The mathematical theory of probabilities, John Wiley ed.

Fukunaga K. (1972). *Introduction to statistical pattern recognition, School of Electrical Engineering*, Purdue University, Lafayette, Indiana, Academic Press, New York and London.

Fukunaga K. (1990). *Introduction to statistical pattern recognition*, Academic Press, New York, 2nd edition.

Gallager R. G.(1997). A minimum delay routing algorithm using distributed computations, *IEEE Transactions on Communications*, Vol. COM-25.

Gelenbe E., Lent R., Xu Z. (2002). Networking with Cognitive Packets, *Proc. ICANN 2002*, Madrid, Spain, August 27-30.

Goetz P., Kumar S., Miikkulainen R. (1996). On-Line Adaptation of a Signal Predistorter through Dual Reinforcement Learning, Proc. of the 13th Annual Conference Machine Learning, Bari, Italy.

Goonatilake S., Khebbal S. (1996). Intelligent Hybrid Systems: Issues, Classification and Future Directions, *Intelligent Hybrid Systems*, John Wiley & Sons Ed., pp.1-20.

Hannibal A. (1993). VLSI Building Block for Neural Networks with on chip Back Learning, *Neurocomputing*, n°5, pp.25-37.

Hartigan J. (1975). *Clustering Algorithms*. John Wiley and Sons Ed., New York.

Haykin S (1988). *Neural Networks– A Comprehensive Foundation*, Mcmillan College Publishing.

Haykin S. (1999). Neural *Networks – a Comprehensive foundation*, Prentice Hall Int.

Ho T.K., Basu M (2000). Measuring the complexity of classification problems, *Proceedings of the 15th Intenational Conference on Pattern Recognition*, Barcelona, Spain, pp. 43-47, September 3-8.

Ho T.K., Baird H.S. (1998). Pattern classification with compact distribution maps, *Computer Vision and Image Understanding*, vol. 70, no.1, pp.101-110.

Ho T.K., Basu M. (2002). Complexity measures of supervised classification problems, *IEEE Transactions on pattern Analysis and Machine Intelligence*, vol. 24, issue 3, March, pp.289-300.

Hoare C.A.R. (1962). Quicksort, *Computer Journal*, 5(1), pp.10-15.

Jacobson V. (1988). Congestion Avoidance of Network Traffic, Computer Communication" *Review*, vol. 18, no. 4, pp.314-329.

Jordan M.I., Jacobs R.A. (1993). Hierarchical mixtures of experts and the EM algorithm, *Technical Report AIM-1440*.

Jordan M.I., Jacobs R.A (2002). Learning in Modular and hierarchical systems, The Handbook of Brain Theory and Neural Networks, 2nd edition. Cambridge, MA: MIT Press, 2002.

Kohonen T. (1984). Self-Organization and Associative Memory, Springer-Verlag.

Kohn A., Nakano L. G., and Mani V. (1996). A class discriminability measure based on feature space partitioning, *Pattern Recognition*, 29(5), pp.873-887.

Krogh A., Vedelsby J. (1995). Neural Network Ensembles, Cross Validation, and Active Learning, *Advances in Neural Information Processing Systems 7*, The MIT Press, Ed by G. Tesauro, pp. 231-238.

Kumar S. and Miikkualainen R. (1998). Confidence-based Q-routing: an on-queue adaptive routing algorithm, *Proceedings of Symp. of Neural Networks in Engineering*.

Kumar S. and Miikkualainen R. (1997). Dual reinforcement Q-routing: an on-queue adaptive routing algorithm, *Proceedings of Symp. of Neural Networks in Engineering*.

Lang K. J. and Witbrock M. J. (1998). Learning to tell two spirals apart, *Proc. of the Connectionist Models Summer School*, Morgan Kauffman Ed., pp. 52-59.

Lavrac N., Flach P., Kavsek B., Todorovski L. (2002). Rule induction for subgroup discovery with CN2-SD, *Proc. Of IEEE ICDM*, pp. 266-273.

McLachlan, G.J., Basford, K.E. (1988). *Mixture Models: Interference and Applications to Clustering*", New York: Marcel Dekker.

Madani K., Chebira A., Rybnik M. (2003). Data Driven Multiple Neural Network Models Generator Based on a Tree-like Scheduler, *Lecture Notes in Computer Science n°2686*, SI on Computational Methods in Neural Modelling, (Jose Mira, Jose R. Alvarez Ed.) - Springer Verlag Berlin Heidelberg, ISBN 3-540-40210-1, pp.382-389.

Madani K., Thiaw L., Malti R., Sow G. (2005). Multi-Modeling: a Different Way to Design Intelligent Predictors, *LNCS 3512*, Ed.: J. Cabestany, A. Prieto, and F. Sandoval, Springer Verlag Berlin Heidelberg, pp.976 – 984.

Madani K. (2007). Toward Higher Level Intelligent Systems, *(Key-Note Paper), proceedings of IEEE- 6th International conference on Computer Information Systems and Industrial Management Applications (IEEE-CISIM'07)*, IEEE Computer Society, Elk, Poland, June, 28-30, pp.31-36.

Madani K. (2008). Artificial Neural Networks Based Image Processing & Pattern Recognition: From Concepts to Real-World Applications, *(Plenary Tutorial Talk, Key-Note Paper), proceedings of IEEE- 1st International workshop on Image Processing Theory, Tools and Applications (IEEE-IPTA'08)*, IEEE Computer Society, Sousse, Tunisia, November 23-26, pp. 19-27.

Maddox J. (1990). Complicated measure of complexity, *Nature*, vol. 344, *pp. 705.*

Maes, P. (1994). Social interface agents: Acquiring competence by learning from users and other agents, *Spring Symposium on Software Agents (Technical Report SS-94-03),* O. Etzioni (ed.), AAAI Press. pp.71-78.

Malkin G. (1998). RIP version2, Carrying Additional Information, *IETF RFC 1388 RFC 1993.*

Moy J. (1998). OSPF Version 2, *IETF RFC2328.*

Markovitch S., Rosenstein D. (2002). Feature Generation Using General Constructor Functions, *Machine Learning*, Springer Ed., Volume 45, N°. 1, pp.59-98.

Matusita K. (1967). On the notion of anity of several distributions and some of its applications. *Annals Inst. Statistical Mathematics*, Vol., 19, pp.181-192.

Mayoubi M., Schafer M., Sinsel S. (1995). Dynamic Neural Units for Non-linear Dynamic Systems Identification, *LNCS Vol. 930*, Springer Verlag, pp.1045-1051.

Mehmet I. S., Bingul Y., Okan K. E. (2003). Classification of Satellite Images by Using Self-organizing Map and Linear Support Vector Machine Decision Tree, *2nd Annual Asian Conference and Exhibition in the field of GIS*.

Mellouk A. (2008a). *End to End Quality of Service Engineering in Next Generation Heteregenous Networks,* ISTE/Wiley Ed.

Mellouk A., Hoceini S., Cheurfa M. (2008b). Reinforcing Probabilistic Selective Quality of service Routes in Dynamic Heterogeneous Networks, *Journal of Computer Communication*, Elsevier Ed., Vol 31, n°11, pp. 2706-2715.

Mellouk A., Lorenz P., Boukerche A., Lee M. H. (2007). Quality of Service Based Routing Algorithms for heterogeneous networks, *IEEE Communication Magazine*, Vol. 45, n°2, pp.65-66.

Mellouk A., Hoceini S., Amirat Y. (2006). Adaptive Quality of Service Based Routing Approaches: Development of a Neuro-Dynamic State-Dependent Reinforcement Learning Algorithm, *International Journal of Communication Systems,* Ed. Wiley InterSciences, Vol 20, n°10, pp.1113-1130.

Murray-Smith R. and Johansen T.A. (1997). *Multiple Model Approaches to Modeling and Control*, ed. Murray-Smith R. and T.A. Johansen, Taylor & Francis Publishers.

Naftaly, U., Intrator, N., Horn, D. (1997). Optimal ensemble averaging of neural networks, Network, vol.8, pp.283-296.

Ozdaglar A.E., Bertsekas D. P. (2003). Optimal Solution of Integer Multicommodity Flow Problem with Application in Optical Networks, *Proc. Of Symposium on Global Optimisation.*

Partridge C. (1992). A proposed flow specification, *IETF RFC1363.*

Pierson W.E. (1998). Using boundary methods for estimating class separability, *PhD thesis, Department of Electrical Engineering, Oho State University*.

Rahman A. F. R., Fairhurst M. (1998). Measuring classification complexity of image databases : a novel approach, *Proceedings of International Conference on Image Analysis and Processing*, pp.893-897.

Rosen E., Viswanathan A., Callon R. (1999). Multiprotocol Label Switching Architecture, IETF *Internet Draft draft-ietf-mpls-arch-06.txt*.

Saeed K., Tabedzki M., Adamski M. (2003). A View-Based Approach for Object Recognition, *Conradi Research Review Finland*, Vol. 2, Issue 1, pp.85-95.

Schapire R. E. (1999). A Brief Introduction to Boosting, *Proc. Of IJCAI*, pp.1401-1406.

Shenker S., Partridge C., Guerin R. (1997). Specification of guaranteed quality of service, *IETF RFC2212*.

Singh S. (2003). Multiresolution Estimates of classification complexity, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 25 , Issue 12, pp 1534 – 1539.

Singh S., A.P. Galton (2002). Pattern Recognition using Information Slicing Model (PRISM), *Proc. 15th International Conference on Pattern Recognition (ICPR2002),* Quebec.

Stallings W. (2001). MPLS , *Internet Protocol Journal*, Vol. 4, n° 3, pp.34-46.

Strassner J. (2003), *Policy-Based Network Management: Solutions for the Next Generation*, Morgan-Kaufmann Ed.

Subramanian D., Druschel P., and Chen J. (1997). Ants and reinforcement learning: A case study in routing in dynamic networks, *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence*, vol. 2, pp.832-839.

Sung K. K., Niyogi P. (1995). Active Learning for Function Approximation, *Advances in Neural Information Processing Systems7*, pp.593-600.

Sutton R. S. and Barto A. G. (1994). *Reinforcement Learning*, MIT Press.

Takeshita, T., Kimura, F., Miyake, Y. (1987). On the Estimation Error of Mahalanobis Distanc, *Trans. IEICE* Journal, 70-D, pp.567-573.

Titterington D. M., Smith A.F., Makov V.E. (1985). *Statistical Analysis of Finite Mixture Distributions*, Wiley New York.

Tresp V. (2001). *Handbook for Neural Network Signal Processing*, CRC Press.

Turner J. (1986). New directions in communications (or which way to the information age), *IEEE Communications Magazine*, vol. 24(10), pp.8-15.

Vapnik V.N. (1998). *Statistical Learning Theory*, New York Wiley Ed.

Wasserman P. D. (1993)., Advanced Methods in Neural Computing, *New York: Van Nostrand Reinhold,* pp.35-55.

Wang Z. and Crowcroft J. (1996). QoS Routing for Supporting Resource Reservation, IEEE Journal on Selected Areas in Communications, 17 (8), pp. 1488-1504.

Watkins C. J., Dayan P. (1989). Q-Learning, *Machine Learning*, Vol.8, pp.279–292.

Watkins C. J. (1989). Learning from Delayed Rewards, *Ph.D. thesis, University of Cambridge, England.*

Welzl M. (2003). *Scalable Performance Signalling and Congestion Avoidance*, Kluwer Academic Publishers.

Wooldridge M., Jennings N. (1995). Intelligent agents: theory and practice, *The Knowledge Engineering Review*, Vol.10:2, pp.115-152.

Zhang L., Deering S., Estrin D. et Zappala D. (1993). RSVP : A New Resource ReSerVation Protocol, IEEE *Network*, vol. 7, No 5, pp.8–18.

2

From Automation To Autonomy

Kentarou Kurashige[1], Yukiko Onoue[1] and Toshio Fukuda[2]
[1]Muroran Institute of Technology,
[2]Nagoya University
Japan

1. Introduction

Recently, there are many researches on intelligence in the field of engineering from various viewpoints. Representative aim is to satisfy two desires. One desire is to want more convenient machine (Kawamoto et al., 2003; Kobayashi et al., 1999; Hasegawa et al., 2004). Researchers have tried to improve existing machines or invent new machines. And now, researchers consider realizing new one by incorporating with a mechanism of life intelligence. Another desire is to want to know what intelligence is. Here, a purpose is to elucidate a mechanism of intelligence and to create it (Asada et al., 2001; Brooks & Stein, 1994; Goodwin, 1994). Researchers have expected that utility will be made known as a result of various studies.
As the milestone for intelligent machine, realizing autonomy on machine as a progress from automation is expected. The research of automation can be regarded as study how to make proper outputs by rules which human prepared. It is smarter than operating machine manually, but still not intelligent. Autonomy can be regarded as a mechanism which can make rules corresponding with surrounding environment and make proper outputs by making rules.
As one method to realize autonomy on machine, there are researches into machine learning. Especially, researches using soft computing method are so active. Essence of learning is making knowledge through trial and error and making outputs using this knowledge (Jordan, 1992). Expression of knowledge is different between each method, for example neural network (Nolfi & Parisi, 1997) has knowledge with weight matrix, but knowledge can be regarded as a rule which is mapping from input to output. Here, we have been free from necessity of a load that we must make rules to get proper outputs for all situations machine will face.
But new problem has occurred and we have gotten new load when we use learning method. We must make evaluation to learn a task or environment. In the framework of machine learning, human imagines a task which he/she gives to machine at first. Next, human must design evaluation which is a way how to teach a machine human desire. Evaluation functions expressed by numerical formula are used mostly as evaluation. The point of this is that these functions are closely related with context. So it is possible that evaluations of one output on different tasks are different values. Evaluation is strongly affected by a task, environment or a viewpoint of researchers. For this reason, a machine can work only for taught task and it is difficult to apply acquired knowledge or rules for other tasks. Human must design evaluation for all tasks individually. This load is heavy; especially in a case of

robot which has the ability to achieve various tasks and cause changeful environment by its moving ability, human must persevere in design of evaluation functions.
To overcome this problem, we focus on learning based on universal evaluation. We define universal evaluation as evaluation which is independent of a task or task information and environment a machine will be used. And we try to realize a mechanism which can learn with universal evaluation. In this chapter, we show two challenges using robot as application. One challenge is study of learning with sense of pain as universal evaluation (Kurashige & Onoue, 2007). Another challenge is study about creation of evaluation functions for concrete task and environment with energy as universal evaluation (Kurashige et al., 2002). On both challenges, we show robot can learn and create proper movement for a task or environment robot will face.

2. Learning with sense of pain on robot

In this section, we show a case of learning by using sense of pain on robot as universal evaluation (Kurashige & Onoue, 2007). We think universal evaluation must be independent of information related with each task and environment robot will face. Here, we consulted evolutionary process. Instinct which life has innately is important to keep living, and is independent of concrete environment it will face to a certain extent. Sense of pain, which is a kind of instinct, is especially important to detect abnormal state. Life can learn avoiding fatal injury with this instinct. We define sense of pain on robot and make robot learn to protect itself. And it is so hard to learn various concrete tasks only with universal evaluation. So we combine learning based on universal evaluation with usual learning method. We construct a learning system with both learning and expect that operator will be able to design evaluation function for each task easier by focusing only on a task.
We explain proposed system at first, and next we show an experiment with small-sized humanoid robot.

2.1 Outline of proposed system using sense of pain

Proposed system consist of three component; usual learning method, learning by sense of pain, action adjuster. Usual learning method is for learning a task human wants to give a robot. Here operator designs evaluation function for a task. Learning by sense of pain is for learning avoiding fatal injury. This learning is not related with each task and can be used to various tasks. Each component creates or selects action independently, so these actions conflict sometimes. Proposed system must need action adjuster to solve this problem.
We show outline of proposed system in fig. 1.

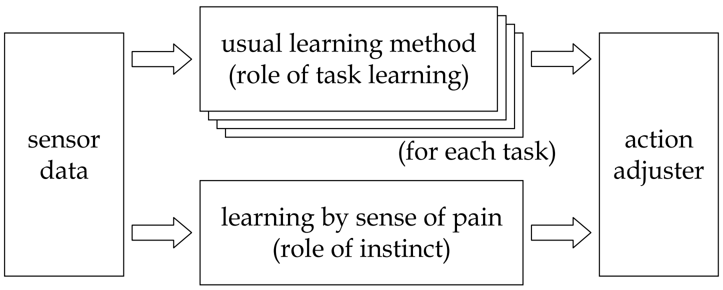

sensor data
usual learning method (role of task learning)
(for each task)
learning by sense of pain (role of instinct)
action adjuster

Fig. 1. Component of proposed system

2.2 Experimental robot

We use small-sized humanoid robot as application. We show the robot in fig. 2. This robot is about 50cm tall and has 23 degrees of freedom and various sensors. Especially, each servomotor has sensors about a position, a load and its temperature. This robot has processor unit on which UNIX OS runs internally. I show the detail in table 1.

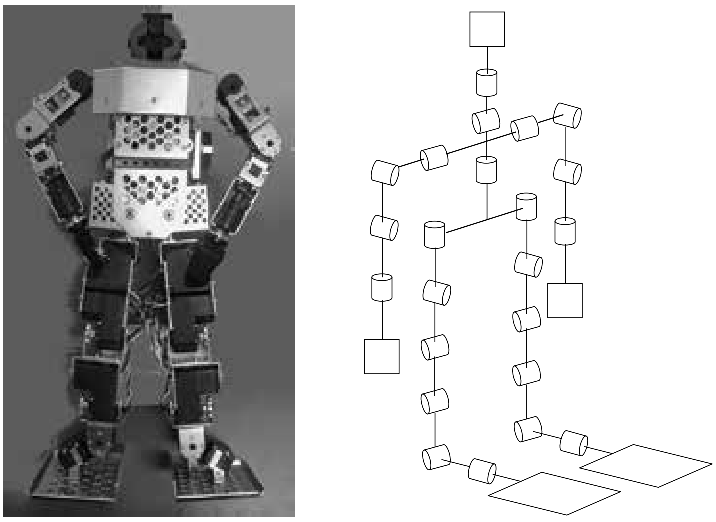


Fig. 2. The photo of the robot and the structure of the robot

Tall / Weight	50cm / 3.7kg
Degree of freedom	23 axes
sensing	single-degree-of-freedom gyro
	three-degrees-of-freedom gravity
	CMOS color camera
	2 x monaural microphone
sensing (each servomotor)	angle
	torque
	temperature
other interface	2 x LED (3 color)
	speaker
	wireless LAN (IEEE 802.11b)

Table 1. The specification of experimental robot

2.3 Definition of pain on the robot

We define pain on the robot based on its sensor values. We consider that a robot has N kinds of sensors. For each sensor, we define normal value and abnormal value. And if there is over one sensor which has abnormal value, we define a robot feels pain. In this section, we use a torque sensor which can detect a load on a servomotor and define pain on experimental robot. Using L_i which is the value of i-th torque sensor, we define the state which the sensor has abnormal value as $L_i > L_i'$. By this, we define $pain_i$ as follows; value of 1 means robot feels pain on place of i-th sensor, value of 0 means robot doesn't feel pain on it.

$$\begin{aligned} L_i > L_i' &\rightarrow pain_i = 1 \\ L_i \le L_i' &\rightarrow pain_i = 0 \end{aligned} \tag{1}$$

To determine L_i', we examine pre-experiment which made the robot move randomly, collect data of values of L_i and calculate average μ_i and deviation σ_i. By these values, we define L_i' as follows

$$L_i' = \mu_i + 3\sigma_i \tag{2}$$

Using $pain_i$, we define $pain$ as follow.

$$pain = \bigcup_i pain_i \tag{3}$$

2.4 Learning a given task and avoiding fatal injury using RL as learning method

We give the robot a task which is to select action human want the robot to do. Here we decide desired action as follows.

learning task :

a. If the robot detects load on arm in back and forth, desired action is to move its arm back and forth.
b. If the robot detects load on arm in right and left, desired action is to move its arm right and left.

At the same time, we expect that the robot learn by sense of pain and avoiding fatal injury.

learning by sense of pain :

c. If the robot detects abnormal load on arm, desired action is avoidance action.

We use reinforcement learning (Sutton & Barto, 1998) to realize these learning. We adopt Q learning as a learning method (eq. 4). This way, we applied same equation to both learning.

$$Q_\#(s_{t\#}, a_{t\#}) \leftarrow Q_\#(s_{t\#}, a_{t\#}) + \alpha_\#\left[r_{t+1\#} - Q_\#(s_{t\#}, a_{t\#})\right] \tag{4}$$

Here, $S_{t\#}$ is a current state, $a_{t\#}$ is a selected action, $r_{t\#}$ is a reward obtained by the action. Subscript symbol "*t*" is discrete time step, and "#" is whether "*pain*" or "*task*". For example, $a_{t\,pain}$ is action *a* at time *t* considering at learning based on sense of pain. And we adopt Softmax Action Selection defined by eq. 5 to select action *a*.

$$\pi_\#(s_{t\#}, a_{t\#}) = \frac{e^{Q_\#(s_{t\#}, a_{t\#})/\tau_\#}}{\sum_{a_\#} e^{Q_\#(s_{t\#}, a_{t\#})/\tau_\#}} \tag{5}$$

Here, $\tau_\#$ is a positive constant called temperature. Other is same meaning as upper case.

Next, we define states and actions to use reinforcement learning. For learning task, we define these as table 2. And for learning by sense of pain we define these as table 3.

We use plural learning which is for task and is based on sense of pain, so plural actions will be selected. To make the robot move actually, one action must be selected. We consider action adjuster to select an action the robot will act. On this mechanism, an action which has maximum value in $\pi_\#$ at "#" is selected. We show the outline of action adjuster in fig. 3.

Using proposed system, we realize to learn given task and to learn avoiding fatal injury at the same time. At the experiment, the learning for given task is tried 100 times in each state.

$s_{0\,task}$	load detection in back and forth
$s_{1\,task}$	load detection in right and left

(a) state

$a_{0\,task}$	move arm back and forth
$a_{1\,task}$	move arm right and left

(b) action

Table 2. States and actions for learning task

$s_{0\,pain}$	*pain* = 0 (robot doesn't feels pain)
$s_{1\,pain}$	*pain* = 1 (robot feels pain)

(c) state

$a_{0\,pain}$	continue a present action (no action for avoidance)
$a_{1\,pain}$	return the servo to an initial position (avoidance action)

(d) action

Table 3. States and actions for learning by sense of pain

Learning for given task	positive reward	5
	negative reward	-3
	α_{task}	0.1
	τ_{task}	3
Learning by sense of pain	reward if return the servo to an initial position	-1
	reward if servo become to be abnormal state	-100
	α_{pain}	0.5
	τ_{pain}	0.5

Table 4. The parameter for the experiment

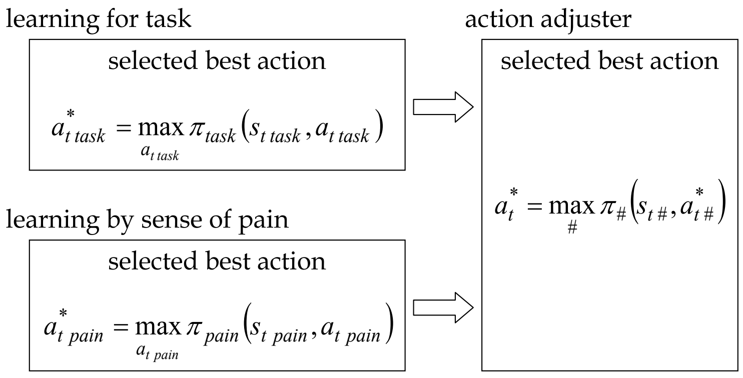

learning for task
selected best action
$a^*_{t\,task} = \max_{a_{t\,task}} \pi_{task}(s_{t\,task}, a_{t\,task})$
learning by sense of pain
selected best action
$a^*_{t\,pain} = \max_{a_{t\,pain}} \pi_{pain}(s_{t\,pain}, a_{t\,pain})$
action adjuster
selected best action
$a^*_t = \max_{\#} \pi_{\#}(s_{t\,\#}, a^*_{t\,\#})$

Fig. 3. Outline of action adjuster

And the learning by sense of pain is done once every 500msec. Other parameter is shown in table 4.

2.5 Result

We show results in fig. 4 and fig. 5 and table 5. The transition of action selection probability in learning for given task is shown in fig. 4. It shows that the selection probability of the best action was rising with progress of the trial time. The transition of action selection probability in learning by sense of pain is shown in fig. 5. It shows that the learning was done and the robot got the ability of avoiding fatal injury.

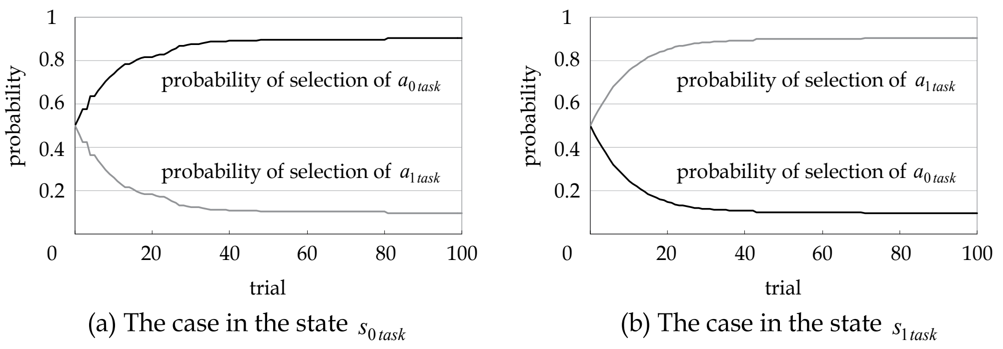


Fig. 4. The transition of probability of action selection in learning for given task

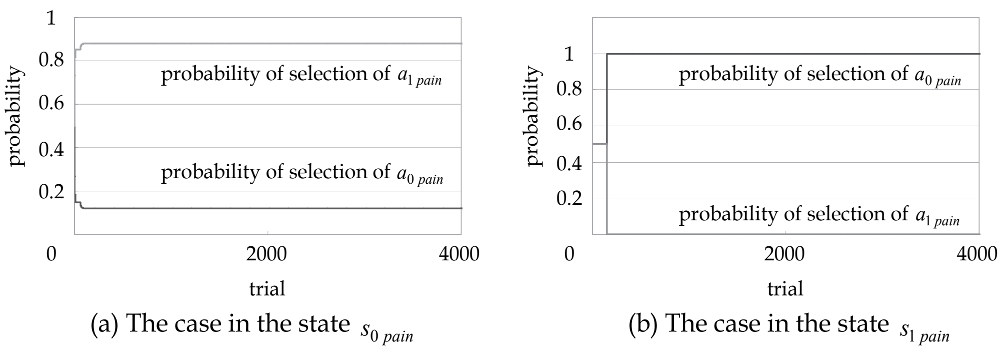


Fig. 5. The transition of probability of action selection in learning by sense of pain

	$a_{0\,pain}$ (no avoidance action)		$a_{1\,pain}$
	$a_{0\,task}$	$a_{1\,task}$	
$s_{0\,task}$ and $s_{0\,pain}$	99.67%	3.33%	0%
$s_{1\,task}$ and $s_{0\,pain}$	6.67%	93.33%	0%
$s_{0\,task}$ and $s_{1\,pain}$	0%	0%	100%
$s_{1\,task}$ and $s_{1\,pain}$	0%	0%	100%

Table 5. The result of action selection after 120 times learning

After learning, we experimented to confirm the result of the learning. We give the robot given task at 120times including the case caused abnormal state. The result of this confirmation is shown in table 5.

3. Creation of evaluation functions with energy

In this section, we show a study about creation of evaluation functions by using energy of robot as universal evaluation (Kurashige et al., 2002). How to evaluate robot's action changes in different contexts, in different tasks or environment. For usual learning, human must design evaluation functions for concrete task or environment robot will face. We have tried to create proper evaluations along concrete task and environment by universal evaluation. We proposed a method based on motivation to drive action on life. Here, we show motivation model we proposed and experiment on computer simulation.

3.1 Proposed concept "motivation model" based on life

Life has desire to feel satisfaction, especially when they feel insufficiency. They act for the aim of being satisfied with their status. The force that causes life to take action by desire is called "motive" in the field of psychology (Atkinson et al., 1999). Motive is classified roughly into two types; one is called basic motive and another is called derived motive. Basic motive is considered as motive which life has innately and which is equally among life or a species. Derived motive is considered as motive which is acquired through individual experience and is different on each other. And derived motive is considered as one gained based on basic motive. But this acquisition process isn't fixed on yet.

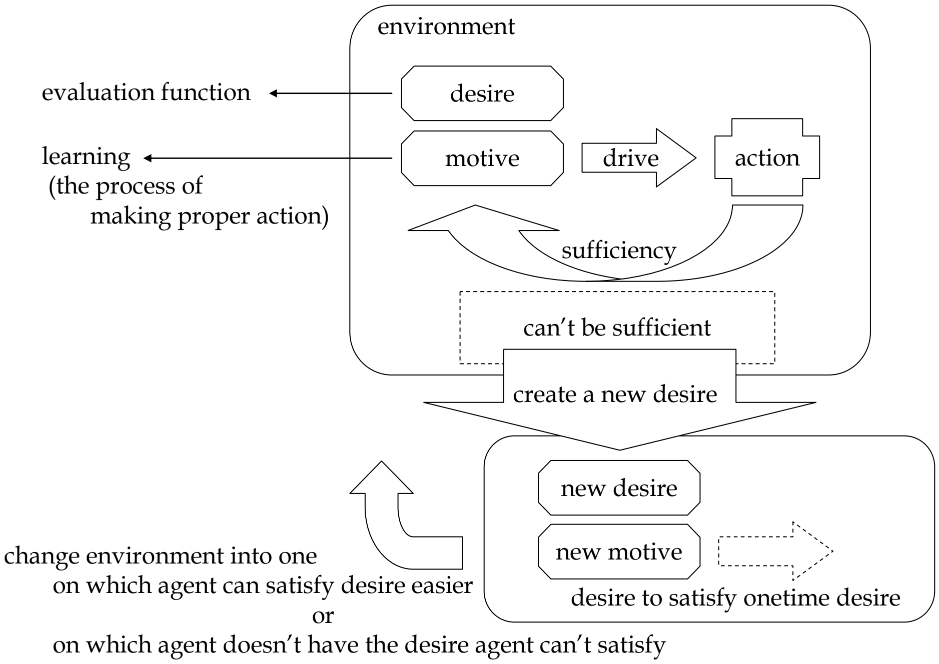


Fig. 6. Proposed concept named motivation model

Here, we thought of basic idea based on this knowledge as follows. Desire on agent, which is robot or etc., is a direction or index of satisfaction. And motive on agent is the process which agent creates or selects action to satisfy its desire. We consider desire as evaluation function and motive as learning process. If agent can learn and satisfy its desire, there is no problem. If it is hard or impossible to make proper action for satisfaction of its desire, there is problem that agent can't satisfy its desire. To solve this problem, we consider that agent creates new desire which is to satisfy one time desire. By action caused by new motive to try to satisfy corresponding desire, agent tries to change an environment into the others on which agent can satisfy its desire easier or on which agent doesn't have the desire it can't satisfy. Especially by the latter case, agent tries to avoid an environment on which agent can't satisfy its desire, and tries to learn proper action on other environment to satisfy its desire. This is outline of idea named "motivation model". We show proposed concept in fig. 6. Next, we construct concrete algorithm by motivation model.

3.2 The algorithm to generate evaluation functions based on motivation model

We propose an algorithm to generate evaluation functions based on motivation model. Here, we construct the algorithm by modifying reinforcement learning (Sutton & Barto, 1998). The outline of proposed algorithm is shown in fig. 7. Evaluation μ_i is i-th evaluation and produces reward which is decided according to an agent's state. Knowledge space is the space composed by μ_i, s, a and is made by learning. If agent can get high reward and be sufficient by learning, there is no problem. If it is hard or impossible to get high reward, we think there is problem and try to make agent create new evaluation to solve the problem. We explain when agent creates new evaluation, and next explain the algorithm how to create it.

We define the timing to create new evaluation by a shape of knowledge space. At first, we define knowledge space corresponding to i-th evaluation as $M_i : \mu_i \times (s, a)$ and show outline in fig. 8. We classify this under four typical types to explain a concept of creation of new

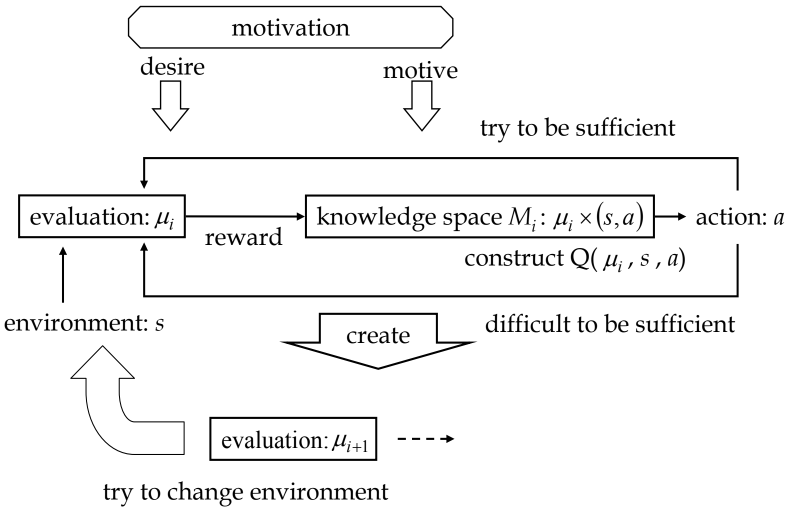

motivation
desire
motive
try to be sufficient
evaluation: μ_i
reward
knowledge space $M_i : \mu_i \times (s,a)$
action: a
construct Q(μ_i , s , a)
environment: s
create
difficult to be sufficient
evaluation: μ_{i+1}
try to change environment

Fig. 7. The outline of proposed algorithm

evaluation as shown in fig. 9. In the case of fig. 9(a), both an agent's action and a state of environment agent faces influence an evaluation score, so they have the strong relationship. In the case of fig. 9(b) and (c), the relationship between an agent's action and a state is weaker than in the case of fig. 9(a). Evaluation score depends only on a state of environment in the case of fig. 9(b) and depends only on an agent's action in the case of fig. 9(c). Lastly, there is no relationship between an agent's action and a state of environment in the case of fig. 9(d). Here, we focus on cases of fig. 9(a) and (b). In these cases, an agent can't control its evaluation score only by its action. The evaluation score depends on a state of environment. So we consider that an agent creates new evaluation in these cases, and by created action under new evaluation an agent tries to be in a state which has possibility to get high reward. To judge whether new evaluation must be created or not, we use joint probability distribution $P(\mu_i, s, a)$. By this, we can calculate marginal probability distribution $g(\mu_i, a)$ as shown in eq. 5.

$$g(\mu_i, a) = \sum_s P(\mu_i, s, a) \tag{5}$$

At this time, we can calculate existence probability p on r_{μ_i} as follows.

$$p = g(r_{\mu_i}, a_t) \tag{6}$$

Here, r_{μ_i} is reward for an action a_t under a state s_t according to evaluation μ_i. Using existence probability p, we define the probability of generation of evaluation function as $1\text{-}p$.

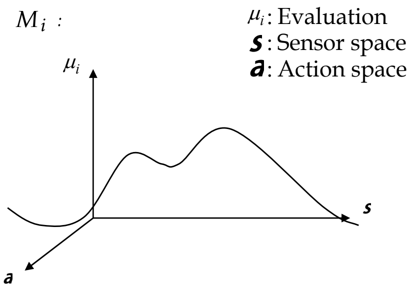
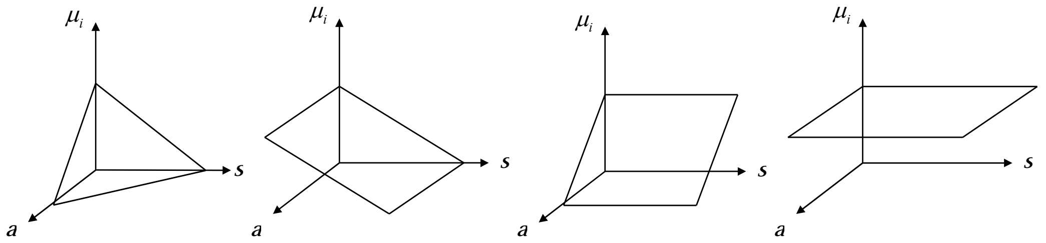


Fig. 8. The outline of knowledge space



(a) strong relationship (b) insensitive to action (c) insensitive to env. (d) weak relationship

Fig. 9. Four typical types of knowledge space

Next, we explain how to create new evaluation. We think that an agent tries to be in a state which agent can get higher reward by an action derived by new evaluation. So we define new evaluation μ_j with a state s. On knowledge space M_i , we can calculate marginal probability distribution $f(\mu_i, s)$ as shown in eq. 7.

$$f(\mu_i, s) = \sum_a P(\mu_i, s, a) \tag{7}$$

An action a_t at time t under evaluation μ_j is action to make profitable environment under evaluation μ_i. So we define μ_j using a state s_{t+1} derived by a_t as follows. And we show the concept of how to create new evaluation in fig. 10.

$$\mu_j = \mu_j(s_{t+1}) = E(\mu_i(s_{t+1})) = \sum_{r_{\mu_i}} r_{\mu_i} \cdot f(r_{\mu_i}, s_{t+1}) \tag{8}$$

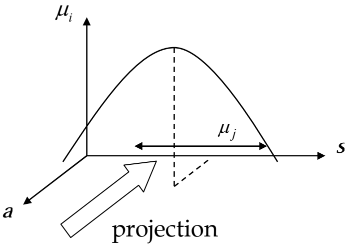


Fig. 10. Concept of how to create new evaluation

Finally, we explain how to select action using these evaluation functions. On each evaluation μ_i, an action a_i which can take $\mu_{i\,\max}$ is selected. Here, $\mu_{i\,\max}$ is maximum value of evaluation μ_i. The number of candidate actions is equal to the number of evaluation functions. We define probability of selection for each action a_i as eq. 9. An agent decides an action based on this probability of selection.

$$q(a_i) = \frac{\mu_{i\,\max}}{\sum_i \mu_{i\,\max}} \tag{9}$$

3.3 Burden-carrying task

We applied proposed algorithm to burden-carrying task. The object environment is shown in fig. 11. The task is to carry burdens from loading station to unloading station. The robot which is the agent at this task can get energy β per one burden as a reward for work. In the environment, there are several kinds of hindrances. They are walls and burdens. Walls bar robot's way. If the robot puts burden down on any place except unloading station, it will become hindrance.

For this task and environment, the robot can takes several actions: Load, Unload, Forward, Left, Right and Stop. The robot needs energy to execute each action whether the robot can do or not. So if the robot fails to execute an action, for example the robot tries to go through a wall, the robot loses same amount of energy when the robot succeeds to take that action and a state of the robot doesn't change. In this task, we set energy to take any action as α. Actions the robot can take and perceptions the robot can use are as follows.

Load	get a burden in front of the robot
Unload	put a burden down in front of the robot
Forward	take a step forward
Left	turn to the left
Right	turn to the right
Stop	stop

Table 6. Actions the robot can take

$state_{direc}$	state around the robot (direc : forward, right, left, back)
$state_{burden}$	state whether the robot has burden or not
$(x, y, direc)$	current location and direction
$\Delta energy$	change of energy

Table 7. Perceptions the robot can use

We define initial evaluation function by using the change of energy of the robot as eq. 10. This is basic motive at this task. And it plays the role of universal evaluation because of the definition which is independent of environment.

$$\mu_1 = \Delta energy = \begin{cases} -\alpha \\ \beta \end{cases} \tag{10}$$

Here, α is energy to take an action and β is a reward for work when the robot can get at unloading station.

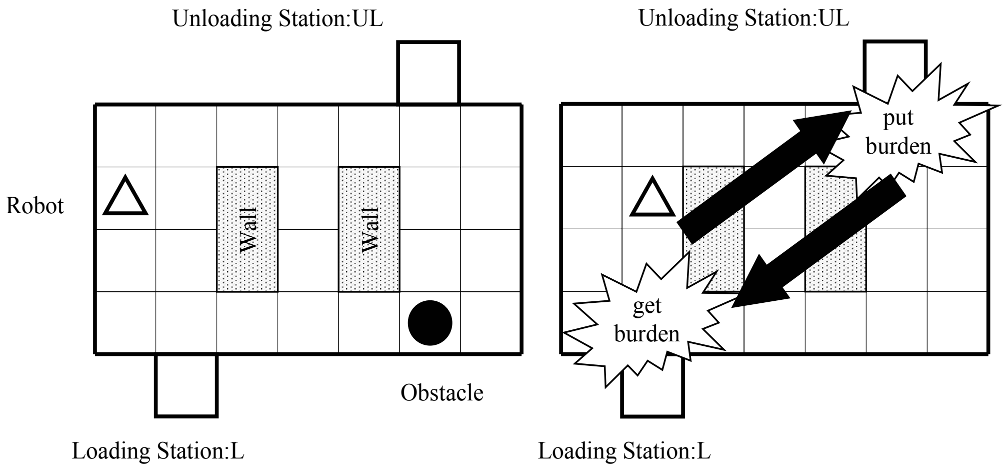


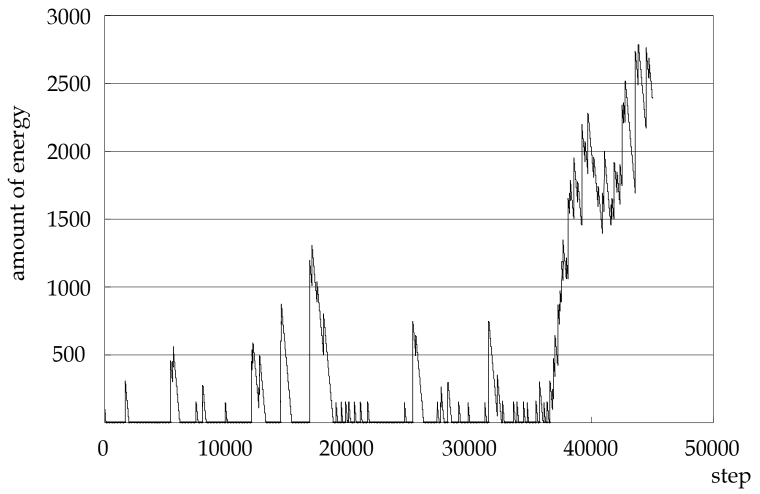
Fig. 11. Outline of load-carrying task

3.4 Results of computer simulation

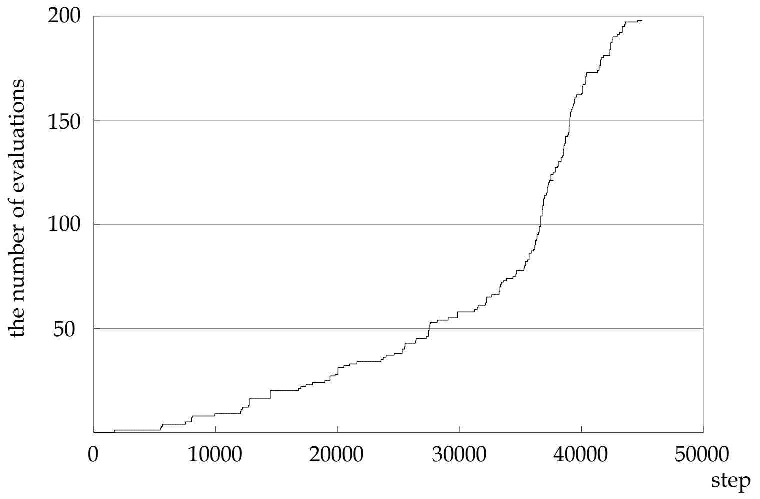
We experiment burden-carrying task on computer simulation. The robot has energy φ as initial energy. If energy of the robot drops to zero, we give the robot energy γ in the midst of learning as recharging. The number of burden which the robot can carry at once is expressed as χ. We show the parameter of simulation in table 8.

α	-1
β	150
φ	100
γ	10
χ	10

Table 8. The parameter of simulation



(a) Transition of the amount of energy robot keeps



(b) Transition of the number of evaluation

Fig. 12. Results through learning

The results of simulation under this condition are shown in fig. 12. Figure 12(a) represents transition of amount of energy on the robot. Figure 12(b) represents the number of evaluation which the robot creates with proposed algorithm.

In the first part of fig. 12(a), the amount of energy which the robot kept was low. We consider it was occurred because the robot took actions randomly in this phase which is early phase of learning. And increasing the number of evaluation, we can see the amount of energy which robot kept was rising.

And we show the existence probability of the robot on the environment from 40000 step to 50000 step in fig. 13. This shows the robot went round between loading station and unloading station.

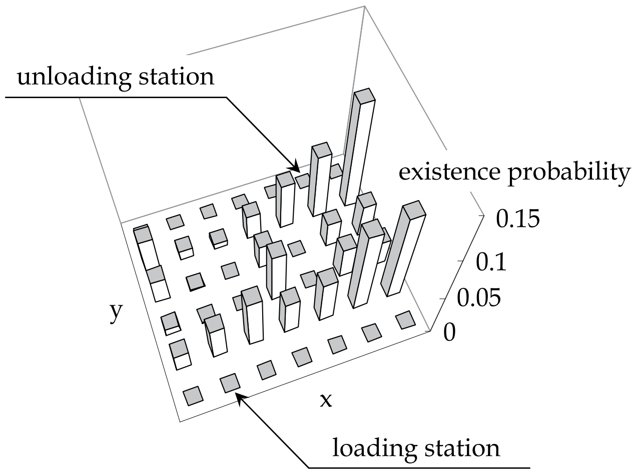


Fig. 13. Existence probability of the robot on learning between 4000 step and 5000 step

4. Conclusion

Our goal is to realize a system which keeps adapting various tasks and environment with universal mechanism which is independent of concrete tasks and environment. In this chapter, we proposed the concept of universal evaluation as a kind of universal mechanism. Here, we showed two experiments as instances. One is the study using sense of pain as universal evaluation. With this universal evaluation, the robot could avoid being injured by unexpected load. Another is the study to create evaluation functions for concrete environment by universal evaluation. We showed recursive algorithm to create evaluation functions by existing evaluation functions. And we used evaluation about energy on robot as the beginning and universal evaluation. We showed the robot could take more proper action as it created evaluation functions by proposed algorithm.

As the future works, we try to find and propose better universal mechanisms. For example, we consider that a rule how to interact environment can be used as a universal mechanism. As the first step of this, we have tried to create evaluation functions for concrete task and environment with an interaction rule which is defined by variance of sensor data (Kurashige, 2007). By importing a concept of universal mechanism into learning method, we try to divide between how to design a robot and how to use a robot, and we try to realize a system which can get necessary knowledge whenever it is necessary only with an operation of its information. We think that is next step for autonomy.

5. References

Asada, M.; MacDorman, K. F.; Ishiguro, H. & Kuniyoshi, Y. (2001). Cognitive Developmental Robotics As a New Paradigm for the Design of Humanoid Robots, *Robotics and Autonomous System*, 37, (2001), pp. 185-193

Atkinson, R. L.; Smith, E. E.; Bem, D. J. Nolen-Hoeksema, S. (1999). *Hilgard's introduction to psychology*, Thirteenth edition, Wadsworth Publishing

Brooks, R. A. & Stein, L. A. (1994). Building Brains for Bodies, *Autonomous Robots*, 1, 1, (November, 1994), pp.7-25

Goodwin, R. (1994). Reasoning about when to start Acting, *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, pp.86-91, Chicago, IL, USA, May, 1994

Hasegawa, Y; Yokoe, K; Kawai, Y. & Fukuda, T. (2004). GPR-based Adaptive Sensing -GPR Manipulation According to Terrain Configurations, Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.3021-3026, Sendai, Japan, October, 2004

Jordan, I. M. (1992). Forward models: Supervised learning with a distal teacher, *Cognitive Science*, 16, (1992), pp.307-354

Kawamoto, H.; Lee, S.; Kanbe, S. & Sankai, Y. (2003). Power Assist Method for HAL-3 using EMG-based Feedback Controller, *Proceedings of International Conference on Systems, Man and Cybernetics*, pp.1648-1653, Washington DC, USA, October, 2003

Kobayashi, E.; Masamune, K.; Sakuma, I.; Dohi, T. & Hashimoto, D. (1999). A New Safe Laparoscopic Manipulator System with a Five-Bar Linkage Mechanism and an Optical Zoom, *Computer Aided Surgery*, 4, 4, pp.182-192

Kurashige, K. (2007). A simple rule how to make a reward for learning with human interaction., *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp.202-205, Jacksonville, FL, USA, June, 2007

Kurashige, K.; Aramaki, S. & Fukuda, T. (2002), Proposal of the motion planning with Motivation Model, *Proceedings of Fuzzy, Artificial Intelligence, Neural Networks and Computational Intelligence*, pp. 467-472, Saga, Japan, November, 2002

Kurashige, K. & Onoue, Y. (2007). The robot learning by using "sense of pain", *Proceedings of International Symposium on Humanized Systems 2007*, pp.1-4, Muroran, Japan, September, 2007

Nolfi, S. & Parisi, D. (1997). Learning to adapt to changing environments in evolving neural networks, *Adaptive Behavior*, 5, 1, (1997), pp.75-98

Sutton, R. S.; Barto, A. G. (1998). *Reinforcement Learning An Introduction*, The MIT Press, Cambridge

3

Taking Experience to a Whole New Level

Luis Ignacio Lopera
Universidad de los Andes
Colombia

1. Introduction

Human kind through out history has shown a keen ability to learn by observation and to create. He's the only species on earth that has drastically changed his surroundings by constructing cities, houses and parks among other things. He also has left the planet for the nearest celestial body and built a home on the stars. But if one takes the knowledge needed to build something as complicated as the space station, one soon realizes that one did not have to learn everything at once. As matter of fact the knowledge needed to build the station is the result of a very long learning process that was done one step at the time.

This type of learning process, based very strongly on previous experiences, has proved to be efficient in the way that once something works it is fairly easy to replicate or do it better. However, it is interesting to point out that regardless if it is the best method for learning it is the only method used. The school systems all around the world expect a child to learn certain skills during the first years of schooling, such as reading, writing and spatial reasoning. Then these skills are broadly used from there on to learn things like basic algebra, logic reasoning, arts, crafts, history and so on. Once in college the student is expected to choose an area of interest and study the extra skills necessary to learn the advanced subjects of the area and be able to use them in a professional environment. If the student pursues a higher degree of education his success will reside on his ability to interconnect past experiences to produce some new bits of knowledge.

Interesting enough, the power of knowledge is derived not only from personal experience but from a collective experience as well. This can be seen in very isolated communities as well as in the global community of today. In aborigine tribes, the collective experience is passed from generation to generation usually by means of oral tradition. For example, the best way to hunt, the best grassing places for cattle and so on. Such knowledge is updated by the most recent personal experiences. In today's more globalized community experiences are shared through many different channels, such as books or the internet.

The discussion comes to the point where it becomes important to define experience. In a general context; the Merriam Webster's dictionary defines experience as:

"1 a: direct observation of or participation in events as a basis of knowledge b: the fact or state of having been affected by or gained knowledge through direct observation or participation

2 a: practical knowledge, skill, or practice derived from direct observation of or participation in events or in a particular activity b: the length of such participation <has 10 years' experience in the job>

3 a: the conscious events that make up an individual life b: the events that make up the conscious past of a community or nation or humankind generally
4: something personally encountered, undergone, or lived through
5: the act or process of directly perceiving events or reality".
These definitions illustrate clearly how knowledge can derive from direct or indirect involvement in an activity. It also defines a way of learning. More precisely in the context of this book, "*Learning* is done by a machine when it records its experience into internal system changes that causes its behavior to be changed." (Looney, 1997). Most algorithms in machine learning use this definition to better adjust the detected classes and generate new ones if necessary.
But unfortunately, these inner changes do not take the machine closer to a human like learning method. It only perfects the machine output to a constrained set of variables. But if the set of variables, all of the sudden, become unconstrained or the constraints change drastically, the previews experiences become obsolete and the training process has to star all over again (Hagras et al., 1997). As a result, for machine learning applications, you want the problem to be as constrained as possible and the machine as invariable as possible. These limitations become the "*Achilles' heel*" of systems that have to undertake unexplored and unstructured environments.
Understanding human experience has been the material of study by many philosophers, and scientists. Is not the intention of this chapter to enter in the discussion on any way, however, it is relevant to point out that the basic definition given before falls short to describe experience that transcends the observed event's context; in other words, experience that is used in something else than the set of events where it was generated. This is best illustrated by an example: An electrical technician learned throughout his career how to repair CRT TV's, now he is faced with the challenge of repairing LCD screens. It is evident that some additional learning has to be done, but, a lot of the skills used to fix the CRT will be useful to fix the LCD. And furthermore, if another CRT TV comes to his shop, he would still be able to fix it.
From a systemic point of view, the agent's physical capabilities, such as sensors actuators, computational power, etc, can be considered *services* to the way of doing things. And these services become the framework to design and develop the architecture that will take experience to the next level.
The chapter starts the discussion by analyzing the way people carry out tasks, then introduces a concept of knowledge and its intricate relation to experience then a series of architectures are presented that illustrate the way next level experience can be implemented. These architectures are thought out to implement the ability, very often seen in human reasoning, of extrapolating experience; as in the example of the TV technician. The goal of the presented architectures is to establish the ways in which to use the agent's services to obtain the most of the agent's capabilities and increase the chance of success when faced with various problems and circumstances. Then it shows the application of one of the architectures to a theoretical problem and ends the discussion with some final remarks about the practical implications of using the proposed architectures.

2. Simplicity, fun facts of the way we do things.

'STOP, think on what are you about to do!', many times we have heard mothers instruct their children, usually because the youngster is about to harm himself, or engage in some mischievous behavior. This phrase is going to be the motto for this chapter's section.

Must people have certainly come across the annoying problem of having to fix a house appliance or an office gadget. And the resulting outcome for most of these people is to throw it away or call tech services. The focus of this section is the small portion of users who actually try to fix the broken object. For them here's the motto: *'STOP!!, think on what are you about to do!'*. Otherwise, how are we ever going to understand what's going on in the users' heads?
The problem of fixing things is very interesting to study the way we do things, mostly because it involves several brain actions/properties, like experience, analysis, observation, decision making, and coordination of movement, among others.

2.1 Case 1: opening the black box problem.

Once upon a time there was a black box. This box had a lid which was screwed shut with flat head screws, there where four of them one on each corner. The box had a "*broken*" behavior. (At this point it is irrelevant what the problem of the box is) To fix it, the person who's going to fix it (from here on, the fixer) must open the box and see what is causing the problem.
Inspired by the motto, an interesting question arises: What is the sequence of steps that are required to get in to the box? Let's follow a line of reasoning to find the answer to this question.
First step is always observation, observe the problem in detail and get as many characteristics as possible. From here the fixer will know things like there's a lid, there are 4 flat head screws, their position and size, and so on. It seems obvious after reading the problem's description, but bear in mind that the fixer is presented with the black box and not the description of the black box.
Next step is experience, the fixer must ask himself, *"Have I opened THIS black box before?"* if the answer to this question is YES, the answer to the "what is the sequence…" question is immediately found, the steps are somewhere in the fixer's head. But this trivial answer is not what we are looking for. If we take the NO answer, the following question arises: *"Have I opened SOMETHING LIKE this before?"* In this case a YES answer would lead to compare the black box, with every experience of opening THINGS LIKE this one, and using the best match to try and open it. In essence, finding similarities with previous elements would give us a starting point that is further ahead in the solution process than starting from scratch. On the other hand, the NO answer would lead to the next step.
Analysis, here the fixer must determine the type of tool he's going to use to unscrew the screws. Probably establish if there's a sequence to follow or just any random order will do the job, determine if it is sufficient to unfasten the screws or if they have to be removed completely.
The final step is action: the fixer does something, for example, removes a screw, from here on, the road can take two paths: trial and error or a methodic process of disassembly. Either one will get the job done, it's important to appreciate that in both roads the process becomes cyclic, as the fixer will have to stop and observe after each step is taken to determine if he's going to achieve his goal and apply this sequence of steps for each particular problem encountered. Figure 1, shows a flow chart of the Meta algorithm of opening the black box.
It is interesting to notice how there are two type of experiences that become very useful in this process. The first type of experience is very much like defined in section 1, and used widely in machine learning algorithms: direct experience over the event, the second type of

experience is an extrapolated experience, in other words, it is experience achieved in other events that is used to find a quick solution to the problem, a starting point further ahead in the road of solving the problem. As an example, opening the black box would allow the fixer to understand the way the computer's cover is quickly removed.

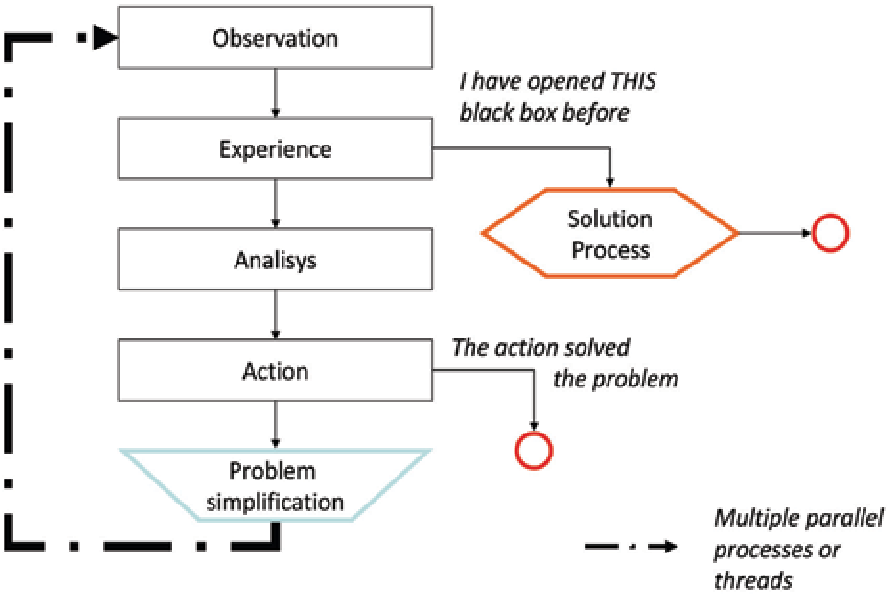

Observation
I have opened THIS black box before
Experience
Solution Process
Analisys
The action solved the problem
Action
Problem simplification
Multiple parallel processes or threads

Fig. 1. Meta algorithm for problem solving

Other interesting observation on this case is the way it can be compared to recursive programming. In recursive programming the algorithm is called several times but every time with a simpler task, in terms, the same thing happens in case 1, the same four basic steps are recalled every time with partitions of the bigger problem.
Simplicity in this case is related to understanding that only 4 steps are needed, and that they repeat themselves over and over.

2.2 Case 2: fixing the black box's broken behavior.

At this point the fixer has opened the black box, and needs to fix the problem, as in the previous case, a very similar question arises: What is the sequence of steps needed to repair the problem? To find the answer to this question this time, we are going to take a different path; there is a meta-algorithm used widely for fixing things, it can be simplified to three stages as: Diagnose, repairing (replace, reposition, reconfigure, reinstall) and test.
With this meta-algorithm it is important to subdivide the task in two types. First type, it is the kind that comes with a manual, in this type of fixing, the fixer only needs to follow a set of steps designed to pinpoint the problem and fix it. Its only reasonable to mention that on this type of process, the fixer needs direct experience on how to solve the little details, the ones the "manual" assumes the fixer knows how to do. So, only one type of experience is needed. This is the type of activity people train for.
The second type of subdivision is the one with no "manual" or only limited information available. There are no steps or a determined sequence to follow, in this case (which is very interesting for this chapter), the fixer must use experience of different types to diagnose the

problem, and fix it. Interesting enough, if the meta-algorithm applied in case 1 is used for the diagnostics, repairing and testing stages, a solution to the problem can be found.

To illustrate, let's say the black box's broken behavior consists of a failure on an indication led that informs the status of the connection to a wireless network. Assume that the problem is a burnt resistor from the led's amplifying circuit. To understand what is going on (diagnostics stage) the fixer starts proving, looking to see if the box is actually able to connect, regardless of the *led*. The fixer must see that the box seems to work fine in this regard (*observation*), he turns to *analyze* the led's circuitry, un-solders the led (*action*) and tests it by itself. This because he knows from his *experience*, that L.E.Ds blow out rather often (It is important to mention that this is based on the fixer's experience, and only for the purpose of the example). When he finds that the led is not the problem, then he solders back the L.E.D, and starts checking for voltage level in the amplifying circuit until he finds the blown resistor. Again it is clear how the four basic steps of the meta-algorithm are used over and over again.

Then he gets the replacement resistor (repairing stage), un-solders the blown one and solders the new one. Repairing is usually a trained activity, therefore, this stage usually does not use the meta-algorithm; rather, it will use a list of steps or procedures. However, once in a while, to repair something the fixer must get creative. Assume now that he doesn't have the right value resistor, better yet, he has no resistors at all. He could run to the store and buy a new one; but again, not a very interesting solution. He could get the resistor from another broken gadget. In this case the meta-algorithm could be used to find and recover the part, and as it usually happens, the replacement is probably not a perfect fit, so he would have to use the meta-algorithm again to modify it and make it fit.

Finally testing, the fixer has to undergo a procedure to figure out if the repair was well done. Again we stumble with the duality of procedure vs. experience. The fixer could use procedures if they exist. But if not, he must rely heavily on experience to test the system until a suitable set of possibilities for failure is tried out and pass satisfactorily. In the example of the black box, it is rather simple: Activate wireless communication and see if the L.E.D blinks as it is supposed to.

With case 2 it becomes clear that there's a layer-like architecture to the process of fixing something. Upper layers determine the general procedure to follow, and lower layers take care of particular tasks. Furthermore, simplicity is associated to the use of the meta-algorithm in several occasions and contexts.

After having "stopped and thought" on what we where about to do to the black box; it is important to extrapolate at this point. If all possible problems are grouped together in to categories based on the agent's capacity to solve them, only three categories arise: problems which already have been solved, those which haven't and those which can't be solved by the agent. Those that have been solved become procedures like how to build a computer or a car, in the case of people, they could also become instinct, like running or dancing. Those that haven't been solved are the ones that present a challenge, and there's where the meta-algorithm comes in action, always observing, putting all other experiences to the test, analyzing and acting upon.

Although it is not the intention of this section either to undermine or to simplify the creative process, the act of problem solving of the human mind, which relies on creativity, can be approximated by understanding that a big part of the creative process comes from melding experiences achieved through out a series of events in a similar or even in completely different context than that of the problem at hand. A glance at the way any engineer's talent evolves shows that although early stages could be magnificent, the best work is always later in the career because is fueled in part by the new experiences achieved in the early stages.

3. Storage, the key for knowledge.

Although the debate on a definition of Knowledge is still on-going, for all purposes of this chapter knowledge would be understood as defined in the Oxford English Dictionary: (i) expertise, and skills acquired by a person through experience or education; the theoretical or practical understanding of a subject, (ii) what is known in a particular field or in total; facts and information or (iii) awareness or familiarity gained by experience of a fact or situation.

It is interesting how knowledge and experience are intricately related. From the definition can be derived that since machine learning algorithms use a process of experience to better perform the given tasks, ergo, any system that uses a machine learning algorithm has *knowledge* of the specific task. The only problem with this statement is that by definition, knowledge seems to be a trait exclusive of a "person". Never the less it is still valid, if we understand a person as the ultimate system or agent. In other words, extrapolating the concept of knowledge to lesser systems, such as mechanical or electronic system, to describe the information, expertise and familiarity obtained through experience or education.

The term information is clear to see in current day technology, people store hundreds of thousands of information represented in bytes. It is also clear to see how a few fields in memory describing the algorithm's results or properties can be considered valid information, and that such can be acquired or refined through experience or programming (the equivalent of education in "lesser" systems),therefore also considered as knowledge. However, what to make of awareness and expertise? Can they be replicated in a non human system?

Expertise can be defined as the capacity of the system to carry out a task efficiently. Therefore, it can be replicated as it has been widely demonstrated that for certain tasks, machines are far more efficient than people. Awareness at a very primitive level has been replicated in machines (Bongard, Zykov, Lipson, 2006), and as a matter of fact is achieved through a method of experience. So, it is safe to extrapolate the term knowledge to a wider variety of systems.

A system has knowledge of how to carry out the task it is meant to do, because, in the worst case, the system was programmed to do it, since programming was proposed equivalent to education, the statement becomes true by definition.

But in the interest of this chapter, how does having knowledge take experience to the next level? From section 2 it can be determined that next level experience starts when the system can extrapolate what was learnt in one problem and use that to solve something else, and it ends when the system has evaluated the level of success on solving the problem. Then, knowledge of other problems is useful when using next level experience. But as experience goes up on level, so does knowledge, because by definition, if there is experience, the information achieved by it is knowledge.

A quick look on what could be seen in next level knowledge would throw probably some algorithms and some indicators on how efficient it was under certain circumstances. There would be an algorithm that would know how to choose and combine algorithms to solve new problems, and there must certainly be an algorithm that would store procedures that had effectively solved a problem. In this case, traditional machine learning algorithms and any algorithm designed to specifically solve a problem becomes an essential component to the algorithms found in next level knowledge.

People store information by creating interconnection between different neurons, part of that information, which is consider knowledge, is actually information about the way people carry out tasks. Some of it is fuzzy knowledge as the person knows that certain algorithm works well under certain cases in a certain way, while other may not work as well. There's also deterministic knowledge of this kind, for example, the way a person writes; clearly there is certainty that the algorithm for writing works every time.

Without the neurons' connections the storing of information wouldn't be possible, and without storage, comparison, characterization and choosing are not feasible. One of the reasons would be that there would be no knowledge (because there's no information) about the efficiency of an algorithm, so there would be no factors in which to base the choice other than randomness; there would also be no information to compare any two algorithms and no information about any algorithm could be generated because it would be immediately forgotten.

As in people, machines have various methods to store information. From the simpler latch or flip-flop all the way trough to quantum dots (Stick, Sterk, Monroe, 2007) and buckyballs(Anderson, 2007) passing by registers, and more traditional R.A.Ms, R.O.Ms, and magnetic hard drives. Although some neuroscientist despise the idea of comparing the human brain to a computer, some similarities can be pointed out; for instance, the "natural instinct" or "born instinct" can be compared to the functionality of the ROM in the computer, the short term memory to the RAM, and the long term memory to the hard drive.

Information in the brain seems to be stored in different sectors of the brain, depending of where it comes from or what it does; in a system, the information also has to be structure to achieve functionality.

By design artificial systems have a "natural" partition, in one hand there's the program memory while on the other is the data memory. In a way, this separates the "how to" from information, as mentioned before a program is knowledge achieved through "education" so this basic natural partition could be sufficient in some cases. However, the downside of this storage strategy is that the size of program memory is usually limited. This lack of space obligates to simplify algorithms and use only a small set of them. It also implies that the complexity of the higher level algorithms (HLA) is reduced to simple lookup tables as the actual algorithms could not be changed or manipulated.

In modern computing systems this lack of capacity is a matter of the past, today it is very inexpensive to have large amounts of memory available. This means a large number of programs and a large amount of information could be made available to a CPU or the processing unit of choice. Under this circumstances, HLA do not have to be limited to a look up table, they can be very sophisticated algorithm that could spawn new versions of basic level algorithms (BLA).

It is clear at this point that in order to have knowledge, there has to be a storage system. And such storage system has to be capable not only of storing data, but it has to be able to store algorithms as well, and if the HLA are sophisticated enough, it must allow them to manipulate the algorithms.

There are three characteristics intrinsic to a storage system of any kind. First of all it must have an appropriate capacity, not too much that the system would have trouble carrying the extra space not too little that algorithms could not work or be worked around easily. And second, the storage system has to be fast, even it means that it must compensate for latencies associated to slow media, it also means that it needs to be organized so it will find the data or algorithm that the HLA is looking for almost immediately. Last but actually the most

important, the storage system has to allow algorithm modification; with ever increasing complexity a good HLA could evolve an algorithm with time, so it is important to allow for such type of action over the algorithm.
Based on the second characteristics, the way an algorithm is stored has a great impact on the overall performance of the system. If the storage system is not fast enough, the system is going to have critical waiting periods while it loads the next algorithm to execute, and if such times are grater than the system's natural response time. The system could become unstable or collapse all together. Therefore it is crucial to structure the storage system to have a fast response.

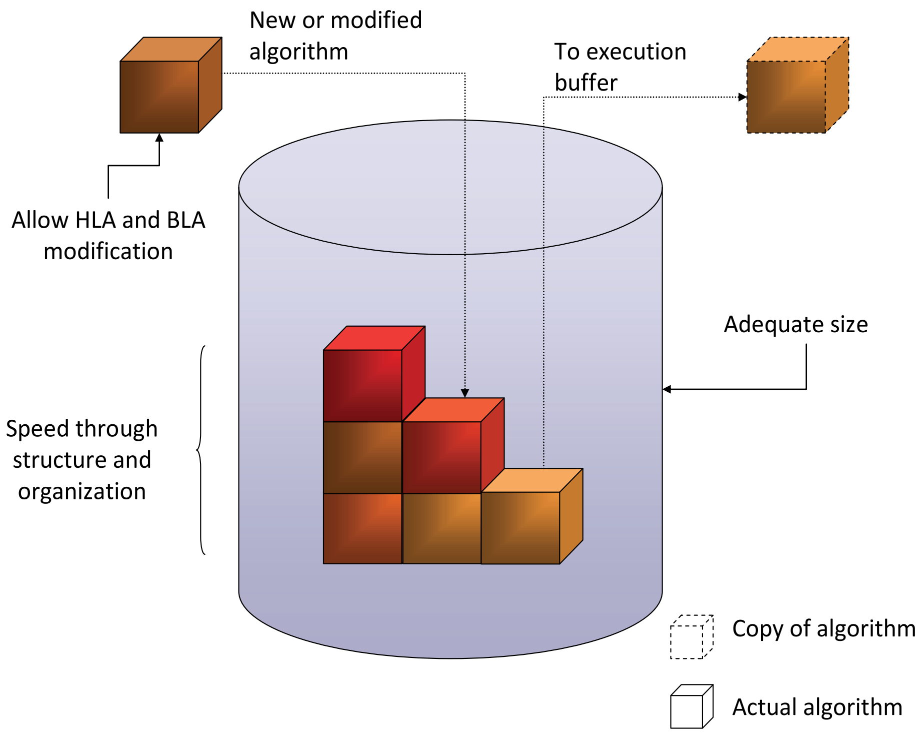

New or modified algorithm
To execution buffer
Allow HLA and BLA modification
Adequate size
Speed through structure and organization
Copy of algorithm
Actual algorithm

Fig. 2. Storage system characteristics

4. Architectures that allow for higher level algorithms.

Any means of storage could be considered a valid architecture for HLAs; however, it is important to keep in mind the three qualities associated for a good storage system for knowledge. Furthermore, any architecture has to provide the means to evaluate or at least have a grading mechanism to choose the appropriate algorithm for the given set of circumstances.
Without evaluation there's no experience to be achieved, because there wouldn't be the means to measure an improvement in certain task. In other words, if there is an HLA, it needs to keep track of how well it has resolved the problems at hand with the BLAs, meaning it needs to evaluate each BLA's performance. So whether the evaluation is

embedded into the HLA or its part of the system design and it is made available to the HLA as a service, it needs to be present.

Turning to the architectures, they can be divided into two groups, software architectures and hardware architectures. Although software architectures are the easiest to implement and the most familiar for developers, resent studies in hardware design are showing promising results. Software based architectures have several advantages, for starters, must of the elements needed to create them are intrinsic to an operating system or a program i.e. multi-thread multi-process operations, file management or dynamic library loading. Other important advantage is the level of possible manipulation; an algorithm can be disassemble and assemble with changed properties. But the downside is that all that preparedness has a high cost in size, operating systems usually take a lot of space in order to give all that functionality as does the additional software.

In contrast, the speed achieved in hardware is dazzling, and with reconfigurable hardware techniques, drastically changing algorithms is possible. The problem is that there's a higher cost in design time, because all the interfaces needed to use massive storage, and reconfigure hardware have to be hard wired and hard coded; also there's less portability to other systems due to the hardware specificity.

Regardless of the technical issues that embrace each technology, it is important to take a look at some examples as for different practical problems there'll be a most appropriate implementation.

4.1 Software architectures: using a file system.

Despite the operating system of preference, it is going to present the developer with a file system. This File system allows the storage of massive amounts of information, and usually lets you handle multiple storage media like USB memories or hard drives with ease.

Figure 3 illustrates the basic layout of an architecture based on a file system. The evaluation subsystem could be an independent module; or as mentioned before, embedded in the HLA. The file type of choice is a dynamically linked library that can be loaded and unloaded as needed. The HLA is the entity that decides which algorithm to load based on the information stored in the evaluation file. The execution module runs the algorithm achieving a change on the system's stat; the efficiency and accuracy of the operation is measured by the HLA and the result is stored through the evaluation module.

The algorithms are recommended to be stored in compiled form, in other words in an executable format, i.e. *.dll* for Windows operating systems. This ensures a faster execution and allows the direct interaction with all of the systems' services; also it allows the direct use multi thread technology, leaving the responsibility of processor time assignment to the operating system.

Interpreted formats, like a Matlab file, are not recommended for the BLA as they become costly to execute because they have to load the interpreter. Also the algorithm has to use the interface provided by the interpreter in order to access the system's services, this usually has an impact on performance and some services are restricted. Things like multi threading depend exclusively on the interpreter of choice so it is not always available. In (Lopera, 2005) the Matlab algorithms always caused the execution time to default to the worst case.

When regarding direct algorithm manipulation by the HLA, a few things have to be taken care of. First of all naming new libraries, the HLA has to keep track of the new libraries created otherwise it might not keep an appropriate performance log and thus, it might not use the newly created algorithms even if they turn out to be more efficient.

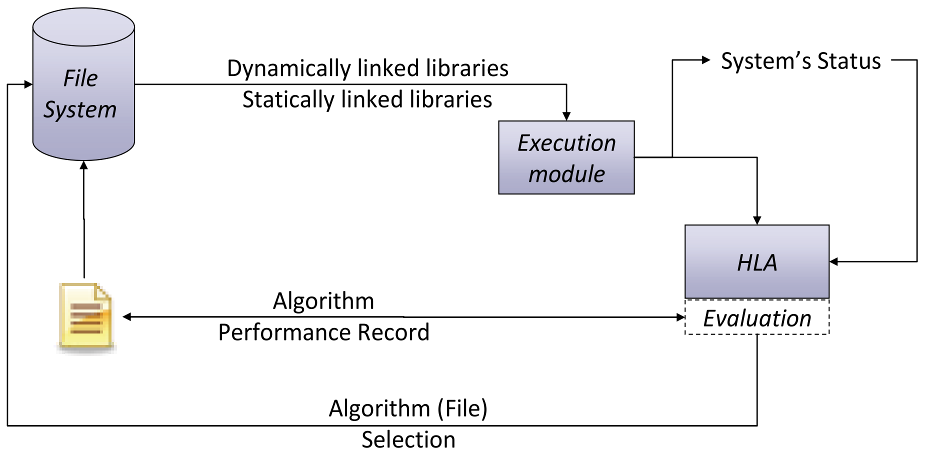


Fig. 3. Basic layout of file system based architecture

Algorithm manipulation is easier to do in interpreted formats because is a natural way to partition and mix functionalities, in compiled formats, it requires more steps but it can be done, weather is combining at a source file level and recompiling, or mixing in binary format; which it hasn't been tested and requires a profound knowledge of the binary structure of the compiled library. This also means that the HLA has to keep track of what source code belongs to which library.

One of the advantages of file system based architecture, especially when using compiled format, is that the system will only load what ever algorithm is executing and the system's services, nothing else, so it can be very efficient in respect to memory usage.

By designed, a file system complies with the characteristics proposed for knowledge storage; however is the responsibility of the HLA to keep the order and structure of the file system. A poorly designed HLA can end up clogging the file system surrendering it inefficient and ultimately halting the system. Other advantage of the file system is its portability; the hardware architecture is some what transparent, as long as it supports the operating system: it will support the file system.

The disadvantages lie with the evaluation module; because is a file based module, all searches have to be carried on within files, so a lot of searching and updating functions have to be written in order to allow the HLA to effectively evaluate and choose BLAs.

4.2 Software architectures: databases

The database architecture is an expansion of the file system architecture; it seeks to improve where the file system presents its most weaknesses. It also takes care of the evaluation structure, which allows having multiple HLAs that share the same information about the algorithms and simplifies overall HLA development.

A database is designed to store information, and as such it allows storage of multiple types of information in an orderly fashion; its internal structure is designed to relate information between tables so it facilitates data management and storage structure, furthermore, it also specializes on information retrieval; it is designed to fetch huge amounts of information in short periods of time. This makes it ideal to take care of storing the algorithms in binary

form as well as in source code form, associating all sorts of parameters that allow the HLA to choose the best algorithm for a more complex context.
Figure 4 illustrates a general architecture, in this case the HLA works with the database to manipulate and evaluate the stored algorithms, once it has chosen one, it retrieves it and saves it to the file system for execution. This because most operating systems don't allow executing information that is considered data, except for executable files on the file system.

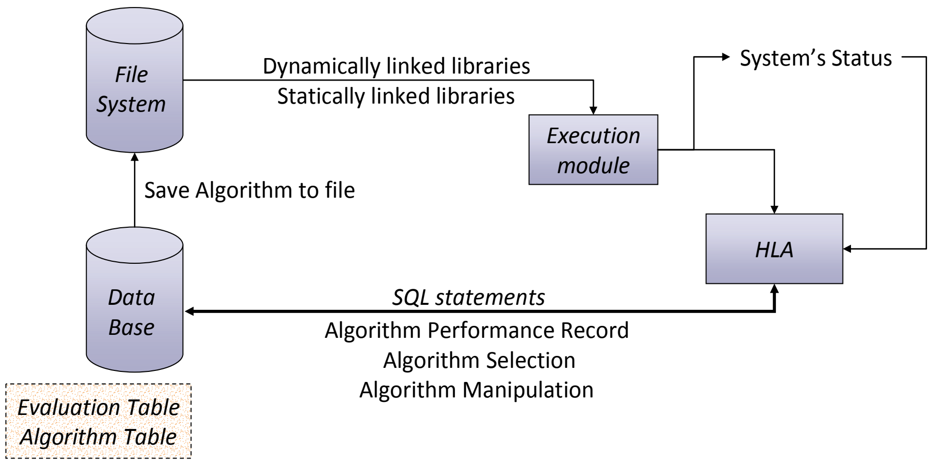


Fig. 4. Basic layout for the database architecture

As mentioned before, the database improves performance and facilitates the job of the HLAs, at the cost of having to load the database server which implies some memory usage and processor time; however for most systems based on pc computers this is not a problem. The advantages outweigh the cost. In (Lopera, 2007) there is an interesting analysis about the pros and cons between both architectures.

4.3 Software architecture: when space is limited

This type of architecture is considering systems that are developed using microcontrollers where access to memory resources is limited and no operating system is available or does not have file system capabilities much less a database server.

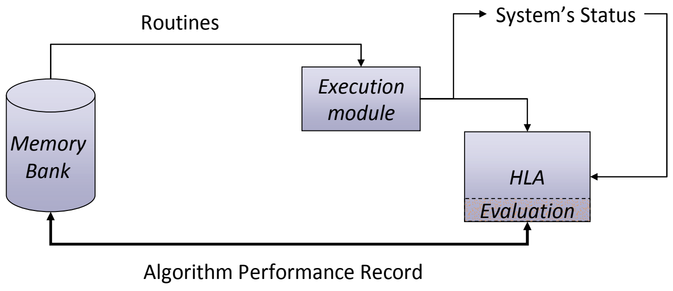


Fig. 5. basic layout for limited space architectures

In this case HLA must have embedded the evaluation module; it should work over a memory area, keeping a rather simple record of performance and link to the respective program counter's position of each routine. It is recommended that the routines be constructed in an interrupt basis so in that way they'll return handle to the HLA so it will be able to monitor the system status and the routine's performance.
To achieve some level of routines manipulation they should be parameterize, in that way the HLA can modify the parameters to fine tune the routine's efficiency.

4.4 Hardware architectures: reconfigurable hardware

A typical architecture for reconfigurable hardware is the cooperation of a processor unit with a programmable electronic device (PED) as PSOC or FPGA. In this configuration the processor has the responsibility of programming the PED, and for that, the processor uses storage memory to store the binary files that contain the programming sequences, usually downloaded in to the PED through JTAG.

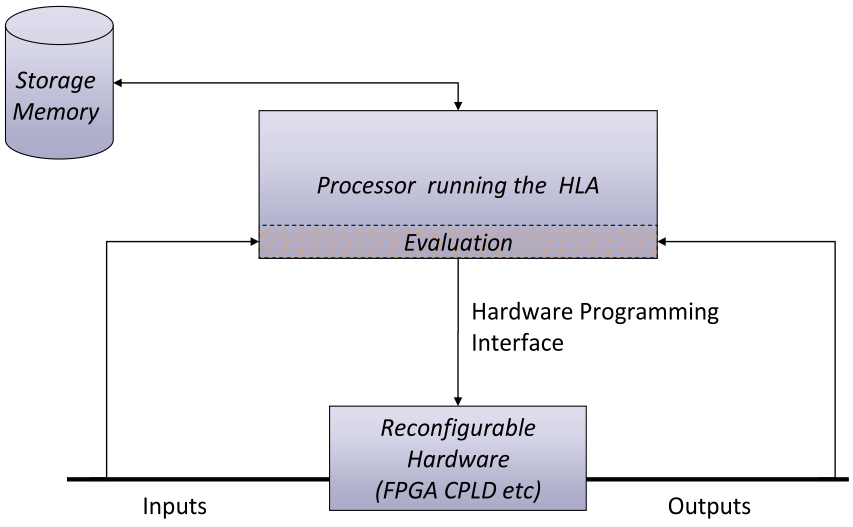

Storage Memory
Processor running the HLA
Evaluation
Hardware Programming Interface
Reconfigurable Hardware (FPGA CPLD etc)
Inputs
Outputs

Fig. 6. basic layout for reconfigurable hardware architectures

In this case there are several configurations that can be carried out, and they all depend on the capacity and speed of the processor as well as the PED. For instance the evaluation module can be run at the processor along with the HLA or can be programmed and configured in the PED so it will match its internal configuration and facilitate performance measurement.
The HLA is recommended to be executing in the non-reconfigurable part of the system as it is pointless to load and reload every time the PED has to go through a programming. This takes up some time, and could compromise system's stability.
Some of this configurations support small operating systems, this operating systems could run small database servers, in this case leaving the BLA to be implemented at hardware level. This certainly has some performance issues that have to be evaluated based on each specific application.

One of the advantages of this architecture is that PED have become interestingly complex and powerful as they have grown in capacity, mixing microcontrollers with analog cells and digital cells. This resource availability can be used to implement high performance BLAs using very up to date design techniques.

4.5 Hardware architectures: non reconfigurable hardware

Not all types of algorithms are worth the trouble of implementing at a hardware level. In most cases due to the repetitiveness and the sequential nature of its internal operations a software architecture is more suitable. Even though, parallel processing, state machines, and other hardware design techniques can be embraced to implement powerful solutions.

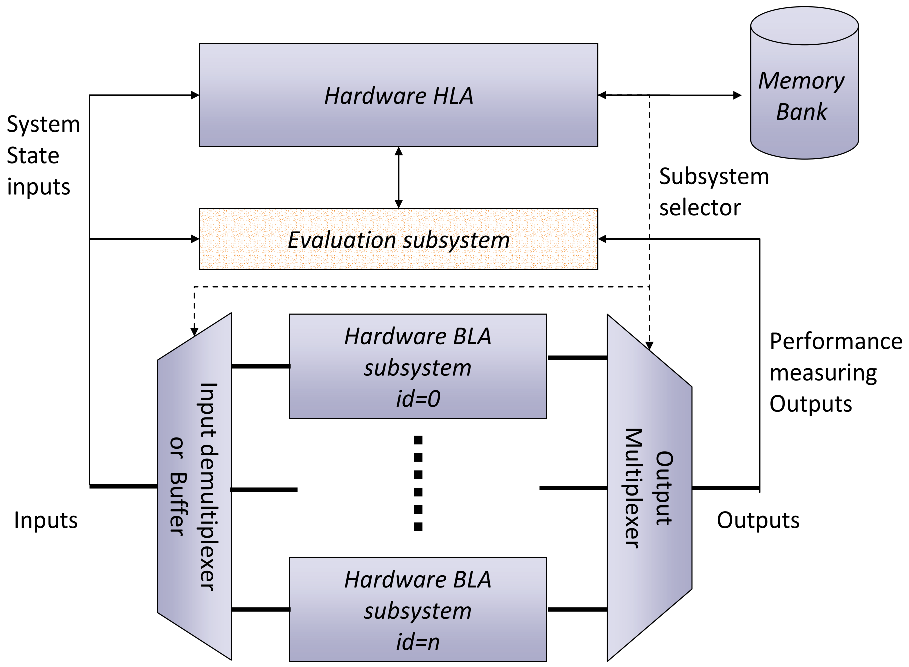

Hardware HLA
Memory Bank
System State inputs
Subsystem selector
Evaluation subsystem
Hardware BLA subsystem id=0
Performance measuring Outputs
Input demultiplexer or Buffer
Output Multiplexer
Inputs
Outputs
Hardware BLA subsystem id=n

Fig. 7. basic layout of non reconfigurable hardware architectures

The way this architecture works is as follows: The HLA controls the output multiplexer and input buffer (or demux) it also must enable the chosen subsystem that will operate over the inputs and produce the appropriate outputs. This choice is based on the information stored in the memory bank. The evaluation subsystem is constantly monitoring the system's state inputs and the outputs selected to measure the hardware algorithm's (HA) performance; it also communicates the results to the HLA which in turn stores that information into the memory bank.

The hardware HLA, from the hardware design point of view, could be conceived as the control unit of the system. The evaluation is considered a separate module in this architecture because based in good hardware design strategies modularity is enforced and

since its job is so distinctly clear and does not mix with any other process the HLA might be doing. The input buffer has to be designed so it will present in adequate form the inputs to the HBLAs, just wiring every system input to the HBLA's input might encounter fan-in, fan-out or loading issues. The output multiplexer is pretty straight forward, the only concern is the signal types, in which case, an appropriate multiplexer has to be designed.
Unfortunately this architecture is the most expensive to implement and the most keen to present problems do to implementation, i.e. wiring and signal coupling issues. Despite its cost and arduous construction, it is worth while presenting this architecture as it illustrates how the HLA can be taken to the must basic level. It also reinforces the following concept: the importance of basic level modularity, which is going to be presented in more detail in section 5. In other words, regarding HBLAs inputs and outputs, they all have to talk the same languages, since they will be connected to the same interfaces; this becomes an important design restriction.

4.6 Remarks

The presented architectures show some alternatives of how to implement HLA and the evaluation mechanism, which are necessary for higher level experience. In general, Figure 8 shows a basic hierarchical structure of how to design an architecture that is considered HLA enabled.

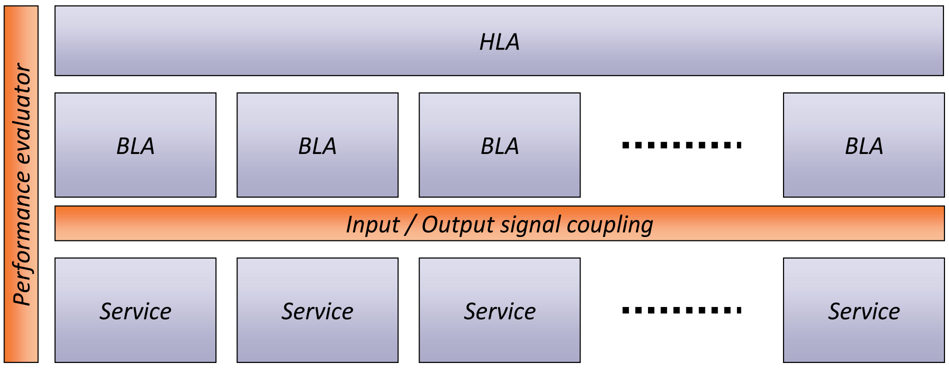


Fig. 8. general architecture

Service based design is crucial for these types of architectures. This way, each BLA knows exactly how to talk and listen to a system service. It also allows the execution of multiple BLA that use independent services at any one time and simplifies the design of the BLA as it only needs to interface with services it requires.

5. How to design robots with Higher Level Algorithms

This section analyses the design procedure of a mobile robot, it does not design a robot itself but assumes that there is certain mechanical infrastructure, hardware, and even software at a service level. The center idea is to structure the general architecture at a high level. For this we assume that the robot is at an advanced stage, in other words the first elements of the design process have been taken care of, the basic physical structure, the control system and electronics of the individual elements like motors, arms, cameras, etc are up and running.

5.1 The robot

BoBoT, which is going to be the robot's name, has three main service subsystems: the first one consist of a set of four independently driven wheels; second, 2 grippers each mounted on a arm with 2 sections and 2 degrees of freedom for each joint (3 in total); and third it has a bundled dual camera system with pan, tilt and zoom capabilities. The brain of the operation is going to be a laptop system and the database architecture is going to be used.
BoBoT is also equipped with a series of sensors that complement the basic instrumentation used to achieve control of the service subsystems:

- A three axis accelerometer
- An up down sensor.
- An applied force sensor for the arms.
- A battery charge meter.
- A GPS

Figure 9, Figure 10 and Figure 11 show the black box models of the service subsystems

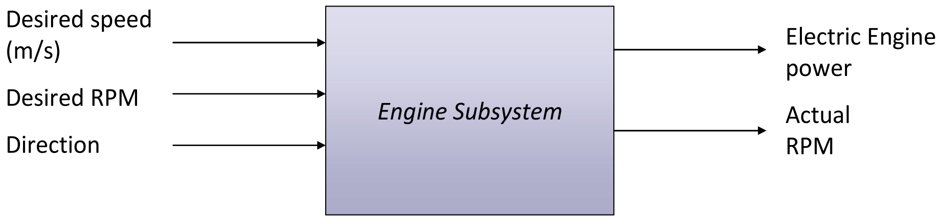


Fig. 9. Black box diagram of the engine subsystem

The engine subsystem has two ways of operation: it can turn by specifying an actual desired speed in meter per second; the engine will turn in the direction implied by the speed's sign. The other way is to establishing the RPMs and a direction. To specify which input to listen to, the unwanted one has to be set to 0. If not, desired speed prevails. In turn, the engine subsystem's outputs inform of the power given to the motor and the measured RPM s.

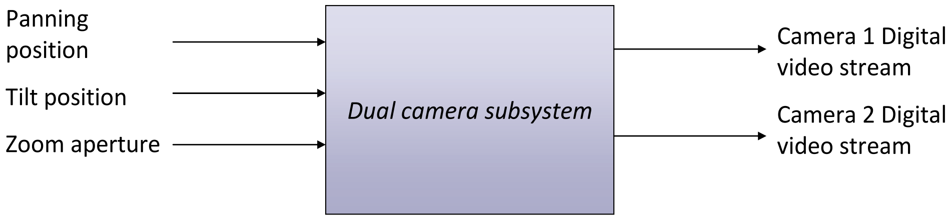


Fig. 10. Black box diagram of de dual camera subsystem

To operate the dual camera subsystem it is sufficient to specify the position in the pan and tilt axis, and how much zoom is desired, the cameras can not be controlled separately. The outputs are the two video streams in a mildly compressed digital format.
To use the arm it is important to understand that the grip operation is independent of arm operation. The grip has two ways of operation: one is by establishing a desired action and a speed of the action, i.e. "close" "fast" and the other is by establishing the action and the forced to be applied, i.e. "close" "hard". In the first example, the grip will close fast and

apply maximum force, in the second it will close slowly until it reaches the desired applied force. The system will constantly give out the grip status, i.e. opened, opening, closed, closing, and the actual force applied.

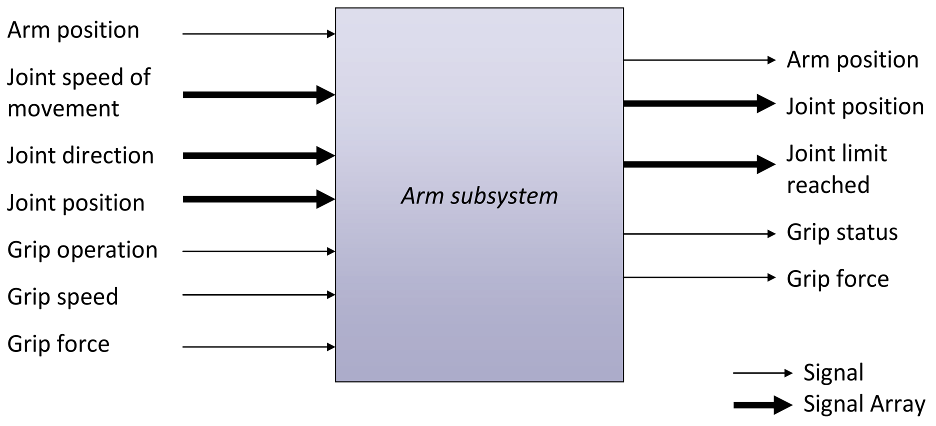


Fig. 11. Black box diagram of arm subsystem

The arm can be operated in three different ways: In the first one, the grip can be positioned in a 3D space with origin at the shoulder. The second way, allows positioning each joint accordingly. And at last, a joint speed and direction can be specified in order to achieve constant movement. And as outputs there are: the grips position with respect to the shoulder, joint position in their local coordinates, and an indicator if any of the joint's limit sensors was reached.
BoBoT has two arm subsystems, 4 engine subsystems and 1 dual camera subsystem.

5.2 The things BoBoT can do:
As part of the design process it is important to know precisely what it is expected of the robot. This section assumes that the robot has to carry out the following actions:
- Vision based navigation with global positioning
- Vision based navigation with inertial positioning
- Vision based navigation with visual terrain recognition for positioning
- Wide turns, forward and backwards.
- Rotations around wheel base center
- Pick up and place delicate objects.
- Pick up and place sturdy object.
- Variable speed and direction.
- Movement with the arms
- Swing
- …

These actions also show that there are commonalities between them, and also give the sense that there is more ways to achieve success, or that they share a common goal, i.e. the first three, the ones using vision based navigation, share the goal of moving from one point to another.

The next step is to identify the possible BLAs, as mentioned before the BLAs have to be extremely modular, so the expected actions not necessarily become BLAs. For instance, to pick up an object BoBoT will have to use vision to identify the object's position and use that position to place the grip at a gripping distance, regardless if it is delicate or sturdy. Thus, there are at least three BLAs, one for object location, one for arm movement, and one to identify if the object is delicate or not so BoBoT can actually grab it.

There can be multiple versions of the BLAs, in the picking up example, moving the arm could be done by controlling the trajectory in a 3D space assuming the trajectory is clear, or also assuming a clear trajectory but monitor the arm's applied force sensor to detect collision, or use the cameras to check for obstacles. If used the latter, the importance of modular service design is critical as the camera would be used by two BLAs. When using HLAs, there's no need to choose one of these three approaches to the same problem, instead you can store all three BLAs and have the evaluation subsystem evaluate them under different circumstances.

To further reassure the importance of modular service design, at least three BLAs can be designed to use the arm modules, one for each input pair, one for 3d positioning, one that uses joint positioning, and other one that uses joint movement. In this case it is simple to develop the BLA, but if instead there were no good service design, each BLA would have to deal with problems related to the direct control of the arm, and maybe wouldn't be as easily interchangeable or their size and complexity would increase.

Once identified all the BLAs with their different versions, the next step is to write them, compile them and individually test them. Also the BLAs have to be tested in group as the way they are expected to be used and correct any interfacing problem that might result from things like resource sharing.

5.3 The storage strategy

Having tested all the BLAs, it is needed to gather the following information:

- Excluding BLAs, those that perform different tasks but can not run at the same time.
- BLAs that perform the same task but in different versions
- Qualifiers of BLA performance
- Environment status variables in which each BLA out performs the others in the same task.
- BLA parameters if any.
- BLAs needed to perform each action
- Qualifiers of action's performance; how efficient was BoBoT to perform the task.
- Switching task times, it is easier to manage system stability at a HLA level, but it only matters when switching times are really critical.
- Which subsystems are used by each BLA

If at this point some incongruence is found among the BLAs they must be corrected before continuing because they might induce critical changes that force to repeat the previews steps.

With this information the database tables can be created; it is recommended but not strictly necessary to: 1 BLA table, 1 BLA parameters table, 1BLA evaluation table, 1 action table, 1 action evaluation table. For the action table it is recommended to use a code, if space is sufficient an extra table could be used to store that code, but it would only be useful for the developer or generating reports, it wouldn't have any effect on the HLA. Figure 12 shows a possible table setup.

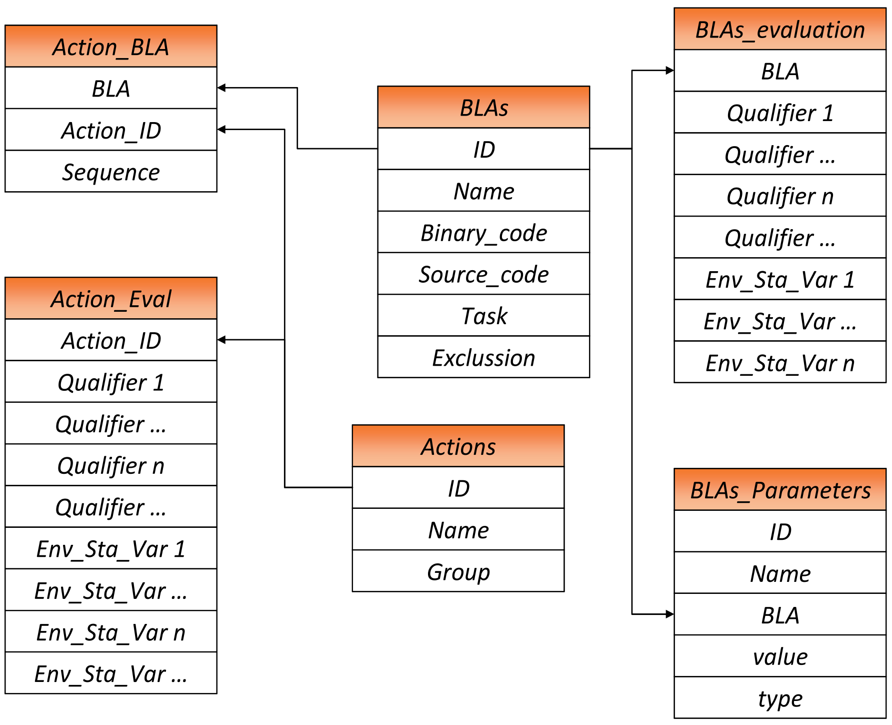


Fig. 12. Table reference diagram

In this setup, there's the *Actions* table that stores the coding but the additional field of *group* allows identifying which actions are the same, so they can be evaluated and associate different, but equivalent, BLAs. Also the *Actions_Eval* table stores information about the Environment Status variables so the HLA can track which combination of BLAs worked best for those conditions of the environment.

In case of using other HLA architecture, the same steps can be followed, only the storage structuring has to be adequate to the choice.

5.6 Finally the HLA

The HLA could have several roles in BoBoT, it could be in charge of fulfilling a mission, deciding the best way to successfully complete it. In this role the HLA would work with the *Action* tables evaluating and calculating constantly course of action, and how far it is to completion.

Other role the HLA could assume is to take a course of action from a user, and follow it; in this case the HLA would work closely with the *BLA_Evaluation* table to choose the best BLA for the given conditions and course of action. A course of action can be expressed in terms of the *Actions* table, the corresponding BLA retrieved from the *Action_BLA* table and the best

BLA from the *BLA_evaluation* searching among the algorithms that share the same value in the task field and are none excluding.
It is important to keep clear the role the HLA is going to take and how is going to take advantage of the tables, if several roles are detected it is a clear sign that the HLA has to be broken into modules, one for each role, and each module assume the appropriate hierarchy. If it turns out that there's something on top of the HLA, those on top could be considered next level HLAs.
Once the HLA's role is established, the type has to be chosen, and there are two types HLA: Those that have programmatic responses, and those that have learnt responses. HLAs with programmatic responses are those algorithms that have transfer function or some mathematical equation that relates the inputs to the outputs and are programmed. In the second type, the HLA learns from experience, it tries actions, evaluates performance and start to mix accordingly to achieve better results. Thus, this type of HLA could be any of several machine learning algorithms, working with other algorithms and a sub set of inputs.
Into what BoBoT is concerned, BLA of al sorts could be written, i.e. to use the four engines individually, in pairs or all together, to use each hand separately or gracefully coordinated, visually inspect the world surrounding him and use vision for a diversity of tasks. He 'would be able to successfully complete hundreds of mission of all sorts.
The level of success can be associated to the complexity or smartness of the HLA, for instance, a very programmatic HLA that was designed for a very specific and stable environment would certainly fail on dynamic environments. However, an adaptive HLA that takes record of how the environment affects its BLA's performance is more likely to succeed.
One of the advantages of using HLAs is that they force the design to be so modular that new BLA could be introduced and the previous work wouldn't be wasted, it will let the HLA evaluate and choose and optimize procedures, and user machine interfacing is done at a more natural way since it could be done by describing actions.
The storage strategy is open for the designer to best choose the tables or structures he needs, and allows to be as sophisticated as to have several levels of associations, or as simple as a few register in the memory bank of a microcontroller.

6. Being practical, final remarks.

In this chapter the discussion has focused on the how to and the what, but it is important to reflect on the "if we should" or the "is it worth it".
A NASA rover sent to mars, even though it seems a promising scenario, is not the best candidate for HLA, at the first glance, because putting it on mars cost a lot of money; and just to have it start trying stuff that won't work and that might cause an unpredictable failure it would be too risky. However, if once the rover has acquired the relevant information materials pictures etc. putting it to try out BLA becomes interesting, at least more interesting than letting it rot there.
The horse gait problem proposed in (Lopera, 2007) which is actually an energy optimization problem is a good example of the power of modularity since each leg is driven differently on each gate, but is worth the trouble of installing an HLA? There's a trick to this problem, and that is that depending on the terrain, especially on its slope, the gait has to be modified

drastically thus its energy consumption. Furthermore, if the gait algorithm (BLA) can be parameterized in order to adjust leg position and rhythm, the HLA becomes a powerful tool since it will start evaluating and adjusting those parameters so the horse would be able to keep doing the gate. But as far as the optimization problem, there would be the need to generate an additional level in which to operate in terms of the speed achieved by each gate, the energy it consumes and the track's layout.

In an industrial application, there's no need to have HLA, because once the process is optimized it would operate like such. The process sequence is usually determined by its nature and there are optimization techniques and algorithms that do this type of process fine tuning rather well.

In multiprocessor/multicore architectures HLAs could be used to supervise the execution of several learning algorithms in parallel to find optimums in highly complex functions. Since it can analyze the topology of the function, it could use the best optimization algorithm for the area of search. In that way, it could also be used to automatically evaluate classifying algorithms.

The appropriate scenarios for HLA are those that present high environment variability, or are highly unstructured, have several possible BLAs, and there's good computational power and memory availability.

The use of HLAs serves as implementation to the problem presented by (Van de Velde 1995) as to how internalize representations. As he puts it in his child example, the walk by holding a hand is an infant BLA that the HLA will perfect until it has a walking by own means BLA, thus constituting the internalization.

7. Future research

There's an interesting discussion, which this chapter purposely avoided getting in to, about if these architectures could be considered as epistemological. It would be interesting to compare what experts in this area have to say.

One line of research that emerges naturally from this proposed architectures, is the involvement of other natural concepts that participate in the experience process, for instance: What use would have concepts like pain or tiredness for a system that has the capacity to choose the way to solve a problem? How could they be implemented and interconnected with the presented architectures?

The presented architectures have a strong hardware based, reality measuring and affecting feeling to it, since they where thought out for physical systems as mobile robots and such. However, it would be interesting to measure the effect of the architecture in purely virtual systems. How would it affect performance compared to more traditional implementations?

And the last couple of question that emerges from this line of reasoning are: How to code creativity? And would we be able to create a HLA that has creativity as one of it biggest traits?

8. Conclusion

This chapter described a few architectures that support a higher level of experience; however they are not the only architectures possible. Any architecture that evaluates and records the performance of basic modules and uses that information to decide which module to use or how to adjust the module's parameters is considered an architecture that supports higher level experience.

It is clear that the intricate relationship between knowledge and experience can be constructed on an artificial system. Furthermore, it can be generated by the system if its architecture and available resources allow it. Unfortunately, the power of the relationship between knowledge and experience and how the system embraces that power is only as good as the HLA allows it to be. In other words, a lookup table HLA would never be able to undertake tasks for which the environment parameters are not within the lookup table.

The architecture has to be carefully chosen for the resources available and the complexity level of the system. As mentioned before, their use in invariant environments, invariant systems and where no learning is involved, becomes a waste of resource and could compromise development time. But, in the other hand, there is little or no knowledge about the environment and it is desired to maximize mission scope, then architectures that support next level experience could simplify the problem dramatically. This simplification occurs in part because the designers do not have to resolve all the possible problems the system could encounter. Instead they solve basic issues, and leave problem solving to the system.

This type of architecture meets the definition by (Van de Velde 1995) of intelligent systems. As it has cognitive knowledge of its environment as evaluation criteria for the BLAs obtained through the inputs subsystems, and uses that knowledge to determine appropriate course of action, establishing a behavior in its environment.

9. References

Josh Bongard, Victor Zykov, Hod Lipson, (2006) "Resilient machines through Continuous self-modeling", Science 17 November 2006: Vol. 314. no. 5802, pp. 1118 – 1121, DOI: 10.1126/science.1133687

Hani Hagras, Martin Colley, Victor Callaghan, (2001) "Life Long Learning and Adaptation for Embedded agents operating in unstructured Environments", IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th Volume 3, Page(s):1547 - 1552 vol.3.

Carl G. Looney (1997), Pattern Recognition Using Neural Networks, oxford university press.

Luis I. Lopera, (2005) "S.N.A.P.A. 'Supervision, navigation and planning architecture': arquitectura de navegación, planificación y navegación para un dirigible no tripulado, Tesis de maestría, Universidad de los Andes, Bogotá Colombia, 2005.

Lopera, L.I.; (2007) "Algorithms Storage System", Electronics, Robotics and Automotive Mechanics Conference, 2007. CERMA 2007 25-28 Sept. 2007 Page(s):370 – 375 Digital Object Identifier 10.1109/CERMA.2007.4367715

Anderson M, (2008) "Buckyballs to boost flash memory", IEEE Spectrum, June 2008, Page 15

Daniel Stick, Jonathan D. Sterk, and Christopher Monroe, (2007) "The trap technique toward a chip based quantum computer", IEEE Spectrum ONLINE, First Published August 2007.

Van de Velde W, (1995) “Cognitive Architectures - From Knowledge Level To Structural Coupling”, L. Steelss (Ed.) The biology and technology of intelligent Autonomous Agents. NATO ASI Series, Series F: Computer and systems Sciences, Vol. 144, pp. 197-221. Springer, Berlin

4

Hamiltonian Neural Networks Based Networks for Learning

Wieslaw Sienko and Wieslaw Citko
Gdynia Maritime University
Poland

1. Introduction

The problem of learning represents a gateway to understanding intelligence in brains and machines. Many researchers believe that supervised learning will become a key technology for extracting information from the flood of data around us. The supervised learning techniques, i.e. learning from examples, can be seen as an implementation of the mappings $\mathbf{y} = \mathbf{F}(\mathbf{x})$, relying on the fitting of given data pairs $\{\mathbf{x}_k, \mathbf{y}_k\}$. The key point is that the fitting should be predictive and uncover an underlying physical law, which is then used in a predictive or anticipatory way. A great number of models implementing the supervised learning techniques have been proposed in literature. Artificial Neural Networks (ANN), Radial Basis Functions (RBF), Support Vector Machines (SVM) and Fuzzy Logic based models (ANFIS) should be here mentioned. Support Vector Machines, distinctive tools for data classification, are the product of statistical learning theory. Recently, a new learning algorithm named Regularized Least Squares Classification (RLSC) has been proposed. The RLSC concept relyies on multivariate function approximation with regularization theory as a natural framework for solving ill-posed problems of approximation. It is worth noting that SVM and RBF models can be regarded as special cases in the framework of approximation and regularization theory. On the other hand, the Hamiltonian Neural Networks (HNN) based orthogonal filters can be regarded as a natural implementation of the regularization technique. Using Hamiltonian Neural Networks based spectrum analysis, recognition, and memorization, gives rise to mapping implementations with skew-symmetric and symmetric kernels. The purpose of this chapter is to present how very large scale networks for learning can be designed by using HNN-based orthogonal filters and, specifically, by using 8-dimensional (octonionic) modules. The unique feature of HNN is the fact that they can exist as either algorithms or physically implementable devices. In this chapter we mainly concentrate on algorithmic description of HNN-based networks. Moreover, since the structures of HNN can be based on the family of Hurwitz-Radon matrices, we present here how to design large scale nonlinear mappings by using neural networks with weight matrices determined by Hurwitz-Radon matrices. Hence, this chapter consists of the following issues:

- Fundamentals of HNN
- Family of Hurwitz-Radon matrices
- RLSC basics

- Orthogonal filter-based approximation
- Modeling classifiers, pattern recognition and associative memories via nonlinear mappings
- Attractors based very large scale associative memories
- Conclusions

There is a large literature on the subject of networks for learning. Here we only refer to some comprehensive and useful, from the point of view of our presentation, reviews: (Evgeniou et al., 2000), (Poggio & Smale, 2003), (Boucheron et al., 2005), (Predd et al., 2006).

2. Hamiltonian neural networks

It is well known that a general description form of an autonomous Hamiltonian network is given by the following state-space equation:

$$\dot{\mathbf{x}} = \mathbf{J}\mathbf{H}'(\mathbf{x}) = \mathbf{v}(\mathbf{x}) \tag{1}$$

where: $\mathbf{x}$ - state vector, $\mathbf{x} \in R^{2n}$

$\mathbf{v}(\mathbf{x})$ – a nonlinear vector field

and: $-\mathbf{J} = \mathbf{J}^T = \mathbf{J}^{-1}$ i.e. $\mathbf{J}$ is skew-symmetric and orthogonal.

Function H($\mathbf{x}$) is a Hamiltonian (energy) of the network. Since Hamiltonian networks are lossless (there is no dissipation of energy), their trajectories in the state space can be very complex for $t \to \pm\infty$. It is, however, worth noting that Eq.(1) has constant solutions, i.e. every points $\mathbf{x}_0 \in R^{2n}$ such that $\mathbf{H}'(\mathbf{x}_0) = \mathbf{0}$ is the equilibrium and $\mathbf{x}(t) \equiv \mathbf{x}_0$ is the solution. Equation (1) gives rise to the modeling of Hamiltonian Neural Networks, as follows:

$$\dot{\mathbf{x}} = \mathbf{W}\mathbf{\Theta}(\mathbf{x}) + \mathbf{d} \tag{2}$$

where: **W-** (2n×2n) skew-symmetric orthogonal weight matrix

$\mathbf{\Theta}(\mathbf{x})$ – vector of activation functions

$\mathbf{d}$ – input vector (input data)

and: $\mathbf{\Theta}(\mathbf{x}) \equiv \mathbf{H}'(\mathbf{x})$

E=H($\mathbf{x}$) - the energy absorbed by HNN

It can be easy seen that HNN, as described by Eq.(2), is a compatible connection of n elementary building blocks – lossless neurons (Fig.1).

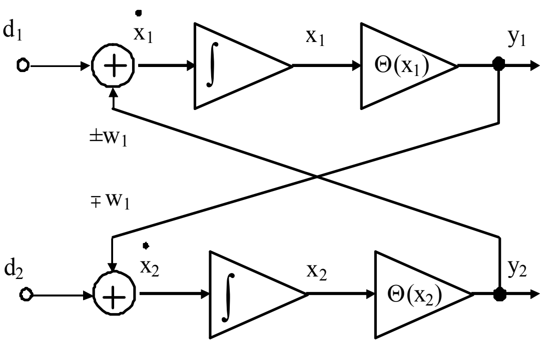


Fig.1. Structure of a lossless neuron.

The state-space description of a lossless neuron is as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & \pm w_1 \\ \mp w_1 & 0 \end{bmatrix} \begin{bmatrix} \Theta(x_1) \\ \Theta(x_2) \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{3}$$

where the activation function $\Theta(x_i)$, i = 1, 2 has been assumed as a passive nonlinearity, i.e.:

$$\mu_1 \le \frac{\Theta(x_i)}{x_i} \le \mu_2 \;;\; \mu_1, \mu_2 \in (0, \infty) \tag{4}$$

A lossless neuron is an elementary Hamiltonian network with absorbed energy given by:

$$E = E_1 + E_2 = \int_0^{x_1} \Theta(\varsigma_1) d\varsigma_1 + \int_0^{x_2} \Theta(\varsigma_2) d\varsigma_2 \ge 0 \tag{5}$$

Formula (5) can be directly extended onto n-neuron HNN:

$$E = \sum_{i=1}^{n} E_i$$

where: E_i – energy absorbed by the i-th neuron.

Note, that for weight matrix **W** skew-symmetric but nonorthogonal, Eq.(2) describes a lossless neural network. The Hamiltonian neural network described by Eq.(1) cannot be realized as a macroscopic scale physical object. But HNN determines a type of orthogonal transformation, namely:

$$\mathbf{W} \cdot \mathbf{\Theta}(\mathbf{x}) + \mathbf{d} = \mathbf{0} \tag{6}$$

$$\mathbf{y} = \mathbf{\Theta}(\mathbf{x}) = \mathbf{W}\mathbf{d} \tag{7}$$

$(\mathbf{y}, \mathbf{d}) = 0$; $(\cdot\,,\cdot)$ – scalar product

Rows (and columns) of **W** constitute an orthogonal Haar basis. The components of output vector are Haar coefficients. Thus, Haar spectrum analysis using HNN is given by:

$$\mathbf{y} = \mathbf{W}\,\mathbf{d} \quad \text{and} \quad \mathbf{d} = -\mathbf{W}\,\mathbf{y} \tag{8}$$

and formula (8) can be used as an algebraic transformation. The problem of physical realizability of HNN can be solved by using HNN-based orthogonal filters. A basic structure of such filters is shown in Fig.2.

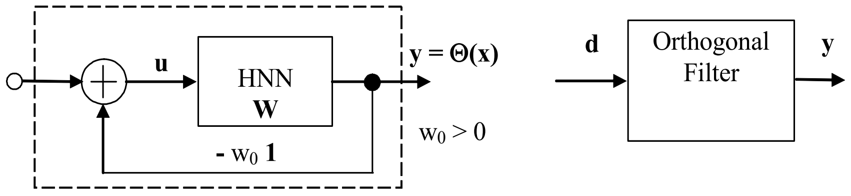


Fig. 2. Structure of an orthogonal filter.

It is worth noting that an orthogonal filter performs the following decomposition:

$$\mathbf{d} = \mathbf{u} + w_0\,\mathbf{y} \tag{9}$$

where: **u**, **y** are orthogonal i.e. (**u**, **y**) = 0
Moreover, Eq.(9) sets up the following orthogonal transformation ($\mathbf{W}^2 = -\mathbf{1}$):

$$\mathbf{y} = \frac{1}{1+w_0}(\mathbf{W} + w_0\mathbf{1})\mathbf{d} \tag{10}$$

where: $(\mathbf{d}, \mathbf{y}) \neq 0$
The output vector $\mathbf{y} = \boldsymbol{\Theta}(\mathbf{x})$ constitutes the Haar spectrum of input data **d**. Since, however, $\mathbf{y} = \boldsymbol{\Theta}(\mathbf{x})$ is the output of a nonlinear dynamical system, Eq.(10) is true for bounded input only. It means, for example, that for neuron activation functions of sigmoidal type, the following conditions have to be fulfilled:

$$|\Theta(x_i)| \leq b_i\,,\ i = 1, 2, \dots, 2n$$

where: b_i – asymptotical value of a sigmoid
Some orthogonal filter based transformations, given by the following formulae, are useful for further consideration:

$$\mathbf{d} = (\mathbf{W}^T + w_0\mathbf{1})\,\mathbf{y} \tag{11}$$

product of transformations ($w_0 = 1$):

$$\mathbf{y} = \frac{1}{4}(\mathbf{W} + \mathbf{1}) \cdot (\mathbf{W} + \mathbf{1})\mathbf{d} = \frac{1}{2}\mathbf{W}\mathbf{d} \tag{12}$$

orthogonalization of outputs for given **d**:

$$\mathbf{y}_1 = \frac{1}{2}(\mathbf{W} + \mathbf{1})\mathbf{d}$$

$$\mathbf{y}_2 = \frac{1}{2}(\mathbf{W}^T + \mathbf{1})\mathbf{d}$$

hence:

$$\mathbf{y}_1^T \cdot \mathbf{y}_2 = (\mathbf{y}_1, \mathbf{y}_2) = 0 \tag{13}$$

Note that the transformations given by formulae (10), (11), (12) and (13) can be regarded either as algebraic algorithms or as physically implementable HNN-based orthogonal filters. Such an implementation is guaranteed by the stabilizing action of negative feedback loops, even if the weight matrix **W** is not exactly skew-symmetric.

3. Hurwitz-Radon matrices

As mentioned above, the main issue in HNN-based orthogonal filters is forming the weight matrices **W** – skew symmetric and orthogonal. The most adequate mathematical framework for this task seems to be an algebraic theory of Hurwitz-Radon matrices. Renewed interest in this old algebraic theory of Hurwitz-Radon matrices can be recently observed. Particularly, a link between this important old matrix problem and refined methods of algebraic topology (homology theories) has been established (Eckmann, 1999), (Vakhania, 1993). The purpose of these considerations is to show how Hurwitz-Radon matrices can be used in design of orthogonal filters. Hence, we provide, below, some basic statements from the theory of Hurwitz-Radon matrices. Let us note that a set of real N×N matrices $\mathbf{A}_j$ fulfilling the following equation (so called Hurwitz matrix equation):

$$\mathbf{A}_j^2 = -\mathbf{1}, \quad \mathbf{A}_j\mathbf{A}_k + \mathbf{A}_k\mathbf{A}_j = \mathbf{0} \tag{14}$$

for $k \neq j$, $k = 1, \dots, s$; $\mathbf{1}$-unit matrix
is called Hurwitz-Radon family matrices (HR family). The matrices $\mathbf{A}_j$ of family are orthogonal, i.e. $\mathbf{A}_j = -\mathbf{A}_j^T$, $\mathbf{A}_j^{-1} = \mathbf{A}_j^T$, $j = 1, \dots, s$. The maximum possible number s of family members for given dimension N is determined by the Radon number $\rho(N)$. It can be found, as follows:
Let $N = 2^a b$, where b is an odd number and $a = 4c + d$; $0 \leq d \leq 4$; $c \geq 0$. Then the Radon number $\rho(N)$ is given by:

$$\rho(N) = 8c + 2^d \tag{15}$$

and such a family consists of s_{max}(N×N)-matrices, where:

$$s_{max} = \rho(N) - 1 \tag{16}$$

Generally: $\rho(N) \leq N$ and for $N = 2, 4, 8$ only, $\rho(N) = N$ and $s_{max} = N - 1$.
Thus, for example, Hurwitz-Radon family of 8-dim. matrices consists of 7 matrices. The following issues in Hurwitz-Radon theory, useful for further consideration, are worth noting:

1. Algebra of complex numbers, quaternions and octonions, is directly related to Hurwitz-Radon families for N = 2, 4, 8, respectively.
2. Maximum number of continuous orthonormal tangent vector fields on sphere $S^{N-1} \in R^N$ is given by $s_{max} = \rho(N) - 1$. Moreover, let $\mathbf{A}_1, \dots, \mathbf{A}_s$ be a family of Hurwitz-Radon integer {-1, 0, 1} matrices. Let $\mathbf{A}_0 = \mathbf{1}$ and $a_0, \dots, a_s$ be real numbers with $\sum_{i=1}^{s} \alpha_i^2 = 1$. Then N×N matrix:

$$\mathbf{A}(\mathbf{a}) = \sum_{i=1}^{s} a_i \mathbf{A}_i \tag{17}$$

   is orthogonal and $\mathbf{A}(\mathbf{a})$ can be considered as a map of sphere S^s into orthogonal group O(N).
3. All 8-dim. HR matrices have the following form (s_{max}=7)

$$\mathbf{H}_8 = \begin{bmatrix} 0 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 \\ -h_1 & 0 & h_3 & -h_2 & h_5 & -h_4 & -h_7 & h_6 \\ -h_2 & -h_3 & 0 & h_1 & h_6 & h_7 & -h_4 & -h_5 \\ -h_3 & h_2 & -h_1 & 0 & h_7 & -h_6 & h_5 & -h_4 \\ -h_4 & -h_5 & -h_6 & -h_7 & 0 & h_1 & h_2 & h_3 \\ -h_5 & h_4 & -h_7 & h_6 & -h_1 & 0 & -h_3 & h_2 \\ -h_6 & h_7 & h_4 & -h_5 & -h_2 & h_3 & 0 & -h_1 \\ -h_7 & -h_6 & h_5 & h_4 & -h_3 & -h_2 & h_1 & 0 \end{bmatrix} \tag{18}$$

where: $h_i \in R$, $i = 1, \ldots, 7$.
Similarly for N =16 HR family consists of s_{max}= 8 matrices and all 16-dim. matrices can be found according to the following structure:

$$\mathbf{H}_{16} = \left[\begin{array}{c:c} \mathbf{H}_8 & \begin{matrix} h_8 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & h_8 \end{matrix} \\ \hdashline \begin{matrix} -h_8 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & -h_8 \end{matrix} & \mathbf{H}_8^T \end{array}\right] = \begin{bmatrix} \mathbf{H}_8 & \mathbf{0} \\ \mathbf{0} & -\mathbf{H}_8 \end{bmatrix} + h_8 \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{-1} & \mathbf{0} \end{bmatrix} \tag{19}$$

where: $h_8 \in R$.
For N = 32, ρ(N) = 10 and s_{max}=9. All 32- dim. HR matrices can be found as:

$$\mathbf{H}_{32} = \begin{bmatrix} \mathbf{H}_{16} & \mathbf{0} \\ \mathbf{0} & -\mathbf{H}_{16} \end{bmatrix} + h_9 \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{-1} & \mathbf{0} \end{bmatrix} \tag{20}$$

Note that the number of free parameters h_i in $\mathbf{H}_8$, $\mathbf{H}_{16}$ and $\mathbf{H}_{32}$ is equal s_{max}. For dimension $N = 2^k$, k = 6, 7, … all 2^k- dim. HR matrices can be similarly found, i.e.

$$\mathbf{H}_{2^k} = \begin{bmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{0} \\ \mathbf{0} & -\mathbf{H}_{2^{k-1}} \end{bmatrix} + h_K \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{-1} & \mathbf{0} \end{bmatrix} \tag{21}$$

where: $h_K \in R$.
But, then the number of free parameters is smaller than $s_{max} = \rho(N) - 1$ ($K < s_{max}$). HR matrices of dimension $N = 2^k$ are particularly interesting due to their structures- the connections of 8-dim. blocks can be here recognized.

4. Taking into account the definition of HNN given by Eq.(2), weight matrices **W** can be implemented by using HR matrices (e.g. $\mathbf{H}_{2^k}$). Moreover, adding diagonal matrix $h_0\mathbf{1}$, where dim $\mathbf{1} = 2^k$, to skew-symmetric matrix $\mathbf{H}_{2^k}$, we obtain an implementation of the orthogonal transformation from Eq.(10), as follows:

$$\mathbf{y} = \frac{1}{1+h_0^2}(\mathbf{H}_{2^k} + h_0\mathbf{1})\mathbf{d} \tag{22}$$

where: $h_0 > 0$.

It is worth noting that for 8-dim. weight matrix $\mathbf{H}_8$, Eq.(22) describes either the following orthogonal transformation:

$$\mathbf{y} = \frac{1}{1+h_0^2}(\mathbf{H}_8 + h_0\mathbf{1})\mathbf{d} \tag{23}$$

or an equivalently 8-dim. orthogonal filter, as shown in Fig.3.

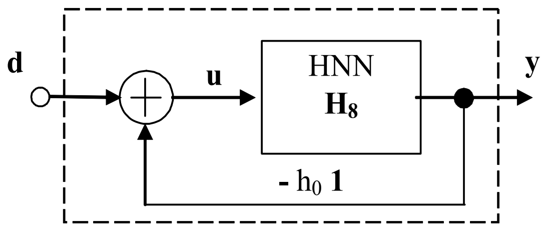


Fig. 3. Structure of 8-dim. orthogonal filter

This type of orthogonal filter will be further called the octonionic module. Because in Eq.(23) we have in disposition eight free design parameters; h_0, h_1 , … , h_7, so this equation allows us to formulate and to solve the following inverse problem: for given input vector $\mathbf{d}_0$ and given output $\mathbf{y}_0$ find parameters h_0, h_1 , … , h_7 such that $\mathbf{d}_0$ is transformed into $\mathbf{y}_0$ ($\mathbf{d}_0 \rightarrow \mathbf{y}_0$). In other words, we set up a best adapted basis for given $\mathbf{d}_0$ and $\mathbf{y}_0$. An adequate solution is given by:

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} = \frac{1}{\sum_{i=1}^{8} y_i^2} \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \\ -y_2 & y_1 & -y_4 & y_3 & -y_6 & y_5 & y_8 & -y_7 \\ -y_3 & y_4 & y_1 & -y_2 & -y_7 & -y_8 & y_5 & y_6 \\ -y_4 & -y_3 & y_2 & y_1 & -y_8 & y_7 & -y_6 & y_5 \\ -y_5 & y_6 & y_7 & y_8 & y_1 & -y_2 & -y_3 & -y_4 \\ -y_6 & -y_5 & y_8 & -y_7 & y_2 & y_1 & y_4 & -y_3 \\ -y_7 & -y_8 & -y_5 & y_6 & y_3 & -y_4 & y_1 & y_2 \\ -y_8 & y_7 & -y_6 & -y_5 & y_4 & y_3 & -y_2 & y_1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \end{bmatrix} \tag{24}$$

Thus, Eq.(24) can be regarded as a design formula for an octonionic module. It is interesting to note that a classical perceptron performing a scalar product of input data **d** and memory vector **m** can be implemented by the octanionic module with best adaptive basis ($\mathbf{m} \rightarrow \mathbf{y}_1 = [\,1, \ldots ,1]^T$), as presented in Fig. 4.

The implementation in Fig. 4 relies on a linear summing of the output flat spectrum of the orthogonal filter.

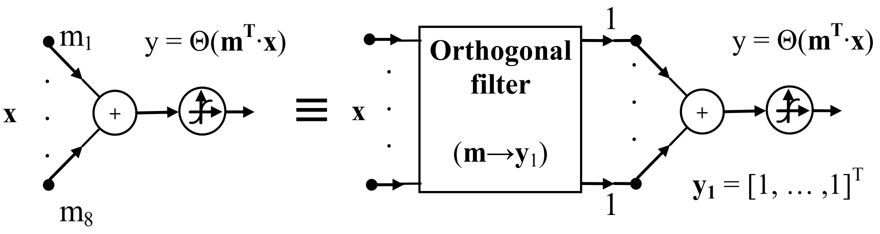


Fig. 4. Implementation of perceptron by octonionic module.

4. RLSC basics

The problem of learning from examples is about predicting the unknown class of observations generated by the underlying physical system. In the last decade the learning problems have been formalized by probabilistic setting, giving rise to statistical learning theory. As products of learning theory, some new and effective techniques, like boosting and support vector machines have been developed. On the other hand, approximation theory, supported by regularization theory, provides a new perspective on learning theory. Regularization theory has been introduced as a natural framework for solving ill-posed problems of approximation (Evgeniou et al., 2000). The purpose of this section is to provide some basic knowledge of Regularization Networks (RN) and, particularly, of RLSC, useful for further consideration. We limit ourselves to briefly describing the main ideas in a simple way.

As mentioned above, learning issues can be formulated as a problem of approximating a multivariate function from sparse data. Starting with training pairs $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$, where input vectors $\mathbf{x}_i \in X \subset R^n$ and $\mathbf{y}_i \in Y \subset R$, one can synthesize a function which represents the relation between the input **x** and **y**, in the best way. In the language of statistics this means that the probability of error f(**x**) ≠ y should be minimal. According to (Evgeniou et al., 2000) the most general framework, unifying several learning techniques can be formulated by considering functionals of the form:

$$H(f) = \frac{1}{m}\sum_{i=1}^{m} V(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_K^2 \qquad (25)$$

where: $f : X \to Y$

$V(\cdot\,,\cdot)$- loss function

λ - regularization parameter

$\|f\|_K^2$ - norm in a Reproducing Kernel Hilbert Space (RKHS)

K - kernel (positive definite function)

The synthesized function f(**x**) corresponds to the minimum of functional H for different loss functions V. Choosing square loss V (L_2 loss function):

$$V(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2 \qquad (26)$$

the approximation scheme arises from the minimization of quadratic functional:

$$\min_{f \in H} H(f) = \frac{1}{m}\sum_{i=1}^{m} V(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_K^2 \tag{27}$$

where: $\lambda > 0$;

H - Reproducing Kernel Hilbert Space (RKHS) defined by symmetric, positive definite function K(**x**, **y**)

$\|f\|_K^2$ - norm in this RKHS

Thus, Eq.(27) presents the classical Tikhonow minimization problem formulated and solved in his regularization theory. It can be shown that the function that solves Eq.(27), i.e. that minimizes the regularized quadratic functional, has the form:

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i K(\mathbf{x}, \mathbf{x}_i) \tag{28}$$

where: $\mathbf{c} = [\, c_1, \dots, c_m]^T$
and kernels $K(\mathbf{x}, \mathbf{x}_i)$ are symmetric, i.e. $K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{x}_i, \mathbf{x})$, positive definite functions continuous on X×X. The coefficients c_i are such as to minimize the error on the training set, i.e. they satisfy the following linear system of the equations:

$$(\mathbf{K} + \lambda\mathbf{1})\mathbf{c} = \mathbf{y} \tag{29}$$

where: **K** is square positive-definite matrix with elements $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j,)$ and **y** is the vector with coordinates y_i. The equation (29) is well-posed, hence a numerical stable solution exists:

$$\mathbf{c} = (\mathbf{K} + \lambda\mathbf{1})^{-1}\mathbf{y} \tag{30}$$

and, moreover, the regularization parameter $\lambda > 0$ determines the approximation errors. It is worth noting that:

1. an approximation is stable if small perturbations in the input data $\mathbf{x}_i$ do not substantially change the performance of the approximator. Hence, the regularization parameter λ can be regarded as the stability control factor.
2. a construction of effective kernels is a challenging task. One of the most distinctive kernels is the Gaussian:

$$K(\mathbf{x}, \mathbf{x}_i) = e^{-\|\mathbf{x}_i - \mathbf{x}\|^2 / 2\sigma^2} \tag{31}$$

leading to RBF networks.

4. Orthogonal filter-based approximation

The purpose of our considerations is to show how a function, given at limited number of training data $\mathbf{x}_i$, can be implemented in the form of composition of HNN based orthogonal filters.
Define f: X→Y by:

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i K(\mathbf{x}, \mathbf{x}_i) \tag{32}$$

where kernels K($\mathbf{x}_i$, $\mathbf{x}$) are defined by the following function (induced by the activation function of neuron, Eq.(4)):

$$K(\mathbf{x},\mathbf{x}_i) = \Theta(\mathbf{x}_i^T \mathbf{H}_n \mathbf{x}) \tag{33}$$

where: $\mathbf{x}_i^T = [x_1,\ldots,x_n], \mathbf{x}_i \in R^n$ is i-th training vector

$\mathbf{H}_n$ is skew-symmetric matrix

$\Theta(\cdot)$ is an odd function

Hence:

$$\mathbf{x}_i^T \mathbf{H}_n \mathbf{x}_i = 0 \tag{34}$$

and

$$\mathbf{x}_i^T \mathbf{H}_n \mathbf{x}_j = -\mathbf{x}_j^T \mathbf{H}_n \mathbf{x}_i \tag{35}$$

Thus, the matrix

$$\mathbf{K} = \left\{K_{ij}\right\} = \left\{K(\mathbf{x}_i,\mathbf{x}_j)\right\} \tag{36}$$

is skew-symmetric

Notice that in the case of kernels given by Eq.(33), regulizer $\|f\|_K^2$ in Eq.(26) is seminorm i.e.:

$$\begin{aligned} \|f\|_K^2 &= \left(\sum_{i=1}^{m} c_i K(\mathbf{x}_i,\ \bullet), \sum_{j=1}^{m} c_j K(\mathbf{x}_j,\ \bullet)\right) = \sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j (K(\mathbf{x}_i,\ \bullet),\ K(\mathbf{x}_j,\ \bullet)) = \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j (K(\mathbf{x}_i,\ \mathbf{x}_j) = \mathbf{c}^T \mathbf{K}\mathbf{c} = 0 \end{aligned} \tag{37}$$

Despite the property given by Eq.(37), we use the key approximation algorithm as formulated by Eq. (29) and (30), i.e. the regularized kernel matrix takes the form:

$$\mathbf{K}_R = (\gamma \mathbf{1} + \mathbf{K}) \tag{38}$$

where: $\gamma > 0$

$\mathbf{K}$ –skew-symmetric kernel matrix

Thus, the key design equation is well-posed:

$$\mathbf{c} = \mathbf{K}_R^{-1}\mathbf{y} = (\gamma \mathbf{1} + \mathbf{K})^{-1}\mathbf{y} \tag{39}$$

It is easy to see that the type of regularization proposed by Eq.(38) means that one changes the type of $\Theta(\cdot)$ function, in kernel definition, as follows:

$$\Theta(\bullet) \to \Theta(\bullet) + \gamma \cdot \gamma_0(\bullet) = \Theta_r(\bullet) \tag{40}$$

where: $\gamma > 0$

$\gamma_0(\cdot)$ – distribution, e.g. $\gamma_0(p) = \lim_{\delta \to 0} e^{-p^2/\delta^2}, p \in R$

In other words, the activation function Θ(·) should be endowed with "a superconducting impulse γ" as shown in Fig.5.

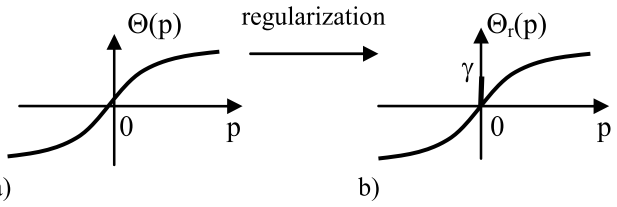


Fig. 5. Regularization by adding γ impuls.

The mechanism of stabilization by means of $\Theta_r(\cdot)$ can be easy explained when one considers the solution of Eq.(39) in a dynamical manner. Such an orthogonal filter-based structure, solving Eq.(39), is shown in Fig.6.

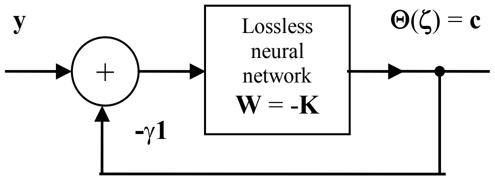


Fig. 6. Structure of orthogonal filter for solution of Eq.(39).

The state-space description of the filter from Fig.(6). is given by:

$$\dot{\varsigma} = -(\gamma\mathbf{1} + \mathbf{K})\mathbf{\Theta}(\varsigma) + \mathbf{y} \tag{41}$$

and the output in steady state as:

$$\mathbf{c} = \mathbf{\Theta}(\varsigma) = (\gamma\mathbf{1} + \mathbf{K})^{-1}\mathbf{y} \tag{42}$$

The stability of approximation in the sense mentioned above can be achieved by damping influence of parameter γ. One of the possible architectures implementing approximation equation (32) is schematically shown in Fig.7 (Sienko & Zamojski, 2006).

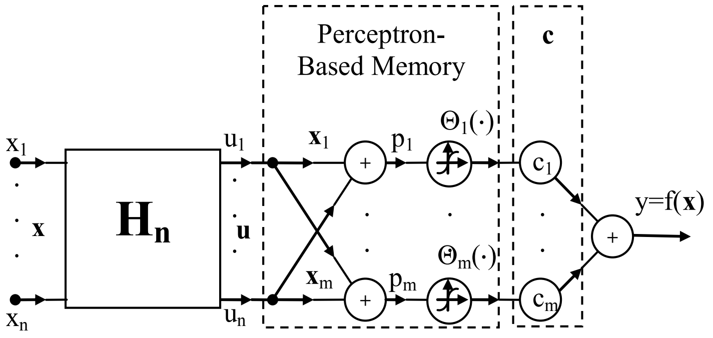


Fig. 7. Basic structure of function approximator.

This structure consists of three basic blocks:

1. Block $\mathbf{H}_n$, where matrix $\mathbf{H}_n$ is randomly skew-symmetric or $\mathbf{H}_n$ belongs to Hurwitz-Radon family, e.g. $\mathbf{H}_n = \mathbf{H}_{2^k}$ (Eq.(21)).
2. Perceptron-Based Memory consists of m perceptrons, each designed at points $\mathbf{x}_i$ of one of the m training points, for any $m < \infty$. Note that activation functions $\Theta_i(\cdot)$, i =1, …, m are odd functions (e.g. sigmoidal) allowing for error approximation at training points $\mathbf{x}_i$. Modeling a nonsmooth function only, they have to be extended by γ impulses.
3. Block of parameters c_i. Note that an implementation of a mapping $\mathbf{y} = F(\mathbf{x})$ needs l such blocks, where $l = \dim \mathbf{y}$.

   The approximation scheme, illustrated in Fig.7., can be described by:

$$\mathbf{p} = \mathbf{S}_{m,n} \cdot \mathbf{H}_n \cdot \mathbf{x} \tag{43}$$

and

$$f(\mathbf{x}) = \boldsymbol{\Theta}^T(\mathbf{p}) \cdot \mathbf{c} \tag{44}$$

where: $\mathbf{H}_n$- (nxn) skew-symmetric matrix
$\mathbf{S}_{m,n}$ – (m×n) memory matrix, $\mathbf{S}_{m,n} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^T$
$\boldsymbol{\Theta}(\mathbf{p}) = [\Theta(p_1), \Theta(p_2), \ldots, \Theta(p_m)]^T$
$\mathbf{c} = [c_1, c_2, \ldots, c_m]^T$

Another orthogonal filter-based structure of function approximator, is shown in Fig.8 (Sienko & Citko 2007).

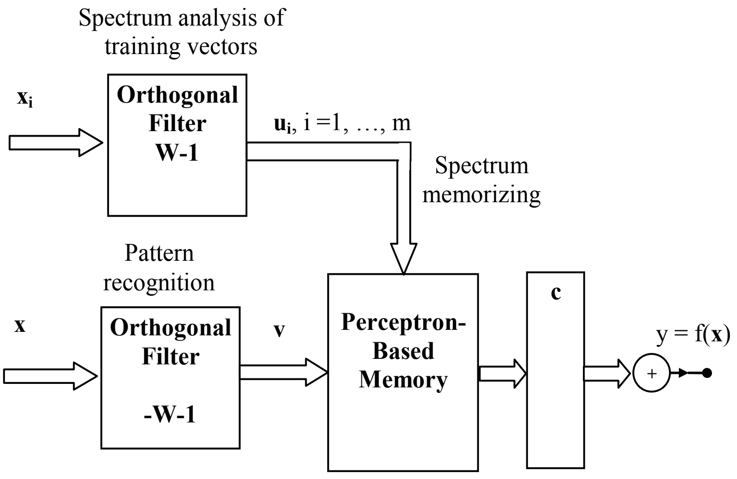


Fig. 8. Orthogonal Filter-Based structure of an approximator.

The structure of the approximator shown in Fig.8. relies on using the skew-symmetric kernels, as given by:

$$K(\mathbf{u}_i, \mathbf{v}) = \Theta(\mathbf{u}_i^T \mathbf{v}) \tag{45}$$

where: $\mathbf{u}_i = (\mathbf{W}\text{-}\mathbf{1})\,\mathbf{x}_i$

$\mathbf{v} = -(\mathbf{W}+\mathbf{1})\,\mathbf{x}$

$\Theta(\bullet)$ is an odd function

Assuming: $\mathbf{W}^2 = -\mathbf{1}$, $\mathbf{W}^T\mathbf{W} = \mathbf{1}$ i.e. $\mathbf{W}$- skew-symmetric, orthogonal e.g. $\mathbf{W} = \mathbf{H}_{2^k}$ Hurwitz-Radon matrix, Eq.(21).

Then: $\mathbf{u}_i$, $\mathbf{v}$ – Haar spectrum of input $\mathbf{x}_i$ and $\mathbf{x}$, respectively.

thus, elements of kernel matrix fulfill:

$$K(\mathbf{u}_i,\mathbf{v}_j) = \Theta(\mathbf{u}_i^T\mathbf{v}_j) = \Theta(2\mathbf{x}_i^T\mathbf{W}\mathbf{x}_j)$$

and

$$K(\mathbf{u}_i,\mathbf{v}_j) = -K(\mathbf{v}_j,\mathbf{u}_i) \tag{46}$$

Hence, matrix

$\mathbf{K} = \{K_{ij}\} = \{K(\mathbf{u}_i,\mathbf{v}_j)\}$ is skew-symmetric.

Note that for the structure from Fig.8., the same key design equation (39) is relevant. However, the structure from Fig.8. can be seen as HNN-based dynamically implemented system, as well. Moreover, taking into account the implementation presented in Fig.4. one can formulate the following statement:

<u>Statement 1</u>

Orthogonal filter-based structures of function approximator can be implemented by compatible connections of octonionic modules.

Other important remarks concluding the above described approximation scheme can be formulated as follows:

<u>Statement 2</u>

Due to the skew-symmetry of kernel matrix, the orthogonal filter based approximation scheme can be regarded as a global method. It means that the neighborhood of the training point $\mathbf{x}_i$ is reconstructed by all the other training points. Exceptionally, this global method is completed by a pointwise local one, if the activation function of used perceptrons has a form $\Theta_r(\cdot)$ (Fig.5.).

<u>Statement 3</u>

Orthogonal filter-based approximation scheme can be easy reformulated as a local technique. Indeed, taking into considerations the kernel defined by Eq.(33), where activation function is an even function e.g. Gaussian function:

$$\Theta(p) = \frac{1}{\sqrt{2\pi\sigma}}\, e^{-\frac{p^2}{\sigma^2}} \tag{47}$$

where: $p \in R$

then the kernel matrix

$$\mathbf{K}_S = \{K_{ij}\} = \{K(\mathbf{x}_i,\mathbf{x}_j)\} = \{\Theta(\mathbf{x}_i^T\mathbf{H}_n\mathbf{x}_j)\} \tag{48}$$

is a symmetric, positive matrix.

For $\Theta(p)$ given by Eq.(47) matrix $\{K_{ij}\}$ fulfils:

$K_{ii} > 0$ for all i,
$K_{ii} > K_{ij}$ for $i \neq j$
and there is such a $\sigma > 0$ that $\det \mathbf{K_s} > 0$.
Thus, matrix $\mathbf{K_s}$ is positive definite.
Hence, it is clear that the key design equation is well-posed:

$$\mathbf{c} = \mathbf{K}_s^{-1}\mathbf{y} \tag{49}$$

and the local properties of this approximation scheme can be controlled by parameter σ. It should be however noted that positive definiteness is not necessary for det $\mathbf{K_s} > 0$ and for existing an inverse $\mathbf{K_s}^{-1}$. To summarize this section, let us note that by choosing a different type of activation functions, one generates a family of functions or mappings fulfilling:

$$\mathbf{y}_i = F_q(\mathbf{x}_i)\,,\ i = 1, \dots, m;\ q = 1, 2, \dots$$

To minimize the approximation errors, one should select a function or mapping which, in terms of learning, optimally transforms a neighborhood $S(\mathbf{x}_i)$ of $\mathbf{x}_i$ onto $\mathbf{y}_i$.

5. Modeling classifiers and associative memories

As mentioned in the previous section, an approximation of a mapping can be obtained as an extended structure of a multivariate function approximation. Hence, for the sake of generalization, we below use a notation of mapping approximation.

Define mapping $F: X \rightarrow Y$

where X, Y are input and output training vector spaces, respectively. The values of mapping are known at training points $\left\{\mathbf{x}_i, \mathbf{y}_j\right\}_{i=1}^{m}$ where, dim $\mathbf{x}_i$ = n and dim $\mathbf{y}_i$= l:
Thus:

$$\mathbf{y}_i = F(\mathbf{x}_i) \quad i = 1, \dots, m \tag{50}$$

where: $\mathbf{x}_i \in X, \mathbf{y}_i \in Y$

Classification issues can be seen as a special problem in mapping approximations. If output vectors **y** of mapping $F(\cdot)$ take values from an unordered finite set, then $F(\cdot)$ performs the function of a classifier. In a two-class classification, one class is labeled by y = 1 and the other class by y = -1. The general functionality of classifiers can be then determined by the following equation:

$$F(\tilde{\mathbf{x}}_i) = \mathbf{y}_i\,,\ i = 1, \dots, m \tag{51}$$

where: $\tilde{\mathbf{x}}_i$ denotes a neighborhood of "center" $\mathbf{x}_i$

$\mathbf{y}_i$ – class label

The determination of neighborhoods $\tilde{\mathbf{x}}_i$ depends on the application of a classifier, but generally, to minimize the erroneous classifications, $\tilde{\mathbf{x}}_i$ have to be densely covered by spheres belonging to $\tilde{\mathbf{x}}_i$, $i = 1, \dots, m$. Thus, the problem of classifier design can be formulated as follows:

1. generate a family of mappings $F_q(\cdot)$, q =1, 2, … fulfilling:

$$X \xrightarrow{F_q} Y, \quad F(\tilde{\mathbf{x}}_i) = \mathbf{y}_i \tag{52}$$

   where X, Y are input and output training vector spaces, respectively.
   Members of this family are created by choosing different type of kernels (antisymmetric or symmetric) and different values of regularization parameters γ or σ
2. select the mapping that transforms input points onto output vectors in an optimal way (minimizing approximation errors):

$$\mathbf{x}(\in X) \xrightarrow{F_{opt}} \mathbf{y}(\in Y)$$

   The problem of optimal mapping selection has been recently formulated in the framework of statistical operators on family (52) (e.g. bagging and boosting techniques). We propose here to consider an optimal solution as a superposition of global and local schemes. In the simplest case, we have the following equation:

$$F_{opt}(\bullet) = (1-\alpha)F_G(\bullet) + \alpha F_L(\bullet) \tag{53}$$

   where: weight parameter α; $0 \le \alpha \le 1$.
   and

   $F_G(\bullet)$ - a global model of mapping obtained by using antisymmetric kernels Eq.(33) and Eq.(45)

   $F_L(\bullet)$ - a local model of mapping obtained by using symmetric kernels, Eq.(48).

   The relation (53) is motivated by the general properties of dynamical systems: a vector field $F(\cdot)$ underlying a physical law, object or process generally consists of two components-global and local (recombination and selection in biological systems, respectively).

To illustrate the considerations above, let us consider the following example:

Example1

Let us design a classification of 8-dim. vector input space X, where $\mathbf{x} = [x_1, x_2, \dots, x_8]^T$, $x_k \in [-1, 1]$, k = 1, … , 8. into 2^5 classes centered in randomly chosen points: $\mathbf{x}_i$, i =1, … , 32. This classification has to be error free, with probability 1, for solid spheres $\mathbf{x} \in S_\rho(\mathbf{x}_i)$, where ρ(radius) = 0.2. It has been experimentally found (i.e. by simulation) that covering randomly every sphere $S_\rho(\mathbf{x}_i)$ with 10 balls, such a classifier design can be reformulated as the following mapping approximation (n = 8, m = 320-number of inputs points):

$$F(\mathbf{x}_{ij}) = \mathbf{y}_i, \; i = 1, \dots, 32; \; j = 1, \dots, 10$$

where: $\mathbf{y}_i = [\pm 1, \pm 1, \pm 1, \pm 1, \pm 1]^T$ (binary label of classes)
The set of input points is given by:

$$\{\mathbf{x}_{ij}\} \in S_\rho(\mathbf{x}_i), \; i = 1, \dots, 32, \; j = 1, \dots, 10$$

where: $\rho = 0.2$
To implement the above defined mapping $F(\mathbf{x}_{ij})$, let us choose the antisymmetric kernels Eq.(33), where:

$$\Theta(p) = 5\left(\frac{2}{1+e^{-3p}} - 1\right), \quad p \in R$$

and

$$\mathbf{H}_8 = \frac{1}{\sqrt{7}}\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 0 & -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 0 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 & 0 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 & -1 & 0 & 1 \\ -1 & -1 & 1 & -1 & 1 & 1 & -1 & 0 \end{bmatrix}$$

Some simulation experiments showed that the mapping $F(\mathbf{x}_{ij})$ fulfils formulated constraints on classification for the case: min $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0.7$ (distance between sphere centers) and under condition that regularization parameters $\gamma \geq 0.75$ (Eq.(38)).

Equation (51) can be seen as a definition of associative memory as well, under the assumption that dim $\mathbf{x}_i$ = dim $\mathbf{y}_i$, where $\mathbf{x}_i$ is a memorized pattern. For $\mathbf{y}_i \equiv \mathbf{x}_i$, one gets a feedforward structure of an autoassociative memory, i.e.:

$$F(\tilde{\mathbf{x}}_i) = \mathbf{x}_i,\ i = 1, \ldots, m \tag{54}$$

Hence, the problem of a nonlinear mapping-based design of the associative memory can be regarded as a covering problem of input space X by spheres $S_\rho(\mathbf{x}_i)$.

Moreover, Eq.(54) determines an identity map i.e. :

$$F(\mathbf{x}_i) = \mathbf{x}_i,\ i = 1, \ldots, m \tag{55}$$

and $F(\cdot)$ is an expansion.

Hence, the mapping $F(\cdot)$ possesses at least one fixed point, i.e. :

$$F(\mathbf{e}) = \mathbf{e} \tag{56}$$

where: $\mathbf{e}$- a fixed point of $F(\cdot)$

Specifically, let us construct the family of identity maps for orthogonal vectors $\mathbf{h}_i$, i =1, …, 8, constituting eight columns of matrix $\mathbf{H}_8$ in Eq.(18), i.e.:

$$F_q(\mathbf{h}_i) = \mathbf{h}_i,\ i = 1, \ldots, m;\ q = 1, 2, \ldots \tag{57}$$

using antisymmetric kernels Eq.(33), $\mathbf{h}_i \in R^8$.

It can be shown that in family (57) there are mappings $F_q(\cdot)$ with the number of fixed points $n_e \leq 256$ (e.g. $n_e = 144$), giving rise to a feedback structure of associative memories. Indeed, let us embed such a $F_q(\cdot)$ into a dynamical system, as shown in Fig. 9.

The state-space equation of structure from Fig.9. is given by:

$$\dot{\varsigma} = -\beta\varsigma + F_q(\varsigma) \tag{58}$$

where: ς - 8-dim. state vector, $0 < \beta \leq 1$.

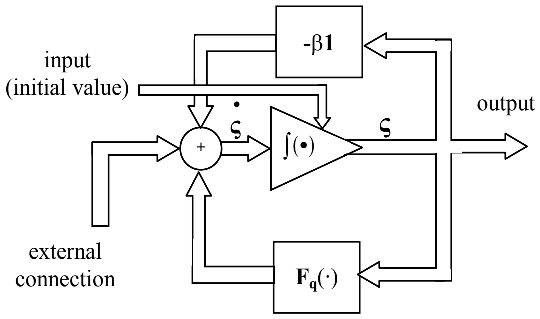


Fig. 9. Dynamical structure of an attractor type associative memory.

Thus, one obtains a feedback type structure of an associative memory with e.g. over 144 asymptotic stable equilibria, but generally with different diameters of attraction basins. Unfortunately the set $\{\mathbf{e}_k\}$of fixed points of a map $F(\cdot)$, can not be found analitically but rather by a method of asymptotic sequences. This can be done relatively simply for 8-dim. identity map presented by Eq.(57) and (58). Thus, due to the exceptional topological properties of a 8-dim vector space, very large scale associative memories could be implemented by a compatible connection of the 8-dim. blocks from Fig.9. An example of such a connection is presented in Fig.10, where two 8-dim. blocks from Fig.9., weakly coupled by parameters $\varepsilon_i > 0$, create a space with a set of equilibria given by:

$$\{\mathbf{e}_c\} = \{\mathbf{e}_k^{(1)}\} \times \{\mathbf{e}_j^{(2)}\} \text{ where: } k = 1, 2, \ldots, 144, \ldots; j = 1, 2, \ldots, 144 \ldots$$

Finally, it is worth noting that the structure from Fig.10 can be scaled up to very large scale memory (by combinatorial diversity), due to its stabilizing type of connections (parameters ε_i). More detailed analysis of the above presented feedback structures is beyond the scope of this chapter.
To summarize, this section points out the main features of orthogonal filter-based mapping approximators:

1. Due to regularization and stability, orthogonal filter-based classifiers can be implemented for any even n (dimension of input vector space) and any $m < \infty$ (number of training vectors). Particularly for $n = 2^k$, $k \geq 3$ such classifiers can be realized by using octonionic modules.
2. As mentioned above, the problem of a nonlinear mapping-based design of classifiers and associative memories can be regarded as a covering problem of input space X by spheres with centers $\mathbf{x}_i$. The radius of the spheres needed to cover X depends on the topology of X and can be changed by a suitably chosen nonlinearity of function $\Theta(\cdot)$. Using, for example, a sigmoidal function for the implementation of $\Theta(\cdot)$., this radius depends on the slope of $\Theta(\cdot)$ at zero. Hence, note that antisymmetric kernels allow us to classify very closely placed input patterns in terms of $\Theta(\cdot) \rightarrow \mathrm{sgn}(\cdot)$.

6. Conclusions

The main issue considered in this chapter is the deterministic learning of mappings. The learning method analysed here relies on multivariate function approximations using mainly skew-symmetric kernels, thus giving rise to very large scale classifiers and associative memories. By using HNN-based orthogonal filters, one obtains regularized and stable structures of networks for learning. Hence, classifiers and memories can be implemented for

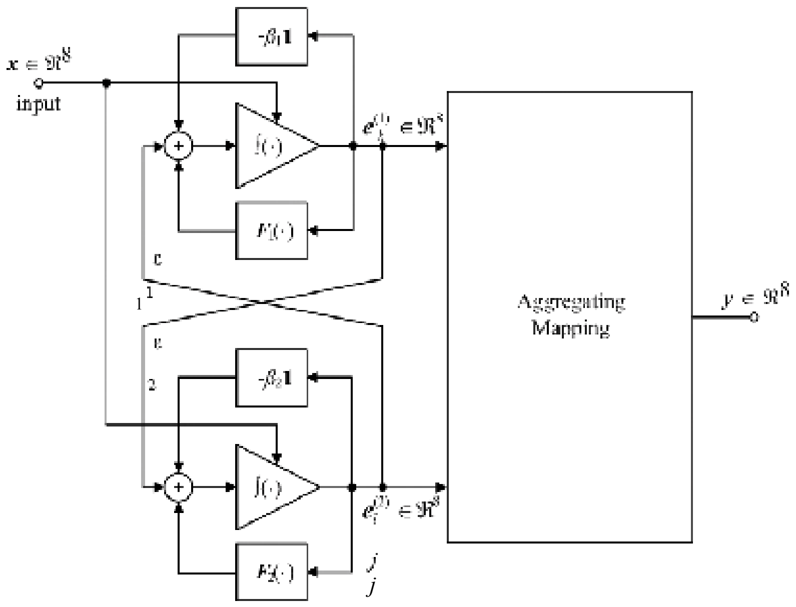


Fig. 10. Very Large Scale Structure of associative memory.

any even n (dimension of input vectors) and any m < ∝ (number of training patterns). Moreover, they can be regarded as numerically well-posed algorithms or physically implementable devices able to perform their functions in real-time. We believe that orthogonal filter-based data processing can be considered as motivated by structures encountered in biological systems.

6. References

Boucheron, S.; Bousquet, O. & Lugosi, G. (2005). Theory of Classification: A survey of some Resent Advances, ESAIM: Probability and Statistics, pp. 323-375.

Eckmann, B. (1999). Topology, Algebra, Analysis-Relations and Missing Links, *Notices of the AMS*, vol. 46, No 5, pp. 520-527.

Evgeniou, T.; Pontil, M. & Poggio, T. (2000). Regularization Networks and Support Vector Machines, In *Advances in Large Margin Classifiers*, Smola, A.; Bartlett, P.; Schoelkopf, G. & Schuurmans, D., (Ed), pp. 171-203, Cambridge, MA, MIT Press.

Poggio, T. & Smale, S. (2003). The Mathematics of Learning. Dealing with Data, *Notices of the AMS*, vol. 50, No 5, pp. 537-544.

Predd, J.; Kulkarni, S. & Poor, H. (2006). Distributed Learning in Wireless Sensor Networks, *IEEE Signal Processing Magazine*, vol. 23, No 4, pp.56-69.

Sienko, W. & Citko, W. (2007). Orthogonal Filter-Based Classifiers and Associative Memories, *Proceedings of International Joint Conference on Neural Networks*, Orlando, USA, pp. 1739-1744.

Sienko, W. & Zamojski, D. (2006). Hamiltonian Neural Networks Based Classifiers and Mappings, *Proceedings of IEEE World Congress on Computational Intelligence*, Vancouver, Canada, pp. 1773-1777.

Vakhania, N. (1993). Orthogonal Random Vectors and the Hurwitz-Radon-Eckmann Theorem, *Proceedings of the Georgian Academy of Sciences, Mathematics*, 1(1), pp. 109-125.

Similarity Discriminant Analysis

Luca Cazzanti
Applied Physics Lab
Box 355640
University of Washington, Seattle, WA 98105,
USA

1. Introduction

This chapter details *similarity discriminant analysis* (SDA), a new framework for similarity-based classification. The two defining characteristics of the SDA classifica- tion framework are *similarity-based* and *generative*. The classifiers in this framework are similarity-based, because they classify based on the pairwise similarities of data samples, and they are generative, because they build class-dependent probability models of the similarities between samples. Similarity-based classifiers already exist; classifiers based on generative models already exist. SDA is a *new* framework for classification comprising classifiers that are *both* similarity-based and generative.

Within the general SDA framework, this chapter describes several families of classifiers: the *SDA classifier*, the *local SDA classifier*, and the *mixture SDA classifier*. The SDA classifier is at the foundation of SDA. It classifies based on the class-conditional generative models of the similarity of the samples to representative class prototypes, or *centroids*. The SDA framework is introduced, developed, and discussed with the aid of this centroid-based SDA classifier. Then, the centroid-based SDA classifier is generalized beyond class centroids to arbitrary class-descriptive statistics. Other possible statistics are described, illustrating the power and generality of the SDA framework.

The local SDA classifier is a local version of the SDA classifier. It builds similarity-based class-conditional generative models within a neighborhood of a test sample to be classified. The local class models are endowed with low bias and retain the powerful quality of interpretability associated with generative probability models. Local SDA is a consistent classifier, in the sense that its error rate converges to the Bayes error rate, which is the best possible error rate attainable by a classifier.

The mixture SDA classifier draws from the well-established metric learning mixture model research. It generalizes the single-centroid SDA classifier to a mixture of single-centroid SDA components. The mixture SDA classifier can be trained with an expectation-maximization (EM) algorithm which parallels the standard EM approach for the well-known Gaussian mixture models.

The problem of classifying samples based only on their pairwise similarities may be divided into two sub-problems: measuring the similarity between samples and classifying the samples based on their pairwise similarities. It is beyond the scope of this chapter to discuss exhaustively and in detail various ways to measure similarity and various similarity-based classifiers. The reader is referred to the references for more details; here, only a brief summary of relevant techniques is provided

1.1 Measuring similarity

Judging similarity between samples characterized by many disparate data types poses challenges of data representation and quantitative comparison. For example, modern databases store information from disparate data sources in different formats: multimedia databases store audio, video and text data; proteomics databases store information on proteins, genetic sequences, and related annotations; internet traffic databases store mouse click histories, user profiles, and marketing rules; homeland security databases may store data on individuals and organizations, annotations from intelligence reports, and maritime shipping records. These database objects, or samples, are described by both numerical and non-numerical data. For example, a security database might store cell phone records in textual form and voice parameters for speaker recognition in numerical form. Representing all these different data types with continuous-valued numbers in a geometric feature space is not appropriate. Thus, current metric space classifiers which rely on metric similarity functions may not be applicable.

Furthermore, in some applications, only the pairwise similarities may be observed, and the underlying features may be inaccessible. For example, one of the datasets discussed in this chapter consists of human-judged similarities between pairs of sonar echoes. For this dataset, the putative perceptual features from which the human similarity ratings are generated are unknown - indeed eliciting the features remains an ongoing research problem (Philips et al., 2006) - but the similarity ratings are nonetheless successfully used for classification. In many applications, the similarity relationship between samples may lack the metric properties usually associated with distance (minimality, symmetry, triangle inequality); thus, using a metric function to express the pairwise similarities is suboptimal. Similarities are more general than distances and require more general functions than metrics (Tversky, 1977). Several researchers have addressed the problem of measuring similarity by rpoposing several simialrity measures. Psychologists, leacd by Tversky, have proposed models of similarity that take into account context and the non-metric way in which humans judge the similarity between complex objects (Tversky, 1977; Tversky & Gati, 1978; Gati & Tversky, 1984; Sattath & Tversky, 1987). The value difference metric (VDM) was originally designed with the goal of improving nearest-neighbor classification (Stanfill & Waltz, 1986) of text documents, and subsequent improvements extended it to classification of objects characterized by both textual and numerical features (Wilson & Martinez, 1997; Cost & Salzberg, 1993). Lin proposed an information-theoretic similarity (Lin, 1998) for document retrieval; (Cazzanti & Gupta, 2006) proposed the *residual entropy similarity* measure by extending Tversky's psychological similarity models with information-theoretic notions, and showed that it strongly takes into account the context in which the similarity is being evaluated. More comprehensive reviews of similarity measures appear in (Santini & Jain, 1999) and (Everitt & Rabe-Hesketh, 1997).

1.2 Similarity-based classifiers

Similarity-based classifiers are defined as those classifiers that require only a pairwise similarity - a description of the samples themselves is not needed. Similarity-based classifiers classify test samples given a labeled set of training samples, the pairwise

similarity values of the training samples, and the similarity of the test sample to the training samples. If the description of the samples in terms of feature vectors is available, an existing or ad hoc similarity function that maps any two samples to a similarity value may be used (Bicego et al., 2006; Pekalska et al., 2001; Jacobs et al., 2000; Hochreiter & Obermayer, 2006).
Among the existing similarity-based classifiers, the simplest method is the nearest neighbor classifier, which determines the most similar training sample z to the test sample x, and classifies x as z's class:

$$\hat{y} = \arg \max_{h=1,\dots,G} \left(\max_{z \in \mathcal{X}_h} s(x,z) \right), \tag{1}$$

where $\mathcal{X}_h$ is the set of training samples from class h. More generally, the k-nearest neighbor classifier (k-NN) determines a neighborhood of k most similar training samples to the test sample x, and classifies x as the most-frequently occurring class label among the neighbors. Experiments have shown that nearest neighbors can perform well on practical similarity-based classification tasks (Cost & Salzberg, 1993; Pekalska et al., 2001; Simard et al., 1993; Belongie et al., 2002). For example, nearest neighbor classifiers using a tangent distortion metric and a shape similarity metric have both been shown to achieve very low error on the MNIST character recognition task.
Condensed near-neighbor strategies replace the set of training samples for each class with a set of prototypes for that class. Usually the prototype set is an edited set of the original training samples (also called edited nearest neighbors), but the prototypes do not need to be from the original training set. Let c_h be the number of the prototypes $\{\mu_{hl}\}$ for class h; then, the condensed nearest neighbor rule is to classify a test sample x as the class of the prototype to which it is most similar,

$$\hat{y} = \arg \max_{h=1,\dots,G} \left(\max_{l=1,\dots,c_h} s(x,\mu_{hl}) \right) \tag{2}$$

Many authors have considered strategies for condensing near-neighbors for similarity-based classification to increase classification speed, decrease the required memory, remove outliers, and possibly attain better performance (Weinshall et al., 1999; Jacobs et al., 2000; Lam et al., 2002; Pekalska et al., 2006; Lozano et al., 2006). A well-known strategy for condensing nearest neighbors in non-metric spaces is the *k-medoids* algorithm (Hastie et al., 2001). Given a set of c_h candidate prototypes selected from $\mathcal{X}_h$, the remaining training samples $z \in \mathcal{X}_h$ are assigned to their nearest (most similar) prototype, so that the set $\mathcal{X}_h$ of all training samples from class h is partitioned in c_h mutually-exclusive subsets $\{\mathcal{X}_{hl}\}$, and each $\mathcal{X}_{hl}$ is uniquely associated with candidate prototype μ_{hl}. Then, the lth prototype for the hth class is updated according to the standard maximum similarity update rule, which selects the new μ_{hl} as the training sample in $\mathcal{X}_{hl}$ which is most similar to all other samples in $\mathcal{X}_{hl}$,

$$\mu_{hl}^{*} = \arg \max_{\mu_{hl} \in \mathcal{X}_{hl}} \sum_{z \in \mathcal{X}_{hl}} s(z,\mu_{hl}). \tag{3}$$

The training samples are then reassigned to the updated prototypes, and the update rule (3) is repeated. The reassignment and update steps are repeated until a predetermined maximum number of iterations is reached or until the updated prototypes $\mu^*_{hl} = \mu_{hl}$ for all h and l. The number of prototypes in each class c_h is determined by cross-validation; the initial prototypes $\{\mu_{hl}\}$ are selected randomly from the training set.

An extreme form of condensed near-neighbors is to replace each class's training samples by one prototypical sample, often called a *centroid*. The resulting nearest centroid classifier can be considered a simple parametric model (Weinshall et al., 1999), though it lacks a probabilistic structure. Let $s(x, z)$ be the similarity between a sample x and a sample z, and let there be a finite set of classes 1, 2, ... ,G. The nearest centroid approach classifies x as the class

$$\hat{y} = \arg \max_{h=1,\ldots,G} s(x, \mu_h), \tag{4}$$

where μ_h is the representative centroid for the class h. A standard definition for the centroid of a set of training samples is the training sample that has the maximum total similarity to all the training samples of the same class (Weinshall et al., 1999; Jacobs et al., 2000):

$$\mu_h = \arg \max_{\mu \in \mathcal{X}_h} \sum_{z \in \mathcal{X}_h} s(z, \mu). \tag{5}$$

A variation of the nearest centroid classifier is the local nearest centroid classifier, which is an analog to the local nearest means classifier proposed by Mitani and Hamamoto (Mitani & Hamamoto, 2006, 2000). In this variant, the class centroids (5) are computed from a local neighborhood of each test point x; they are not computed from the entire training set. The neighborhood may be defined in many ways. The most common definition is the k-nearest neighbors. In this case, local nearest centroid is like the k-NN classifier, except that it classifies x as the class of its nearest centroid where the centroids are computed from the k-nearest neighbors of x.

The nearest centroid classifier is analogous to the nearest-mean classifier in Euclidean space, which is the optimal Euclidean-based classifier if one assumes that the class-conditional distributions are Gaussian, the class priors are equal, and that each class covariance is the identity matrix (Duda et al., 2001; Hastie et al., 2001).

2. Similarity discriminant analysis

In standard metric learning, quadratic discriminant analysis (QDA) is a generative classifier that generalizes the nearest-mean classifier by modeling each class-conditional distribution as a Gaussian (Duda et al., 2001). Analogously, SDA is a generative similarity-based classifier that generalizes the nearest-centroid classifier (Weinshall et al., 1999) by modeling each class-conditional distribution with a parametric probability model (Cazzanti et al.; Gupta et al., 2007). The SDA class-conditional probability models have exponential form, because they are derived as the maximum entropy distributions subject to constraints on the mean similarities of the data to the class centroids. As with other parametric approaches to classification, the resulting log-linear SDA classifier is powerful when it effectively models the true generating distribution. This section introduces SDA and shows how it classifies; then, it extends SDA from using class centroids to using arbitrary descriptive statistics to discriminate between the classes, including continuous-valued statistics.

2.1 A generative centroid-based classifier

Assume a class centroid μ_h has been determined for the hth class, where $h = 1, ..., G$. A problem with the nearest centroid classifier given in (4) is that it does not take into account the variability of the similarities to the centroid within a class. To take into account this variability, first consider a simple generalization of nearest centroid, here called the *adjusted nearest centroid classifier* : classify a test sample x as class $\hat{y}$ where

$$\hat{y} = \arg \max_{h=1,\ldots,G} \frac{s(x, \mu_h)}{\bar{s}_{hh}}, \tag{6}$$

and where $\bar{s}_{hh}$ is the average similarity of class h samples to the class h centroid,

$$\bar{s}_{hh} = \frac{1}{n_h} \sum_{z \in \mathcal{X}_h} s(z, \mu_h),$$

where $n_h = |\mathcal{X}_h|$. The adjusted nearest centroid classifier is analogous to the one-dimensional Gaussian rule of classifying based on the the variance-weighted distances to the class means, $\|x- \tilde{\mu}_h\| / \tilde{\sigma}_h$, where $x, \tilde{\mu}_h, \tilde{\sigma}_h \in \mathbb{R}$. The adjusted nearest centroid classifier is more flexible than the nearest centroid classifier, but lacks a probabilistic structure, and takes into account only the similarity of a sample to one class centroid.

Thus, a generative centroid-based classifier that models the probability distribution of the test sample similarity statistics $s(x, \mu_h)$ for each h is proposed. Begin with the Bayes classifier (Hastie et al., 2001), which assigns a test sample x the class $\hat{y}$ that minimizes the expected misclassification cost,

$$\hat{y} = \arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f, g) P(Y = g|x), \tag{7}$$

where $C(f, g)$ is the cost of classifying the test sample x as class f if the true class is g and $P(g|x)$ is the probability that sample x belongs in class g. In practice the distribution $P(g|x)$ is generally unknown, and thus the Bayes classifier of (7) is an unattainable ideal.

Assume that all test and training samples come from some abstract space of samples $\mathcal{B}$, which might be an ill-defined space, such as $\mathcal{B}$ is the set of all amino acids, or $\mathcal{B}$ is the set of all terrorist events, or $\mathcal{B}$ is the set of all women who gave birth to twins. Let $x, \mu_h, z \in \mathcal{B}$, and let the similarity function be some function $s : \mathcal{B} \times \mathcal{B} \to \Omega$, where $\Omega \subset \mathbb{R}$. If the set of possible samples $\mathcal{B}$ is finite, then the space of the pairwise similarities Ω will also be finite, and hence discrete. For simplicity, in this section assume that Ω is a finite discrete space. Continuous and possibly infinite spaces B, Ω are briefly discussed in Section 2.2.3.

Consider a random test sample X with random class label Y, where x will denote a realization of X. Assume that the relevant information about X's class label is captured by the set $\mathcal{T}(X)$ of G descriptive statistics

$$\mathcal{T}(X) = \{s(X, \mu_1), s(X, \mu_2), \ldots, s(X, \mu_G)\}.$$

That is, the relevant information about x is captured by its similarity to each class centroid. Under this assumption, given a particular test sample x, the classification rule (7) becomes: classify x as class $\hat{y}$ that solves

$$\arg\min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g)P(Y=g|\mathcal{T}(x)).$$

Using Bayes rule, this is equivalent to the problem

$$\arg\min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g)P(\mathcal{T}(x)|Y=g)P(Y=g). \tag{8}$$

Note that $P(\mathcal{T}(x) \mid Y = g)$ is the probability of seeing a particular set of similarities between the test sample x and the G class centroids $\{\mu_1, \mu_2, \ldots, \mu_G\}$ given that x is a class g sample.
Next, assume that each unknown class-conditional distribution $P(\mathcal{T}(x) \mid Y = g)$ has the same average value as the training sample data from class g. That is, given a random test sample X there will be a random similarity $s(X, \mu_h)$; constrain the class-conditional distribution $P(\mathcal{T}(x) \mid Y = g)$ such that

$$E_{P(\mathcal{T}(x)|Y=g)}[s(X,\mu_h)] = \frac{1}{n_g}\sum_{z\in\mathcal{X}_g} s(z,\mu_h), \tag{9}$$

holds for each g and h where n_g is the number of training samples of class g. Each constraint requires that the class-conditional expectation of one of the elements of $\mathcal{T}(X)$ is equal to the maximum likelihood estimate of that element given the training data. This makes for G constraints for each class-conditional distribution, for a total of $G\times G$ constraints because there are G class-conditional distributions. Given these constraints, there is some compact and convex feasible set of class-conditional distributions. A feasible solution will always exist because the constraints are based on the data.
As prescribed by Jaynes' principle of maximum entropy (Jaynes, 1982), a unique class-conditional joint distribution is selected by choosing the maximum entropy solution that satisfies (9). Maximum entropy distributions have the maximum possible uncertainty, such that they are as uniform as possible while still satisfying given constraints. Given a set of moment constraints, the maximum entropy solution is known to have exponential form (Cover & Thomas, 1991). For example, in standard metric learning, the Gaussian class-conditional distribution model used in LDA and QDA is the maximum entropy distribution given a specific mean vector and covariance matrix (Cover & Thomas, 1991).
The maximum entropy distribution that satisfies the moment constraints specified in (9) is

$$\hat{P}(\mathcal{T}(x)|Y=g) = \gamma_g e^{\left(\sum_{h=1}^{G}\lambda_{gh}s(x,\mu_h)\right)}, \tag{10}$$

where $\{\gamma_g, \lambda_{g1}, \lambda_{g2}, \ldots, \lambda_{gG}\}$ are a unique set that ensures that the constraints (9) are satisfied and that $\hat{P}$ ($\mathcal{T}$(x) | Y = g) is non-negative and normalized. Rewrite equation (10) as

$$\hat{P}(\mathcal{T}(x)|Y=g) = \prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh}s(x,\mu_h)} \tag{11}$$

where $\prod_h \gamma_{gh} = \gamma_g$. Let

$$\hat{P}(s(x,\mu_h)|Y=g) = \gamma_{gh} e^{\lambda_{gh} s(x,\mu_h)};$$

then (11) can be written

$$\hat{P}(\mathcal{T}(x)|Y=g) = \prod_{h=1}^{G} \hat{P}(s(x,\mu_h)|Y=g).$$

That is, under the maximum entropy assumption, the joint distribution on $\mathcal{T}(x)$ is the product of the marginal distributions on each similarity statistic comprising the set $\mathcal{T}(X)$. Thus, the similarity statistics are conditionally independent given the class label under this model. Although one does not expect this conditional independence to be strictly valid, the hypothesis is that it will be an effective model, just as the naive Bayes' model that features are independent is optimistic but useful.
Substituting the maximum entropy solution (10) into (8) yields the classification rule: classify x as the class $\hat{y}$ which solves

$$\arg\min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g) \left(\prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh} s(x,\mu_h)} \right) P(Y=g). \tag{12}$$

To solve for the parameters $\{\lambda_{gh}, \gamma_{gh}\}$, one solves the G constraints individually for λ_{gh}. Then given $\{\lambda_{gh}\}$, the $\{\gamma_{gh}\}$ are trivially found using the normalization constraint. Solving for λ_{gh} is straightforward; for example, one uses the Nelder-Mead optimizer built into Matlab (version 15) in the `fminsearch()` function (Mat). This is the method used throughout this work. As an alternative, one may find the probability mass function with maximum entropy, subject to the constraints, without a priori knowledge that the solution is exponential.
The classifier given in (12) is termed the *similarity discriminant analysis* (SDA).

2.2 General generative models for similarity-based classification

The previous section introduced SDA for the case when the descriptive statistics are the similarities of the samples to the class centroids. This section generalizes SDA to arbitrary descriptive statistics $\mathcal{T}(x)$ which can be used to discriminate different classes and describes the resulting general generative model for classifying with arbitrary statistics.

2.2.1 Descriptive statistics

Several possibilities for the descriptive statistics $\mathcal{T}(x)$ are described below.

- Centroid Definitions - A standard centroid definition was given in (5). Another choice is to allow a class prototype that is not constrained to be a training sample,

$$\mu_h^* = \arg\max_{\mu \in \mathcal{B}} \sum_{z \in \mathcal{X}_h} s(z,\mu). \tag{13}$$

In this case the solution μ_h^* requires a description of the entire space of possible samples $\mathcal{B}$. In practice, one may not know the entire sample space $\mathcal{B}$, only the training samples $\mathcal{X}$, so it may not be possible to calculate μ_h^*.

A third definition of a class prototype is based on Tversky's analysis of similarity-based near-neighbor relationships (Tversky & Hutchinson, 1986; Schwartz & Tversky, 1980), and takes into account the similarity-based ranks of a training sample's near-neighbors. Define the neighborhood $\mathcal{N}(z) \subseteq \mathcal{X}$ of a sample z as the set of training samples whose nearest neighbor in similarity space is z. The popularity of z is the size of its neighborhood $|\mathcal{N}(z)|$. The class centroid is the sample with the highest popularity, that is,

$$\mu_h = \arg\max_{z \in \mathcal{X}_h} |\mathcal{N}(z)|. \tag{14}$$

This centroid is the training sample that is most often the closest neighbor of the training samples in the class. Ties in popularity are broken by selecting the sample with the highest total similarity to its neighbors.

- Higher Order and Non-Centroidal Descriptive Statistics - Given a set of class centroids $\{\mu_h\}$, higher-order statistics could be used as, or added to, the set of descriptive statistics $\mathcal{T}(X)$, such as $(s(X, \mu_h) - E[s(X, \mu_h)])^2$, or cross-class statistics, such as $(s(X, \mu_h) - E[s(X, \mu_g)])^2$. Or, instead of the centroid-based statistics fs(X, μ_h)g, it might be more appropriate to use the nonparametric statistics formed by the total pairwise similarity for each class h, such that the hth descriptive statistic in test set $\mathcal{T}(X)$ is $\sum_{z \in \mathcal{X}_h} s(X, z)$.
- Nearest Neighbor Similarity - A descriptive statistic that is not centroid-based is the *nearest neighbor similarity*: a test sample's similarity to its most similar training sample. Given a sample x and the training samples $z \in \mathcal{X}$, the nearest neighbor similarity is defined

$$s_{nn}(x) = \max_{z \in \mathcal{X}} s(x, z). \tag{15}$$

The SDA classifier based on nearest neighbor similarity, denoted by *nnSDA*, may be viewed as a generalization of the similarity-based nearest neighbor classifier (1-NN) defined in 1. That classifier labels x with the same class label as its nearest neighbor without making use of any information about its similarity to such nearest neighbor. The nnSDA classifier, on the other hand, classifies x as the class of its nearest neighbor based on a probabilistic model of $s_{nn}(x)$. The probability model is computed with the mean-constrained maximum entropy approach of Section 2.1, which results in exponential solutions. In this case, the constraint is that the mean of the distribution must be the same as the empirical average of the observed nearest neighbor similarities. Denote by $s_{nn,h}(X)$ the random similarity of a random test sample X to its nearest neighbor in class h. For nnSDA, the constraint is written as

$$E_{P(\mathcal{T}(x)|Y=g)}[s_{nn,h}(X)] = \frac{1}{n_g} \sum_{z \in \mathcal{X}_g} s_{nn,h}(z), \tag{16}$$

and the classification rule becomes to classify as the class $\hat{y}$ that solves

$$\arg\min_{f=1,\dots,G} \sum_{g=1}^{G} C(f,g) \left(\prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh} s_{nn,h}(x)} \right) P(Y=g), \tag{17}$$

where the parameters λ_{gh} and γ_{gh} are computed with the same numerical optimization method used for SDA.

As further discussed in the next section, the SDA framework accommodates any desired set of descriptive statistics $\mathcal{T}(x)$: different similarity functions could be mixed, dissimilarities and similarities can be mixed, and so on.

2.2.2 Generative classifier from arbitrary descriptive statistics

Given an arbitrary set of M descriptive statistics $\mathcal{T}(x)$, the same reasoning of Section 2.1 produces a generative similarity-based classifier. First, the assumption is that $\mathcal{T}(x)$ is sufficient information to classify x leads to the classification rule given in (8). Second, for the mth descriptive statistic $T_m(x) \in \mathcal{T}(x)$, $m = 1, \ldots, M$, one assumes that its mean with respect to the class conditional distribution of $\mathcal{T}(x)$ is equal to the training sample mean:

$$E_{P(\mathcal{T}(x)|g)}[T_m(X)] = \frac{1}{n_g} \sum_{z \in \mathcal{X}_g} T_m(z). \tag{18}$$

Third, given the $M \times G$ constraints specified by (18), one estimates the class-conditional distribution to be the maximum entropy distribution,

$$\begin{aligned} \hat{P}(\mathcal{T}(x)|g) &= \prod_{m=1}^{M} \gamma_{gm} e^{\lambda_{gm} T_m(x)} \\ &= \prod_{m=1}^{M} \hat{P}(T_m(x)|g). \end{aligned} \tag{19}$$

Substituting the maximum entropy solution (19) into (8) yields the SDA classification rule: classify x as the class $\hat{y}$ which solves

$$\arg \min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g) P(g) \prod_{m=1}^{M} \gamma_{gm} e^{\lambda_{gm} T_m(x)} . \tag{20}$$

The parameters $\{\lambda_{gm}, \gamma_{gm}\}$ are calculated as in the centroid-based SDA case described in Section 2.1.

2.2.3 Continuous-valued statistics

The generative classification models presented in this chapter can be extended to the case in which the statistics $\mathcal{T}(x)$ are from a continuous set Ω. This will be the case, for example, when using an overlap similarity (e.g. $\max\{x[i], z[i]\}$) with real-valued features, or when the similarity between X and z is the Euclidean distance. Then, the expectation in (18) is a normalized integral over the continuous set of possible similarity values. Let a and b denote the minimum and maximum possible similarity values (and hence the lower and upper bound on the expectation's integral). Then simplifying (18) yields the relationship

$$\frac{e^{\lambda_{gm} b}(\lambda_{gm} b - 1) - e^{\lambda_{gm} a}(\lambda_{gm} a - 1)}{\lambda_{gm}(e^{\lambda_{gm} b} - e^{\lambda_{gm} a})} = \bar{t}_{gm}, \tag{21}$$

where $\bar{t}_{gm} = \frac{1}{n_g} \sum_{z \in \mathcal{X}_g} T_m(z)$. The solution to (21) can be computed numerically. For the special case $a = 0$ and $b = \infty$, the solution is $\lambda_{gm} = -1/\bar{t}_{gm}$.

3. Local SDA

This chapter introduces *local SDA* (Cazzanti & Gupta, 2007), a similarity-based classifier that is both generative and local. An advantage of generative classifiers is their interpretability: classes are modeled by conditional probability distributions which are assumed to have generated the observed data. An advantage of local classifiers it that they reduce the estimation bias problem which affects generative classifiers. Local SDA combines the qualities of both generative and local classifiers.

For the SDA classifier, the class-conditional generative distributions are exponentials that model the similarities between samples - or more generally the descriptive statistics of the sample. The exponentials are the maximum entropy distributions subject to constraints on the mean values of the similarities. However, when the underlying distributions are complex, a particular set of empirical statistics may fail to capture the necessary information about a sample's class membership. In fact, in SDA, constraining the means of the class-conditional distributions may result in too much model bias, just as the QDA model of one Gaussian per class causes model bias (Hastie et al., 2001). In standard metric learning, one way to address the bias problem while retaining the advantages of a generative approach is to form more flexible Gaussian mixture models. In similarity-based learning, mixture models may also be formed; this approach is discussed in Section 4.

Here, the bias in SDA is addressed by using local classifiers in similarity space. In metric learning, one way to avoid the bias problem is to use local classifiers, e.g. k-NN, which classify test samples based on the class labels of their nearest neighbors. Local classifiers do not estimate probabilistic models for the sample classes and consequently lack the interpretability of generative models. Even so, they provide an intuitive framework for classification through the concepts of nearest-neighbor and neighborhood. In this chapter, SDA is applied to a local neighborhood about the test sample. The resulting *local SDA* classifier trades-off model bias and estimation variance depending on the neighborhood size, while retaining the power of a generative classifier. To the author's knowledge, local SDA is the first example of a classifier that is both generative and local. The only arguable contender is the local nearest- mean classifier (Mitani & Hamamoto, 2000, 2006) for metric learning; however that classifier was not proposed as a generative model.

Local SDA is a straightforward variation of SDA. The local SDA classifier model is that all of the relevant information about classifying a test sample x depends only on the k nearest (most similar) training samples to x. Thus, the local SDA classifier computes the descriptive statistics from a neighborhood of a test sample. More specifically, local SDA is a log-linear generative classifier that models the probability distribution of the similarity $s(x, \mu_h)$ between the test sample x and the class centroids $\{\mu_h\}$, just like SDA. Unlike SDA, the class centroids, the class-conditional similarity probability models, and the estimates of the class priors are computed from a neighborhood of the test sample rather than from the entire training set. Thus, the class centroid definition (5) used for SDA still holds for local SDA; one simply redefines $\mathcal{X}_h$ as the subset of the k nearest neighbors from class h. The class priors are estimated using normalized class membership counts of the neighbors of x, that is $\hat{P}(Y = h) = |\mathcal{X}_h|/k$. The mean similarity constraints (9) for the SDA maximum entropy optimization are formally the same for local SDA, except that the mean is computed from the neighbors of test sample x rather than the whole training set. Thus, the optimized parameters λ_{gh} and γ_{gh} are local. Given the set of local class centroids $\{\mu_h\}$, the local class priors $\hat{P}(Y = g)$, and the local class-conditional model parameters γ_{gh} the local SDA classification rule is identical to the SDA rule (12):

$$\arg\max_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g) \left(\prod_{h=1}^{G} \gamma_{gh} e^{\lambda_{gh} s(x,\mu_h)} \right) \hat{P}(Y = g).$$

A problem can occur if the hth class has few training samples in the neighborhood of test sample x. In this case, the local SDA model for class h is difficult to estimate. To avoid this problem, if the number of local training samples in any of the classes is very small, for example $n_h < 3$, the local SDA classifier reverts to the local nearest centroid classifier. If $n_h = 0$ so that $\mathcal{X}_h$ is the empty set, then the probability of class h is locally zero, and that class is not considered in the classification rule (12). This strategy enables local SDA to gracefully handle small k and very small class priors.

Local classification algorithms have traditionally been weighted voting methods, including classifying with local linear regression, which can be formulated as a weighted voting method (Hastie et al., 2001). These methods are by their nature non-parametric and their use arises in situations when the available training samples are too few to accurately build class models. On the other hand, it is known that the number of training samples required by nonparametric classifiers to achieve low error rates grows exponentially with the number of features (Mitani & Hamamoto, 2006). Thus, when only small training sets are available, nonparametric classifiers are negatively impacted by outliers. In 2000, Mitani and Hamamoto (Mitani & Hamamoto, 2000, 2006) were the first ones to propose a classifier that is both model-based and local. However, they did not develop it as a local generative method; instead, they proposed the classifier as a local weighted-distance method. Their nearest-means classifier can be interpreted as a local QDA classifier with identity covariances. In experiments with simulated and real data sets, the local nearest-means classifier was competitive with, and often better than, nearest neighbor, the Parzen classifier, and an artificial neural network, especially for small training sets and for high dimensional problems.

Local nearest-means differs from local SDA in several aspects. First, the classifier by Mitani and Hamamoto in (Mitani & Hamamoto, 2006) learns a metric problem, not a similarity problem: the class prototypes are the local class-conditional means of the features and a weighted Euclidean distance is used to classify a test sample as the class of its nearest class mean. Second, the neighborhood definition is different than the usual k nearest neighbors: they select k nearest neighbors from each class, so that the total neighborhood size is $k \times G$.

More recently, it was proposed to apply a support vector machine to the k nearest neighbors of the test sample (Zhang et al., 2006). The SVM-KNN method was developed to address the robustness and dimensionality concerns that a²ict nearest neighbors and SVMs. Similarly to the nearest-means classifier, the SVM-KNN is a hybrid local and global classifier developed to mitigate the high variance typical of nearest neighbor methods and the curse-of-dimensionality. However, unlike the nearest means classifier of Mitani and Hamamoto, which is rooted in Euclidean space, the SVM-KNN can be used with any similarity function, as it assumes that the class information about the samples is captured by their pairwise similarities without reference to the underlying feature space. Experiments on benchmark datasets using various similarity functions showed that SVM-KNN outperforms k-NN and its variants especially for cases with small training sets and large number of classes. SVM-KNN differs from local SDA because it is not a generative classifier.

Finally, note that different definitions of neighborhood may be used with local SDA. One could use the Mitani and Hamamoto (Mitani & Hamamoto, 2006) definition described above, or radius-based definitions. For example, the neighborhood of a test sample x may be defined as all the samples that fall within a factor of $1+\alpha$ of its similarity to its most similar neighbor, and α is cross-validated. This work employs the traditional definition of neighborhood, as the k nearest neighbors.

3.1 Consistency of the local SDA classifier

Generative classifiers with a finite number of model parameters, such as QDA or SDA, will not asymptotically converge to the Bayes classifier due to the model bias. This section shows that, like k-NN, the local SDA classifier is consistent such that its expected classification error $E[L]$ converges to the Bayes error rate L^* under the usual asymptotic assumptions that the number of training samples $N \to \infty$, the neighborhood size $k \to \infty$, but that the neighborhood size grows relatively slowly such that $k=N \to 0$. First a lemma is proven that will be used in the proof of the local SDA consistency theorem. Also, the known result that k-NN is a consistent classifier is reviewed in terms of similarity.

Let the similarity function be $s : \mathcal{B} \times \mathcal{B} \to \Omega$, where $\Omega \subset \mathbb{R}$ is discrete and let the largest element of $-\Omega$ be termed s_{max}. Let X be a test sample and let the training samples $\{X_1,X_2, ... ,X_N\}$ be drawn identically and independently. Re-order the training samples according to decreasing similarity and label them $\{Z_1,Z_2, ..., Z_N\}$ such that Z_k is the kth most similar neighbor of X.

Lemma 1 *Suppose $s(x,Z) = s_{max}$ if and only if $x = Z$ and $P(s(x,Z) = s_{max}) > 0$ where Z is a random training sample. Then $P(s(x,Z_k) = s_{max}) \to 1$ as $k, N \to\infty$ and $k/N \to 0$.*

Proof: The proof is by contradiction and is similar to the proof of Lemma 5.1 in (Devroye et al., 1996). Note that $s(x,Z_k) \neq s_{max}$ if and only if

$$\frac{1}{N}\sum_{i=1}^{N} I_{\{s(x,Z_i)=s_{max}\}} < \frac{k}{N}, \tag{22}$$

because if there are less than k training samples whose similarity to x is s_{max}, the similarity of the kth training sample to x cannot be s_{max}. The left-hand side of (22) converges to $P(s(x,Z) = s_{max})$ as $N\to\infty$ with probability one by the strong law of large numbers, and by assumption $P(s(x,Z) = s_{max}) > 0$. However, the right-hand side of (22) converges to 0 by assumption. Thus, assuming $s(x,Z_k) \neq s_{max}$ leads to a contradiction in the limit. Therefore, it must be that $s(x,Z_k) = s_{max}$.

Theorem 1 *Assume the conditions of Lemma 1. Define L to be the probability of error for test sample X given the training sample and label pairs $\{(Z_1, Y_1), (Z_2, Y_2), ... , (Z_N, Y_N)\}$, and let L^* be the Bayes error.* If *$k,N \to \infty$ and $k/N \to 0$, then for the local SDA classifier $E[L] \to L^*$.*

Proof: By Lemma 1, $s(x,Z_i) = s_{max}$ for $i \leq k$ in the limit as $N \to \infty$, and thus in the limit the centroid μ_h of the subset of the k neighbors that are from class h must satisfy $s(x, \mu_h) = s_{max}$, for every class h which is represented by at least one sample in the k neighbors. By definition of the local SDA algorithm, any class $\bar{h}$ that does not have at least one sample in the k neighbors is assigned the class prior probability $P(Y = \bar{h}) = 0$, so it is effectively eliminated from the possible classification outcomes. Then, the constraint (9) on the expected value of the class-conditional similarity for every class g that is represented in the k neighbors of x is

$$E_{P(s(x,\mu_h)|Y=g)}[s(X, \mu_h)] = s_{max}, \tag{23}$$

which is solved by the pmf $P(s(x, \mu_h) \mid Y = g) = 1$ if $s(x, \mu_h) = s_{max}$, and zero otherwise. Thus the local SDA classifier (12) becomes

$$\hat{y} = \arg \max_{g=1,\dots G} \hat{P}(Y = g), \tag{24}$$

where the estimated probability of each class $\hat{P}(Y = g)$ is calculated using a maximum likelihood estimate of the class probabilities for the neighborhood. Then, $\hat{P}(Y = g) \to P(Y = g \mid x)$ as $k \to \infty$ with probability one by the strong law of large numbers. Thus the local SDA classifier converges to the Bayes classifier, and the local SDA average error $E[L] \to L^*$.
The known result that k-NN is a consistent classifier can be stated in terms of similarity as a direct consequence of Lemma 1:
Lemma 2 *Assume the conditions of Lemma 1 and define L and L* as in Theorem 1. For the similarity-based k-NN classifier* $E[L] \to L^*$.
Proof. It follows directly from Lemma 1 that within the size-k neighborhood of x, $Z_i = x$ for $i \leq k$. Thus, the k-NN classifier (1) estimates the most frequent class among the k samples maximally similar to x:

$$\begin{aligned} \hat{y} &= \arg \max_{g=1,\dots,G} \sum_{i=1}^{k} I(Y_i = g) \\ &= \hat{P}(Y = g). \end{aligned}$$

The summation converges to the class prior $P(Y = g \to x)$ as $k \to \infty$ with probability one by the strong law of large numbers, and the k-NN classifier becomes that in (24). Thus the similarity-based k-NN classifier is consistent.

4. Mixture SDA

Like LDA and QDA, basic SDA may be too biased if the similarity space - or more generally the descriptive statistics space - is multi-modal. In analogy to metric space mixture models, the bias problem in similarity space may be alleviated by generalizing the SDA formulation with similarity-based mixture models. In the *mixture SDA* models, the class-conditional probability distribution of the descriptive statistics $\mathcal{T}(x)$ for a test sample x is modeled as a weighted sum of exponential components. Generalizing the single centroid-based SDA classifier and drawing from the metric mixture models (Duda et al., 2001; Hastie et al., 2001), each class h is characterized by c_h centroids $\{\mu_{hl}\}$. The descriptive statistics for test sample x are its similarities to the centroids of class h, $\{s(x, \mu_{h1}), s(x, \mu_{h2}), \dots, s(x, \mu_{hc_h})\}$, for each class h. The mixture SDA model for the probability of the similarities, assuming that test sample x is drawn from class g, is written as

$$P(s(x,\mu_{h1}), s(x,\mu_{h2}), \ldots, s(x,\mu_{hc_h})|Y=g) = \sum_{l=1}^{c_h} w_{ghl}\gamma_{ghl}e^{\lambda_{ghl}s(x,\mu_{hl})}, \tag{25}$$

where $\sum_{l=1}^{c_h} w_{ghl} = 1$ and $w_{ghl} > 0$. Then, the SDA classification rule (12) for mixture SDA becomes to classify x as the class $\hat{y}$ that solves the maximum a posteriori problem

$$\arg\max_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g)\left(\prod_{h=1}^{G}\sum_{l=1}^{c_h} w_{ghl}\gamma_{ghl}e^{\lambda_{ghl}s(x,\mu_{hl})}\right)P(Y=g). \tag{26}$$

Note how the mixture SDA generative model (25) parallels the metric mixture formulation of Gaussian mixture models (GMMs), with the exponentials $\gamma_{ghl}e^{\lambda_{ghl}s(x,\mu_{hl})}$ in place of the Gaussian components. However, there are deep differences between mixture SDA and metric mixture models. In metric learning, the mixtures model the underlying generative probability distributions of the features. Due to the curse of dimensionality, high-dimensional, multi-modal feature spaces require many training samples for robust model parameter estimation. For example, for d features, GMMs require that a $d \times 1$ mean vector and a $d \times d$ covariance matrix be estimated for each component in each class, for a total of $c_h \times (d^2 + 3d)/2$ parameters per mixture. Constraining each Gaussian covariance to be diagonal, at the cost of an increased number of mixture components, alleviates the robust estimation problem, but does not solve it (Reynolds & Rose, 1995).

When relatively few training samples are available, robust parameter estimation becomes particularly di±cult. In similarity-based learning the modeled quantity is the similarity of a sample to a class centroid. The estimation problem is essentially univariate and reduces to estimating the exponent λ_{ghl} in each component of the mixture, for a total of $c_h \times G \times 2$ parameters per mixture (the scaling parameter γ_{ghl} follows trivially). This simpler classifier architecture allows robust parameter estimation from smaller training set depending on the number of centroids per class, or, more generally, the number of descriptive statistics.

Another major difference between mixture SDA and metric mixture models is in the number of class-conditional probability models that must be estimated. In metric learning, G mixtures are estimated, one for each of the G possible classes from which a sample x may be drawn. In mixture SDA, G^2 mixture models are estimated. Each sample x is hypothesized drawn from class $g = 1, 2, \ldots G$, and its similarities to each of the G classes are modeled by the mixture (25), with $h = 1, 2. \ldots G$. When the number of classes grows, or when the number of components in each mixture model grows, the quadratic growth in the number of needed models presents a challenge in robust parameter estimation, especially when the number of available training samples is relatively small. However, this problem is mitigated by the fact that the component SDA parameters may be robustly estimated with smaller training sets than in metric mixture models due to the simpler, univariate estimation problem at the heart of SDA classification. The next section discusses the mixture SDA parameter estimation procedure.

4.1 Estimating the parameters for mixture SDA models

Computing the SDA mixture model for the similarities of samples $x \in \mathcal{X}_g$ to class h requires estimating the number of components c_h, the component centroids $\{\mu_{hl}\}$, the component weights $\{w_{ghl}\}$ and the component SDA parameters $\{\lambda_{ghl}\}$ and $\{\gamma_{ghl}\}$. This section describes an EM algorithm for estimating these mixture parameters. The algorithm parallels the EM approach for estimating GMM parameters (Duda et al., 2001; Hastie et al., 2001); it is first summarized below, and then explained in detail in the following sections.

Let $\theta_{gh} = \{\{w_{ghl}\}, \{\gamma_{ghl}\}, \{\lambda_{ghl}\}\}$ for $l = 1, 2 \ldots c_h$ be the set of parameters for the class h mixture model to be estimated under the assumption that the training samples z_i, for $i = 1, 2, \ldots n_g$ are drawn identically and independently. Denote by C a random component of the mixture and by $P(C = l \,|\, s(z_i, \mu_{hl}), \theta_{gh})$ the responsibility (Hastie et al., 2001) of the lth component for the ith training sample similarity $s(z_i, \mu_{hl})$. Also write $P(s(z_i, \mu_{hl}) \,|\, C = l, \theta_{gh}) = \gamma_{ghl} e^{\lambda_{ghl} s(z_i, \mu_{hl})}$.

The proposed EM algorithm for mixture SDA is:

1. Compute the centroids $\{\mu_{hl}\}$ with K-medoids algorithm.
2. Initialize the parameters $\{w_{ghl}\}$ and the components $P(s(z_i, \mu_{hl}) \,|\, C = l, \theta_{gh})$.
3. E step: compute the responsibilities

$$P(C = l|s(z_i, \mu_{hl}), \theta_{gh}) = \frac{w_{ghl} P(s(z_i, \mu_{hl})|C = l, \theta_{gh})}{\sum_{l=1}^{c_h} w_{ghl} P(s(z_i, \mu_{hl})|C = l, \theta_{gh})}. \tag{27}$$

4. M step: compute model parameters
   (a) Find the λ_{ghl} which solves

$$E_{P(T(x)|Y=g)}[s(X, \mu_{hl})] = \frac{\sum_{i=1}^{n_g} s(z_i, \mu_{hl}) P(C = l|s(z_i, \mu_{hl}), \theta_{gh})}{\sum_{i=1}^{n_g} P(C = l|s(z_i, \mu_{hl}), \theta_{gh})}. \tag{28}$$

   (b) Compute the corresponding scaling factor

$$\gamma_{ghl} = \frac{1}{\sum\limits_{s(X,\mu_{hl}) \in \Omega} e^{\lambda_{ghl} s(X, \mu_{hl})}}. \tag{29}$$

   (c) Compute the component weights

$$w_{ghl} = \frac{1}{n_g} \sum_{i=1}^{n_g} P(C = l|s(z_i, \mu_{hl}), \theta_{gh}). \tag{30}$$

5. Repeat E and M steps until convergence criterion is satisfied.

Note that, just like EM for GMMs, the EM algorithm for mixture SDA involves iterating the E step, which estimates the responsibilities, and the M step, which estimates the parameters that maximize the expected log-likelihood of the training data. At each iteration of the M step, the explicit expression (30) updates the component weights. However, unlike EM for GMMs, the update expression for the component parameters (28) is implicit and must be solved numerically. Another difference between the GMM and SDA EM algorithms is in how the centroids are estimated. For GMMs, the component means $\{u_{hl}\}$, which are the metric centroids, are updated at each iteration of the M step. For mixture SDA, the centroids $\{\mu_{hl}\}$ are estimated at the beginning of the algorithm and kept constant throughout the iterations.

The update expressions for the mixture SDA parameters are derived from the expression of the expected log-likelihood of the observed similarities. A standard assumption in EM is that the observed data are independent and identically distributed given the class and mixture component. For mixture SDA, this assumption means that the training sample similarities $\{\mathcal{T}_g(z_i)\} = \{s(z_i, \mu_{hl})\}$, $z_i \in \mathcal{X}_g$ to the component centroids are identically distributed and conditionally independent given the *l*th class component. Then, the expected log-likelihood of $\{\mathcal{T}_g(z_i)\}$ is

$$L(\{\mathcal{T}_g(z_i)\}|\theta_{gh}) = \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \log\left(w_{ghl}\gamma_{ghl}e^{\lambda_{ghl}s(z_i,\mu_{hl})}\right) P(C=l|s(z_i,\mu_{hl}),\theta_{gh}). \quad (31)$$

Using the properties of the logarithm and rearranging the terms, $L(\{\mathcal{T}_g(z_i)\} \,|\, \theta_{gh})$ splits into the terms depending on w_{ghl} and the terms depending on λ_{ghl} and γ_{ghl}:

$$\begin{aligned} L(\{\mathcal{T}_g(z_i)\}|\theta_{gh}) &= \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \log(w_{ghl})P(C=l|s(z_i,\mu_{hl}),\theta_{gh}) \\ &+ \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \log(\gamma_{ghl})P(C=l|s(z_i,\mu_{hl}),\theta_{gh}) \qquad (32) \\ &+ \sum_{i=1}^{n_g}\sum_{h=1}^{G}\sum_{l=1}^{c_h} \lambda_{ghl}s(z_i,\mu_{hl})P(C=l|s(z_i,\mu_{hl}),\theta_{gh}). \end{aligned}$$

The standard EM approach to maximizing (32) is to set its partial derivatives with respect to the parameters to zero and solve the resulting equations. This is the approach adopted here for estimating the mixture SDA parameters θ_{gh} for all g, h.
The derivation of the expression for the component weights $\{w_{ghl}\}$ follows directly from (32); both the derivation of and the final expression for the component weights are identical to the metric mixtures case. Section 4.1.1 re-derives the well-known expression for w_{ghl}.
Applying the EM approach, however, does not lead to explicit expressions for $\{\lambda_{ghl}\}$ and $\{\gamma_{ghl}\}$. Instead, it leads to many single-parameter constraint expressions for the mean similarities of the training data to the mixture component centroids. These expressions are solved with the same numerical solver used in the single-centroid SDA classifier.

4.1.1 Estimating the component weights

To compute the log-likelihood-maximizing weights w_{ghl}, one uses the standard technique of taking the derivative of the log-likelihood with respect to w_{ghl}, setting it to zero, and solving the resulting expression for w_{ghl}. The constraint $\sum_{l=1}^{c_h} w_{ghl} = 1$ is taken into account with the Lagrange multiplier η:

$$\frac{\partial}{\partial w_{ghl}}\left\{L(\{\mathcal{T}_g(z_i)\}|\theta_{gh}) + \eta\left(\sum_{l=1}^{c_h} w_{ghl} - 1\right)\right\} = \sum_{i=1}^{n_g}\frac{1}{w_{ghl}}P(C=l|s(z_i,\mu_{hl}),\theta_{gh}) + \eta = 0,$$

which gives the well-known expression for the component weights of a mixture model in terms of the responsibilities:

$$w_{ghl} = \frac{1}{n_g} \sum_{i=1}^{n_g} P(C = l | s(z_i, \mu_{hl}), \theta_{gh}). \tag{33}$$

4.1.2 Estimating γ_{ghl} and λ_{ghl}

The same approach used for estimating the component weights $\{w_{ghl}\}$ is adopted to estimate the SDA parameters $\{\gamma_{ghl}\}$ and $\{\lambda_{ghl}\}$: Find the likelihood-maximizing values of the parameters by setting the corresponding partial derivatives to zero and solving the resulting equations. First, since each γ_{ghl} is simply a scaling factor that ensures that each mixture component is a probability mass function, one rewrites

$$\gamma_{ghl} = \frac{1}{\sum_{s(X,\mu_{hl})\in\Omega} e^{\lambda_{ghl} s(X,\mu_{hl})}}, \tag{34}$$

where $X \in \mathcal{X}_g$ is a random sample from class g, $s(X, \mu_{hl})$ is its corresponding random similarity to component centroid μ_{hl}, and Ω is the set of all possible similarity values. Substituting (34) into (32), setting the partial derivative of $L(\{\mathcal{T}_h(z_i)\} \mid \theta_{gh})$ with respect to λ_{ghl} to zero, and rearranging the terms gives

$$\frac{\sum_{s(X,\mu_{hl})\in\Omega} s(X,\mu_{hl}) e^{\lambda_{ghl} s(X,\mu_{hl})}}{\sum_{s(X,\mu_{hl})\in\Omega} e^{\lambda_{ghl} s(X,\mu_{hl})}} \sum_{i=1}^{n_g} P(C = l | s(z_i, \mu_{hl}), \theta_{gh}) = \sum_{i=1}^{n_g} s(z_i, \mu_{hl}) P(C = l | s(z_i, \mu_{hl}), \theta_{gh}). \tag{35}$$

The first term on the left side of (35) is simply the definition of the expected value of the similarity of samples in class g to the lth centroid of class h. Thus, one rewrites (35)

$$E_{P(\mathcal{T}(x)|Y=g)}[s(X, \mu_{hl})] = \frac{\sum_{i=1}^{n_g} s(z_i, \mu_{hl}) P(C = l | s(z_i, \mu_{hl}), \theta_{gh})}{\sum_{i=1}^{n_g} P(C = l | s(z_i, \mu_{hl}), \theta_{gh})}. \tag{36}$$

Expression (36) is an equality constraint on the expected value of the similarity of samples $z_i \in \mathcal{X}_g$ to the component centroids μ_{hl} of class h. This is the same type of constraint that must be solved in the mean-constrained, maximum entropy formulation of single-centroid SDA (9). In (9), the mean similarity of samples from class g to the single centroid of class h is constrained to be equal to the observed average similarity. Analogously, in (36), the mean similarity of the samples from class g to the lth centroid of class h is constrained to be equal to the weighted sum of the observed similarities, where each similarity is weighted by its normalized responsibility. To solve for λ_{ghl}, one uses the same numerical procedure used to solve (9) and described in Section 2.1. Thus, solving for all the $\{\lambda_{ghl}\}$ requires solving the $G \times \sum_{h=1}^{G} c_h$ expressions of (36).

It is not surprising that taking the EM approach to estimating λ_{ghl} has lead to the same expressions for the mean constraints in the maximum entropy approach to density estimation. It is known that maximum likelihood (ML) - the foundation for EM - and maximum entropy are dual approaches to estimating distribution parameters which lead to the same unique solution based on the observed data (Jordan, 20xx). The ML approach assumes exponential distributions for the similarities, maximizes the likelihood, and arrives at constraint expressions whose solutions give the desired values for the parameters. The maximum entropy approach assumes the constraints, maximizes the entropy, and arrives at exponential distributions whose parameters satisfy the given constraints. This powerful dual relationship between ML and maximum entropy extends from metric problems to similarity-based problems; for this reason it leads to the the constraint expression (36), from which λ_{ghl} is numerically computed. The corresponding γ_{ghl} is found by applying (34).

4.1.3 Estimating the centroids

Estimating the centroids of a mixture model encompasses two problems: estimating the number of components (i.e. centroids) $\{c_h\}$, and estimating the centroids $\{\mu_{hl}\}$. This work adopts the common metric learning practice of cross-validating the number of mixture components $\{c_h\}$. The centroids $\{\mu_{hl}\}$ are estimated with the K-medoids algorithm (Hastie et al., 2001), using the maximum-sum-similarity criterion (3). The initial centroids are selected randomly from the training set samples $z_i \in \mathcal{X}_h$.

4.1.4 Initializing EM for SDA

In this work, the component weights $\{w_{ghl}\}$ are uniformly initialized to w_{ghl} = 1=c_h and the components are assigned uniform initial probability $P(s(z_i, \mu_{hl}) \,|\, C = l, \theta_{gh}) = 1/c_h$. This initialization reflects the assumption that initially the mixture components equally contribute to a sample's class-conditional probability: it is the least-assumptive initialization. Another strategy would be to initialize the weights by the fraction of training samples assigned to the clusters which result from estimating the centroids with K-medoids. The component probabilities may also be initialized by estimating the SDA parameters $\{\lambda_{ghl}\}$ and $\{\gamma_{ghl}\}$ from the K-medoids clusters. This is analogous to the GMM initialization strategy based on the results of the K-means algorithm. In practice, the simple uniform initialization works well.

5. Experimental results

SDA, local SDA, mixture SDA, and nnSDA are compared to other similarity-based classifiers in a series of experiments: the tested classifiers are the nearest centroid (NC), local nearest centroid (local NC), k-nearest neighbors (k-NN) in similarity space, condensed nearest neighbor (CNN) (Hastie et al., 2001) in similarity space, and the potential support vector machine (PSVM) (Hochreiter & Obermayer, 2006). When the features underlying the similarity are available, the classifiers are also compared to the naive Bayes classifier (Hastie et al., 2001). The counting similarity (the number of features identically shared by two binary vectors) and the VDM (Stanfill & Waltz, 1986; Cost & Salzberg, 1993; Wilson & Martinez, 1997) similarities are used to compute the similarities on which the classifiers operate, except for cases in which similarity is provided as part of benchmark datasets.

The first set of comparisons involves simulated binary data, where each class is generated by random perturbations of one or two centroids. The *perturbed centroids* simulation is a scenario where each class is characterized by one or two prototypical samples (centroids), but samples have random perturbations that make them different from their class centroid

in some features. Thus, this simulation fits the centroid- based SDA models, in that each class is defined by perturbations around one or two prototypical centroids.
Then, three benchmark datasets are investigated: the protein dataset, the voting dataset, and the sonar dataset. The results on the simulated and benchmark datasets show that the proposed similarity-based classifiers are effective in classification problems spanning several application domains, including cases when the similarity measures do not possess the metric properties usually assumed for metric classifiers and when the underlying features are unavailable.
For local SDA and local NC, the class prior probabilities are estimated as the empirical frequency of each class in the neighborhood; for SDA, mixture SDA, nnSDA, NC, and CNN they are estimated as the empirical frequency of each class in the entire training data set. The k-NN classifier is implemented in the standard way, with the neighborhood defined by the test sample's k most similar training samples, irrespective of the training samples class. Ties are broken by assigning a test sample to class one.

5.1 Perturbed centroids

In this two-class simulation, each sample is described by d binary features such that $B = \{0, 1\}^d$. Each class is defined by one or two prototypical sets of features (one or two centroids). Every sample drawn from each class is a class centroid with some features possibly changed, according to a feature perturbation probability. Several variants of the simulation are presented, using different combinations of number of class centroids, feature perturbation probabilities, and similarity measures. Given samples $x, z \in B$, $s(x, z)$ is either the counting or the VDM similarity. The simulations span several values for the feature dimensions d and are run several times to better estimate mean error rates. For each run of the simulation and for each number of features considered, the neighborhood size k for local SDA, local NC, and k-NN is determined independently for the three classifiers by leave-one-out cross-validation on the training set of 100 samples; the range of tested values for k is $\{1, 2, ... 20, 29, 39, ... , 99\}$. The optimum k is then used to classify 1000 test samples. Similarly, the candidate numbers of components for mixture SDA and for CNN are $\{2, 3, 4, 5, 7, 10\}$. To keep the experiment run time within a manageable practical limit, five-fold cross validation was used to determine the number of components for mixture SDA, and the mixture SDA EM algorithm was limited to 30 iterations for each cross-validated mixture model. The parameters for the PSVM classifier are cross-validated over the range of possible values $\epsilon = \{0.1, 0.2, ... 1\}$ and $C = \{1, 51, 101, ... 951\}$.

The perturbed centroid simulation results are in Tables 1-8. For each value of d, the lowest mean cross-validation error rate is in bold. Also in bold for each d are the error rates which are not statistically significantly different from the lowest mean error rate, as determined by the Wilcoxon signed rank test for paired differences, with a significance level of 0.05. The naive Bayes classifier results are also included for reference.

5.1.1 Perturbed centroids – one centroid per class

Each class is generated by perturbing one centroidal sample. There are two, equally likely classes, and each class is defined by one prototypical set of d binary features, c_1 or c_2, where c_1 and c_2 are each drawn uniformly and independently from $\{0, 1\}^d$. A training or test sample z drawn from class g has the ith feature $z[i] = c_g[i]$ with probability $1 - p_g$, and $z[i] \neq c_g[i]$ with perturbation probability p_g. In one set of simulation results $p_1 = 1/3$ and $p_2 = 1/30$; thus, class two is well-clustered around its generating centroid and the two classes are well-separated. In another set of simulation results, $p_1 = 1/3$ and $p_2 = 1/4$ and the two classes are not as well separated. Classifiers are trained on 100 training samples and tested on 1000 test samples per run; twenty runs are executed for a total of 20, 000 test samples. The number of features d ranges from $d = 2$ to $d = 200$ in the simulation, but the number of training samples is kept constant at 100, so that $d = 200$ is a sparsely populated feature space. This procedure was repeated for the counting and for the VDM similarities, so there are four sets of results for the one centroid simulation, depending on the perturbation probabilities and the similarity measure used. The results are in Tables 1-4.

The performance of all classifiers increases as d increases. For large d, the feature space is sparsely populated by the training and test samples, which are segregated around their corresponding generating centroids. This leads to good classification performance for all classifiers. For small d, the feature space is densely populated by the samples, and the two classes considerably overlap, negatively affecting the classification performance.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**15.58**	**15.58**	35.13	23.47	49.80	**15.58**	19.22	**16.07**	15.58
4	**11.74**	**11.82**	23.97	22.54	39.08	12.05	13.85	13.01	11.98
8	**4.88**	6.10	**12.85**	14.07	**6.35**	6.19	7.86	6.21	4.63
12	**3.35**	3.97	10.16	11.50	5.00	4.26	6.01	3.74	2.46
25	**2.27**	3.50	**7.36**	11.49	**2.27**	3.49	5.30	**2.16**	1.77
40	1.86	2.79	**3.65**	8.79	**1.38**	2.79	4.38	**1.33**	1.24
50	2.02	2.37	**2.71**	7.94	**1.60**	2.31	3.24	**1.33**	1.29
75	1.90	2.58	2.56	7.83	**1.31**	2.27	3.82	**1.43**	1.43
100	2.11	2.32	2.05	5.92	**1.03**	2.16	3.69	1.65	1.65
125	**1.86**	**2.07**	**1.67**	6.21	**1.47**	**1.96**	3.56	**1.58**	1.58
150	**1.40**	**1.50**	**1.23**	4.86	**1.08**	**1.44**	2.55	**1.20**	1.20
175	**1.63**	**1.64**	**1.37**	4.28	**1.29**	**1.60**	2.59	**1.33**	1.33
200	**1.41**	**1.42**	**1.26**	4.20	**0.99**	**1.38**	2.67	**1.26**	1.26

Table 1. Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **counting** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/30}$.

Across all four sets of results, the naive Bayes classifier almost always gives the best performance. Its assumption that the features are independent captures the true underlying relationship of the sample features makes the naive Bayes classifier well suited for these particular data sets: indeed the samples are generated as random vectors of independent binary features. The consequent excellent performance of the naive Bayes classifier provides a reference point for the other classifiers. More generally, when a classification problem involves samples natively embedded in an Euclidean space, as in these perturbed centroids experiments, metric-space classifiers like naive Bayes can perform well. In these cases, the similarity-based classification framework provides no clear advantage.

On the other hand, naive Bayes cannot be used when the samples are not described by vectors of independent features, either because the features are not known, the independence assumption is too restrictive for effective performance, or because the Euclidean representation does not sufficiently capture the pairwise relationships of the samples. In these cases, the similarity-based techniques provide solutions to classification problems. Thus, in these perturbed centroids experiments, the naive Bayes classifier is a good reference for assessing the effectiveness of the similarity-based classifiers, but it is not considered for the Wilcoxon significance tests because it is not generally applicable to similarity-based classification.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**30.88**	**30.88**	48.48	**30.29**	49.70	**30.88**	31.07	**30.66**	30.62
4	31.19	**30.30**	35.56	**29.83**	44.41	30.92	32.63	**29.25**	29.18
8	22.63	22.56	**23.30**	**21.95**	33.13	23.12	24.13	**21.18**	21.02
12	18.11	18.58	**18.39**	**16.99**	29.52	18.42	20.04	**17.03**	16.56
25	12.16	13.90	13.40	13.17	26.21	10.40	14.82	**8.84**	7.96
40	7.87	11.42	10.33	12.45	17.59	7.26	11.58	**5.67**	4.91
50	6.59	8.98	9.47	11.32	19.36	6.42	10.37	**4.43**	3.69
75	5.32	6.96	6.06	8.42	12.29	4.17	7.89	**2.67**	2.19
100	4.84	6.56	5.66	6.96	9.09	3.93	5.61	**2.88**	2.69
125	3.23	5.10	3.98	6.25	11.82	2.65	4.81	**2.08**	2.01
150	3.03	3.84	3.07	5.13	**6.38**	2.61	4.50	**1.97**	1.94
175	3.56	3.86	3.86	6.30	**4.81**	2.83	4.33	**2.38**	2.38
200	2.61	2.66	2.78	3.66	**2.42**	2.08	3.15	**1.75**	1.75

Table 2. Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **counting** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/4}$.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**16.36**	**16.36**	34.13	26.41	48.90	**16.36**	22.17	**16.87**	16.36
4	**11.28**	**11.18**	15.22	19.10	38.16	**11.37**	12.23	12.20	11.23
8	**6.71**	7.51	**9.89**	14.52	**8.09**	**7.44**	**8.71**	**6.89**	5.42
12	**4.69**	**6.17**	**5.20**	12.99	**4.48**	5.85	7.36	**4.78**	3.33
25	2.96	3.46	2.56	9.87	4.08	3.35	4.90	**2.09**	1.59
40	2.36	2.60	2.62	7.28	4.65	2.49	4.37	**1.78**	1.67
50	2.60	2.86	2.61	7.02	4.45	2.80	4.70	**1.97**	1.94
75	2.42	2.59	2.11	6.09	2.96	2.47	4.03	**1.93**	1.93
100	1.88	1.90	1.74	3.97	2.38	1.88	2.46	**1.68**	1.68
125	1.67	1.68	**1.54**	3.25	**2.03**	1.67	2.39	**1.52**	1.52
150	1.68	1.68	**1.65**	2.92	**1.89**	1.68	2.17	**1.64**	1.64
175	1.60	1.61	**1.57**	2.59	**1.63**	1.61	2.08	**1.56**	1.56
200	**1.63**	**1.63**	**1.62**	2.25	**2.14**	**1.63**	1.84	**1.62**	1.62

Table 3. Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/30}$.

d	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**34.38**	**34.38**	42.72	**34.17**	48.50	**34.38**	**33.76**	**34.56**	34.66
4	29.85	30.10	30.04	28.47	44.27	29.59	30.55	**27.85**	27.46
8	25.66	26.05	24.38	**24.16**	26.99	25.52	26.47	**24.24**	23.41
12	18.71	19.28	**17.99**	**17.98**	22.86	19.95	20.82	18.26	16.90
25	10.75	11.49	9.92	10.58	14.21	11.01	12.04	**9.10**	8.03
40	8.00	8.28	6.91	7.88	8.68	7.57	9.10	**6.02**	5.00
50	6.86	7.97	5.93	6.98	8.03	5.98	8.81	**4.81**	3.82
75	4.18	5.24	3.45	4.81	4.08	3.53	4.22	**2.63**	2.02
100	3.76	4.06	3.19	4.02	**2.71**	3.26	3.62	2.50	2.22
125	2.67	3.02	2.03	2.73	2.15	2.13	2.86	**1.65**	1.52
150	**2.90**	2.97	2.56	3.41	**1.73**	2.68	3.16	**2.25**	2.13
175	**2.56**	**2.75**	2.25	2.62	**2.07**	2.46	2.54	**2.12**	2.10
200	**2.31**	**2.39**	2.18	2.56	**1.77**	2.27	2.90	**2.02**	1.98

Table 4. Perturbed centroids experiment - One centroid per class. Misclasssification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p_1 = 1/3}$ and $\mathbf{p_2 = 1/4}$.

With few exceptions the PSVM performs best on the four sets of results on a wide range of *d*. This is likely because the PSVM classifies a test sample based on its similarities to the entire training set. In contrast, local methods such as local SDA, local NC, nnSDA, k-NN, and CNN make use of a subset of the training samples and thus have less information available to classify. Global methods based on the similarity-to-class-centroid summary statistic such as SDA, NC, and CNN also use less information. It is plausible that the ability to make use of all the similarity information in the training set and to optimally weight the similarities to the training samples gives the PSVM a performance advantage over the other techniques. However, in spite of this advantage, the results show that for low and high values of *d* the SDA-based techniques yield statistically equivalent performance to the PSVM, and in some cases match or exceed its results. When the PSVM statistically produces significantly different results from the other techniques, its performance does not hugely surpass them. Thus the similarity-based techniques possess the ability to produce good classification results using less information. This quality can be immensely useful when few training samples are available.

In all four sets of results, the SDA-based algorithms generally perform better than their non-generative counterparts: local SDA performs better than local NC and SDA performs better than NC. This shows that generative models based on the similarity of samples to local or global class centroids provide increased discriminative power over the non-generative centroid-based similarity models. Furthermore, in almost all cases across the four sets of results, local SDA performs better than SDA. While the classification performance of SDA is good, its inherent model bias prevents it from achieving even better performance; local SDA is not as susceptible to model bias, and is able to perform very well. Still, the SDA performance is close to that of the local SDA in all cases and sometimes it surpasses it (VDM similarity with $p_2 = 1/4$), a confirmation that the single-centroid generative model at the heart of SDA matches well the perturbed single-centroid experimental setup for these sets of results.

The similarity-space k-NN performs well, albeit not as well as the PSVM. Compared to SDA, k-NN performs better only for the counting similarity and $p_2 = 1/4$. Since SDA matches well the class models for the generated samples, it is not surprising that it performs better than k-NN, which does not rely on class models. However, k-NN does better when the class two perturbed samples are more likely to differ from their generating class two centroid ($p_2 = 1/4$), that is when the classes overlap more. In this case, it is more di±cult to estimate the class centroids, and the SDA performance is affected. On the other hand, SDA is better than k-NN for the VDM similarity, for both $p_2 = 1/30$ and $p_2 = 1/4$. The VDM similarity is calculated from class-dependent lookup tables pre-computed from the training set, and this additional information seems to favor the SDA classifier more than the k-NN. Local SDA, performs slightly better than k-NN when $p_2 = 1/30$ for both counting and VDM similarities.
The CNN classifier generally does not perform as well as k-NN. This is expected, because, as for its metric learning analog, the condensing process primarily aims to reduce the size of large training sets and possibly eliminate outliers rather than to improve classification performance. The observed lower performance of CNN compared to k-NN reflects the expectation that classification performance will degrade when using the condensed training set instead of the full set of available training samples.
The nnSDA classifier performs well for the counting similarity when $p_2 = 1/30$, and in general for higher values of d. For low values of d the performance is particularly poor: for $d = 2$ the error rate is essentially equal to that of a random classifier (50%) and for $d = 4$ it is only slightly better. In fact, the nnSDA performance is limited by the interplay of its asymptotic behavior and the value of d. Recall that by Lemma (1) from Section 3.1, $P(s(x,Z_k) = s_{max}) \to 1$ as $k,N \to\infty$ and $k/N \to 0$, where k is the neighborhood size, N is the number of available training samples, and Z_k is the k-th nearest neighbor of test sample x. Then, it follows that $P(s_{nn,h}(x) = s_{max}) \to 1$ for all h as k, $n \to\infty$, because $s_{nn,h}(x) = s(x,Z_1)$ for $Z_1 \in \mathcal{X}_h$ as $k\to\infty$. Thus, for nnSDA, the similarities of a test sample to its nearest neighbors in each class are all identical in the limit of infinite number of training samples. Consequently, for a large training set, all class discriminants in the nnSDA classification rule (17) are identical and therefore uninformative. The classification rule (17) reduces to the trivial rule that classifies according to the cost-adjusted class priors,

$$\hat{y} = \arg\min_{f=1,\ldots,G} \sum_{g=1}^{G} C(f,g)P(Y = g). \tag{37}$$

When 0-1 costs are used, as in this simulation, the rule (37) always classifies as the class g with the highest prior probability $\hat{P}(Y = g)$, estimated as the empirical frequency from the training data:

$$\hat{y} = \arg\max_{g=1,\ldots,G} \hat{P}(y = g). \tag{38}$$

In this experiment, the samples are generated from two, a priori equally likely classes, so the limit misclassification rate is $1 - \max_g \hat{P}(Y = g) \approx 0.5$.

The limit error rate is noticeable when d is small. In this case the similarity can take on values in a limited range bounded by d ($s(x, z) \in [0, 1 \ldots d]$ for the counting similarity) and the training set is highly redundant. Thus, a test sample x is very likely to be maximally similar to its nearest neighbor from each class, and $s_{nn,h}(x)$ is uninformative. In higher dimensions, the experimental results show that the training set is sufficiently sparse for effective classification. Thus nnSDA is a viable classifier for sparse training sets which do not cover the entire range of possible values for the chosen similarity. In applications when few training samples are available, nnSDA can be a valuable tool for achieving actionable classification results.

5.1.2 Perturbed centroids – two centroids per class

In this variation of the perturbed centroids simulation, each class is characterized by two prototypical samples, c_{11}, c_{12} for class one, and c_{21}, c_{22} for class two. Each time the simulation is run, the centroids c_{11}, c_{12}, c_{21}, c_{22} are drawn independently and identically using a uniform distribution over $\mathcal{B}$.

Every sample drawn from each class is a perturbed version of one of the two class prototypes, where the class labels are drawn independently and identically with probability 1/2. A training or test sample z drawn from class one is randomly selected to be $z = c_{11}$ or $z = c_{12}$ with probability 1/2, and then for each $i = 1, \dots, d$, z's ith feature is probabilistically perturbed so that $z[i] \neq c_{11}[i]$ with probability p_{11} (or $z[i] \neq c_{12}[i]$ with probability p_{12}). Thus on average, a randomly drawn sample based on c_{11} will have dp_{11} features that are different from the class prototype c_{11}'s features. Likewise, a training or test sample v drawn from class two starts out as $v = c_{21}$ or $v = c_{22}$ with probability 1/2, but then for each $i = 1, \dots, d$, v's ith feature is changed so that $v[i] \neq c_{21}[i]$ with probability p_{21} (or $v[i] \neq c_{22}[i]$ with probability p_{22}). The number of features d ranges from $d = 2$ to $d = 200$ in the simulation, but the number of training samples is kept constant at 100, so that $d = 200$ is a sparsely populated feature space. Two different sets of values of the perturbation probabilities p_{11}, p_{12}, p_{21}, p_{22} were used: in the first case $p_{11} = p_{12} = 1/3$ and $p_{21} = p_{22} = 1/30$, so that the class two samples are much more tightly clustered around c_{21} and c_{22} than the class one samples are with respect to c_{11} and c_{12}. In the second case, $p_{11} = p_{12} = 1/3$ and $p_{21} = p_{22} = 1/4$, resulting in a higher Bayes error. Each simulation was run twenty times, for a total of 20,000 test samples. The resulting mean error rates are given in Tables 5-8.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**26.41**	**26.41**	47.52	**28.93**	38.98	49.30	**26.41**	**27.51**	**27.16**	29.09
4	**13.80**	**13.68**	34.77	18.32	34.84	38.04	**13.26**	16.84	17.18	17.95
8	**9.23**	**9.25**	29.32	**13.96**	26.77	**9.54**	**9.29**	12.82	12.62	9.96
12	**5.61**	6.47	31.20	**10.56**	27.05	**7.35**	**6.25**	10.87	8.72	7.96
25	3.11	4.37	28.75	**2.39**	25.90	3.21	4.03	9.45	4.08	2.67
40	2.88	4.25	30.84	**6.54**	28.23	**1.91**	3.94	8.69	**2.21**	1.39
50	2.94	4.89	27.77	**1.73**	30.12	**1.32**	4.35	9.10	**1.77**	1.16
75	2.04	3.21	26.38	3.69	27.74	**1.61**	2.75	7.61	**0.95**	1.12
100	2.21	3.03	25.39	**2.30**	24.58	**1.37**	2.60	5.25	**1.52**	1.08
125	2.46	2.96	25.51	**4.74**	24.83	**1.42**	2.68	5.33	**1.59**	1.47
150	1.55	1.80	25.00	4.54	26.55	**1.54**	1.76	5.34	**1.00**	0.78
175	1.93	2.38	25.32	**2.72**	21.40	**1.16**	2.02	4.17	**1.29**	1.21
200	1.44	1.61	23.87	**1.63**	19.28	**1.38**	1.49	4.45	**1.10**	0.95

Table 5. Perturbed centroids experiment - Two centroids per class. Misclassification percentage for **counting** similarity, perturbation probabilities $\mathbf{p_{11} = p_{12} = 1/3}$ and $\mathbf{p_{21} = p_{22} = 1/30}$.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**40.02**	**40.19**	49.36	40.21	42.87	49.90	**39.88**	**37.64**	**39.85**	39.91
4	**33.43**	33.77	39.96	39.13	37.95	46.61	**33.17**	**34.91**	**34.67**	32.62
8	**29.81**	31.84	36.82	33.52	34.94	40.20	**29.10**	35.09	**30.12**	28.43
12	**27.27**	29.42	35.24	39.11	33.19	38.37	**27.19**	30.30	**27.51**	25.90
25	19.89	22.91	29.83	39.86	28.81	33.77	**17.09**	22.75	**16.79**	17.51
40	14.05	16.91	28.60	34.62	26.70	24.45	**11.49**	18.14	13.10	12.72
50	11.65	14.64	26.82	34.22	25.61	31.04	**9.04**	15.90	10.19	9.68
75	8.16	9.01	24.61	30.99	24.48	20.37	**5.84**	12.71	**7.51**	6.00
100	7.67	8.00	23.59	30.20	21.68	17.09	**4.83**	9.68	**4.37**	3.96
125	6.05	6.79	23.70	26.82	22.50	15.18	**3.52**	7.87	**3.94**	2.99
150	5.05	6.31	22.36	26.24	21.62	11.50	**3.13**	6.13	**2.90**	2.79
175	3.72	4.15	25.02	23.29	23.79	10.43	**2.14**	6.29	2.39	1.81
200	3.45	3.85	21.86	21.74	21.83	9.41	**2.19**	5.28	**2.36**	2.24

Table 6. Perturbed centroids experiment - Two centroids per class. Misclassification percentage for **counting** similarity, perturbation probabilities $\mathbf{p}_{11} = \mathbf{p}_{12} = \mathbf{1/3}$ and $\mathbf{p}_{21} = \mathbf{p}_{22} = \mathbf{1/4}$.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**25.83**	**25.61**	30.60	34.05	39.07	49.00	**25.61**	**30.83**	**28.64**	27.95
4	15.22	**13.46**	22.30	17.71	26.00	40.72	**13.59**	20.42	**16.87**	18.01
8	**10.96**	**11.81**	**11.77**	**13.66**	22.53	16.63	**11.15**	**12.14**	14.16	11.76
12	**8.17**	9.46	**7.92**	**9.41**	19.07	**9.41**	**8.52**	12.60	10.11	7.58
25	4.52	6.19	3.77	**4.39**	16.32	**5.95**	5.92	8.30	**4.23**	3.63
40	2.96	3.73	**2.30**	**2.77**	16.41	4.59	3.79	6.66	**2.79**	2.25
50	2.59	3.86	**1.74**	**2.58**	15.96	3.50	3.56	6.61	1.79	1.80
75	2.33	3.40	**1.57**	**1.57**	13.69	**3.20**	2.90	5.59	**1.15**	1.45
100	2.17	3.06	**1.52**	**1.03**	11.35	2.81	2.79	5.62	1.24	1.52
125	2.51	2.90	**1.63**	**1.36**	11.36	2.25	2.74	5.05	**1.39**	1.62
150	2.10	2.50	**1.39**	**1.44**	11.45	2.32	2.30	4.83	**1.03**	1.38
175	2.12	2.33	**1.47**	**1.44**	10.82	**1.62**	2.20	4.21	**1.31**	1.46
200	1.80	1.99	**1.19**	**1.88**	10.46	2.04	1.93	3.28	**1.32**	1.18

Table 7. Perturbed centroids experiment - Two centroids per class. Misclassification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p}_{11} = \mathbf{p}_{12} = \mathbf{1/3}$ and $\mathbf{p}_{21} = \mathbf{p}_{22} = \mathbf{1/30}$.

For all four sets of results, the local SDA classifier performs better than the local NC classifier. This result agrees with the analogous case for the single centroid experiments and attests to the advantage that similarity-based generative models provide over simpler nearest-centroid classifiers. However, the SDA classifier yields better classification than its counterpart NC classifier only for the VDM similarity. For the counting similarity, SDA does not provide an advantage over NC. There are two causes that contribute to this outcome. First, the single-centroid SDA is a biased model that does not match the true two-centroids-per-class experimental setup. Consider class one and its centroids, c_{11} and c_{12}. SDA at best correctly estimates one of the two centroids per class, let's say $\hat{c}_{11}$. Thus, the estimated centroid- based generative model for class one is a good match for the samples which are generated as random perturbations of c_{11}. The model, however, is not a good match for samples generated as random perturbations of c_{12}. The model cannot distinguish the similarities of these class one samples to $\hat{c}_{11}$ from their similarities to the centroids of class two. The result is that the c_{12}-generated samples are classified according to the class priors, that is half as class one and half as class two. The same argument applies to class two, so that overall about 25% of the samples are misclassified. Indeed, the SDA error rates quickly settle to ≈25% for the counting similarity for medium to large values of d. For lower d, the class overlap due to the density of the feature space dominates the misclassification rate.

d	Local SDA	Local NC	SDA	Mixture SDA	NC	nnSDA	k-NN	CNN	PSVM	Naive Bayes
2	**39.98**	**40.01**	**42.57**	**40.26**	**40.77**	48.00	**40.01**	**39.66**	**41.08**	38.89
4	**37.28**	**37.45**	**37.98**	**34.99**	**38.53**	48.95	**37.21**	**37.09**	**36.19**	37.34
8	**30.80**	**32.80**	**30.62**	**31.26**	**30.99**	36.88	31.84	33.43	**30.23**	29.37
12	**27.26**	**28.85**	**27.87**	**26.97**	**28.59**	31.06	**27.65**	29.82	**29.68**	25.15
25	**21.87**	**21.86**	**20.88**	**22.03**	21.77	23.96	**20.82**	23.27	21.55	17.63
40	16.56	18.50	16.41	18.01	17.96	19.08	16.91	18.98	**15.20**	12.44
50	14.92	17.22	16.04	16.11	17.65	16.89	14.96	16.07	**13.92**	11.21
75	11.98	13.40	12.41	11.68	13.91	12.16	10.57	11.65	**8.99**	7.53
100	8.54	9.94	9.04	9.01	11.09	8.83	7.55	9.66	**6.87**	4.66
125	7.24	8.31	7.61	8.04	9.68	8.45	6.09	8.24	**6.07**	3.64
150	6.64	8.04	7.03	6.17	9.68	6.41	**5.03**	6.36	**5.15**	3.02
175	5.00	5.57	5.78	5.32	8.38	5.51	**4.03**	7.18	**4.15**	2.04
200	4.46	5.08	5.00	4.31	6.77	4.86	**3.39**	4.81	**3.91**	2.31

Table 8. Perturbed centroids experiment - Two centroids per class. Misclassification percentage for **VDM** similarity, perturbation probabilities $\mathbf{p}_{11} = \mathbf{p}_{12} = \mathbf{1/3}$ and $\mathbf{p}_{21} = \mathbf{p}_{22} = \mathbf{1/4}$.

The second cause contributing to the observed SDA results stems from the way the class centroids are generated. Each class centroid is generated randomly from a multivariate uniform distribution over the feature space. Thus, there is no guarantee that two centroids from the same class be more similar to each other than two centroids from different classes, that is there is no guarantee that $s(c_{1i}, c_{1j}) < s(c_{1i}, c_{2j})$ for $i, j = 1, 2$. On the contrary, on average over many draws from the sample space, the centroids are equally similar, and consequently the samples generated as perturbations of c_{12}, c_{21}, and c_{22} are approximately equally similar to c_{11}. This amplifies the detrimental effect of the bias in the SDA model. If the condition on the similarities between centroids $s(c_{1i}, c_{1j}) < s(c_{1i}, c_{2j})$ were enforced, then even the biased SDA model would produce better classification results.

The performance of mixture SDA is comparable to that of SDA if not slightly better. For the particularly simple case of the counting similarity with $p_{21} = p_{22} = 1/30$, the mixture SDA provides an order of magnitude improvement over SDA, showing that it is able to alleviate the bias problem inherent to the single-centroid SDA. However, in all other perturbed centroids results the comparison between the performance of mixture SDA and SDA is inconclusive. For $p_{21} = p_{22} = 1/4$, the overlap between the classes overshadows any performance gains mixture SDA might obtain; for the VDM results, the advantage provided by the optimized similarity measure brings the performance of SDA and mixture SDA closer together, and thus limits the gains of mixture SDA. Given the increase in complexity of the mixture SDA classifier and its inconclusive performance advantages, for these experiments it might be more advantageous to use local classifiers such as local SDA to obtain improved performance. The results show that local SDA consistently performs very well, and with only a few exceptions outperforms SDA and mixture SDA.

Note that for the VDM similarity, SDA produces excellent classification results which are very competitive with local SDA and local NC, and consistently outperform NC. The large improvement is attributable to the fact that the VDM undergoes a training phase, performed on the training set, in which the class information is used to optimize the similarity measure for class discrimination. This training step greatly benefits the SDA classifier and yields improved classification results for all classifiers when compared to the counting similarity, which does not rely on such pre-computations.

As for the single-centroid results, nnSDA is most effective at higher values of d, when the feature space is sparsely populated by the samples. A consistently good performer is the k-NN classifier, which is very competitive with local SDA, local NC, and the PSVM when $p_{21} = p_{22} = 1/30$, and often outperforms them when $p_{21} = p_{22} = 1/4$. Using a subset of the training samples, as with CNN, negatively impacts the classification performance for all sets of simulations, consistently with the single-centroids results discussed in the previous section.

5.2 Benchmark data sets

Three benchmark data sets are used to analyze further the performance of various similarity-based classifiers: a data set of protein similarities, a data set of congressional voting records, and a data set of aural sonar similarities. The tested classifiers are the local SDA, local NC, SDA, NC, nnSDA, *k*-NN, and PSVM classifiers. The mixture SDA and CNN classifiers are not tested on these data sets, as the long time required to cross-validate their parameters does not justify their attainable performance.

The performance of the classifiers on all three benchmark data sets is evaluated as the *leave-one-out error*, as follows. One sample is set aside as the test sample, and all other $N - 1$ samples are used for training. The parameters for each classifier are cross-validated on the $N - 1$ training samples using leave-one-out cross validation. The resulting best parameters are used to train each classifier on the entire $N - 1$ training samples, and the trained classifier finally classifies the test sample. The process is repeated until all available samples are tested by the trained classifiers. For local SDA, local NC and k-NN, the neighborhood size is cross-validated on the set of possible sizes {1, 2, ... 20, 30 ... 100, 150, 200}. The PSVM parameters are cross-validated over the sets of possible values C = {1, 51, ... 951}, and ϵ = {0.1, 0.2, ... 1}. The class priors are estimated to be the empirical probability of seeing a sample from each class, with Laplace correction (Jaynes, 2003). Table 9 shows the percent leave-one-out error for each classifier evaluated on the three benchmark datasets. The data sets experiments are discussed in more detail in the following sections.

	Local SDA	Local NC	SDA	NC	nnSDA	k-NN	PSVM
Protein	8.92	37.09	29.58	41.78	11.37	20.66	NA
Voting	9.66	8.05	11.72	12.87	12.18	9.20	**6.44**
Sonar	22	14	16	26	24	18	**10**

Table 9. Percentage of leave-one-out misclassifications on the protein data set.

5.2.1 Protein data

Many bioinformatics prediction problems are formulated in terms of pairwise similarities or dissimilarities. An example is the protein data set used by (Hochreiter & Obermayer, 2006). For this data set, pairwise dissimilarity values are calculated using the evolutionary distance, which is the probability that an amino acid sequence transforms into another one (Hofmann & Buhmann, 1997). The sample space $\mathcal{B}$ is not enumerated, so classification must be done based only on the pairwise dissimilarity values. The dataset contains 213 proteins with class labels "HA" (72 samples), "HB" (72 samples), "M" (39 samples) , and "G" (30 samples). The SDA, local SDA, nearest centroid, local nearest centroid, and k-NN classifiers natively support multiclass classification problems, so they can be applied directly to this four-class experiment. The PSVM, however, is a binary classifier and cannot be applied to this multiclass data set.

Guessing that all samples were from the most prevalent class would yield a 66.2% error rate. The simple one-centroid per class model of SDA achieves half that error, and works better than the more flexible local nearest centroid classifier. Local SDA, local nearest centroid and k-NN all have the same free parameter, the neighborhood size *k*. Of these, local SDA is seen to be best suited to this problem.

5.2.2 Voting data set

The UCI voting data set (Newman et al., 1998) records the voting record of 435 members of the US House of Representatives on 16 bills. The binary classification problem is to predict each member's political party affiliation given the voting records. Each of the 16 votes is either a yes, a no, or "neither", so there are 16 features which can each take on 3 possible values. This classification problem can be treated as a similarity-based classification problem by applying a similarity function to the trinary feature space. The adopted similarity in this experiment is the counting similarity.

5.2.3 Aural sonar echoes classification

In the sonar echoes classification experiment, the data consist of 100 pairwise similarities assessed by human listeners. The listeners rated the pairwise similarities of digitized active sonar echoes from two classes { clutter or target { without knowledge of the class labels, and based their evaluation of similarity only on their perceptual judgement of how the echoes sounded similar; thus, the underlying features of similarity are inaccessible. Each listener assigned a discrete similarity value between 1 and 5 to each pair of echoes; each pair was rated by two different listeners, and the two assigned similarity scores were added, so that the range of possible values for the similarity is [2, 10]. The target and clutter classes are equally likely, each one containing 50 echoes. This set of echoes is particularly difficult to classify in that metric-space classifiers produced incorrect results. Further details on this data set are in (Philips et al., 2006).

6. Summary

The chapter introduced a new framework for classification that is both *similarity-based* and *generative*: *similarity discriminant analysis*, or *SDA*. The experimental results show that the classifiers resulting from the proposed SDA framework have practical advantages in terms of performance, interpretability, and ease of use. SDA is *similarity-based* in that it classifies samples based on their pairwise similarities and does not require that the samples be described by numerical feature vectors, the standard sample description method in metric learning. SDA is *generative*, in that it estimates probabilistic models based on descriptive statistics of the classes. Having access to probability estimates is important. A probabilistic framework seamlessly accommodates multi-class classifiers, asymmetric misclassification costs, and class priors. Furthermore, probability estimates are easily fused into into larger systems, and can be used to identify abnormal samples that have low probability of any class. The generative models in the SDA family are solutions to constrained maximum entropy problems where the constraints are placed on the mean values of the similarity-based descriptive statistics. As dictated by the principle of maximum entropy, the resulting generative class models are exponential functions of the similarity statistics.

Di®erent choices for the descriptive statistics lead to different SDA classifiers. This chapter focused on the centroid-based SDA classifiers: each class is described by a prototypical sample, a *centroid*, and the generative models are based on the similarities of the samples to each class centroid. SDA accommodates various definitions of centroid; this chapter focused on the maximum-sum-similarity centroid. The nearest neighbor similarity is also explored as a descriptive statistic, yielding the nnSDA classifier.

As with LDA and QDA, the power of the SDA generative classifier depends on how well its model matches the true class-conditional distributions. A mismatched model will be biased and produce erroneous classifications. The centroid-based SDA classifier is a good match for single-centroid distributions of objects, but is a biased model for multi-centroidal distributions. This chapter proposes *local SDA* and *mixture SDA* as similarity-based generative classifiers with reduced bias that can be used for multimodal distributions. Local SDA is the SDA classifier applied to a local neighborhood of a test sample. A local class centroid can be viewed as a representative prototype for the class in the neighborhood of a test sample and the class-conditional models provide an estimate of the local distribution of the similarities to the local centroid. Local SDA was shown to be a Bayes error-consistent classifier and is the first classifier to be similarity-based, generative, *and* local. Mixture SDA builds on the metric-learning mixture models by modeling each class as a linear combination of several single-centroid SDA models. The parameters for the mixture SDA classifier can be estimated with the EM algorithm.

The family of SDA classifiers is very competitive with, and often outperforms, their corresponding non-generative similarity-based classifier. SDA competes with nearest centroid; local SDA competes with local NC. The SDA classifiers are also competitive with the PSVM, the state-of-the-art support vector machine for similarity-based classification. The PSVM bases its classification on the entire training set of pairwise similarities. This requires enumeration of size $N \times N$ similarity matrices, thus posing computational challenges for large data sets. Furthermore, PSVM is a non-generative, intrinsically binary classifier: it is di±cult to view it in a probabilistic framework where there are more than two possible classes for the data samples. The SDA classifiers remain competitive while relying on more parsimonious representations of the underlying similarity relationships between the samples. Furthermore, the generative quality of the SDA family of classifiers provides intuitive information about the similarity characteristics of the data. The SDA-generated probability estimates are useful for interpreting the results in a probabilistic framework, and allow for class priors and costs to be seamlessly integrated into the classification rules.

7. References

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(4): 509-522, April 2002.

M. Bicego, V. Murino, M. Pelillo, and A. Torsello. Special issue on similarity-based classification. *Pattern Recognition*, 39, October 2006.

L. Cazzanti and M. R. Gupta. Local similarity discriminant analysis. In *Intl. Conf.on Machine Learning (ICML)*, 2007.

L. Cazzanti and M. R. Gupta. Information-theoretic and set-theoretic similarity. In *Proc. of the IEEE Intl. Symposium on Information Theory*, pages 1836-1840, 2006.

L. Cazzanti, M. R. Gupta, and A. J. Koppal. Generative models for similarity-based classification. *Pattern Recognition*, 41, number = 7, pages = 2289-2297, YEAR = 2008,.

S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57-78, 1993.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, 1991.

L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag Inc., New York, 1996.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

B. S. Everitt and S. Rabe-Hesketh. *The Analysis of Proximity Data*. Arnold, London, 1997.

I. Gati and A. Tversky. Weighting common and distinctive features in perceptual and conceptual judgments. *Cognitive Psychology*, (16):341-370, 1984.

M. R. Gupta, L. Cazzanti, and A. J. Koppal. Maximum entropy generative models for similarity-based learning. In *Proc. IEEE Intl. Symposium on Information Theory*, 2007.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.

S. Hochreiter and K. Obermayer. Support vector machines for dyadic data. *Neural Computation*, 18(6):1472-1510, 2006.

T. Hofmann and J.M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(1), January 1997.

D. W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(6):583-600, June 2000.

E. T. Jaynes. On the rationale for maximum entropy methods. *Proc. of the IEEE*, 70(9):939{952, September 1982.

E. T. Jaynes. *Probability theory: the logic of science.* Cambridge University Press, 2003.

M. I. Jordan. An Introduction to Probabilistic Graphical Models. To be published, 20xx.

W. Lam, C. Keung, and D. Liu. Discovering useful concept prototypes for classification based on filtering and abstraction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(8):1075-1090, August 2002.

D. Lin. An information-theoretic definition of similarity. *Proc. of the Intl. Conf. on Machine Learning*, 1998.

M. Lozano, J. M. Sotoca, J. S. Sánchez, F. Pla, E. Pekalska, and R. P. W. Duin. Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. *Pattern Recognition*, 39:1827-1838, 2006.

MATLAB: The Language of Technical Computing. The MathWorks, Natick, MA, 2006 edition.

Y. Mitani and Y. Hamamoto. Classifier design based on the use of nearest neighbor samples. *Proc. of the Intl. Conf. on Pattern Recognition*, pages 769-772, 2000.

Y. Mitani and Y. Hamamoto. A local mean-based nonparametric classifier. *Pattern Recognition Letters*, 27(10):1151-1159, July 2006.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

E. Pekalska, P. Paclíc, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, pages 175-211, 2001.

E. Pekalska, R. P. W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition Letters*, 39:189-208, 2006.

S. Philips, J. Pitton, and L. Atlas. Perceptual feature identification for active sonar echoes. In *IEEE OCEANS*, 2006.

D. A. Reynolds and R. C. Rose. Robust text-independent speaker-identification using Gaussian mixture speaker models. *IEEE Trans. on Speech and Audio Processing*, 3(1), 1995.

S. Santini and R. Jain. Similarity measures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(9):871-883, September 1999.

S. Sattath and A. Tversky. On the relation between common and distinctive feature models. *Psychological Review*, (94):16-22, 1987.

G. Schwartz and A. Tversky. On the reciprocity of proximity relations. *Journal of Mathematical Psychology*, 22(3):301-307, September 1980.

P. Simard, Y. Le Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems 5*, pages 50-68, 1993.

C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213-1228, December 1986.

A. Tversky. Features of similarity. *Psychological Review*, (84):327-352, 1977.

A. Tversky and I. Gati. Studies of similarity. In E. Rosch and B. Lloyd, editors, *Cognition and Categorization*. Earlbaum, Hillsdale, N.J., 1978.

A. Tversky and J. W. Hutchinson. Nearest neighbor analysis of psychological spaces. *Psychological Review*, 93:3-22, 1986.

D. Weinshall, D. W. Jacobs, and Y. Gdalyahu. Classification in non-metric spaces. *Advances in Neural Information Processing Systems 11*, pages 838-844, 1999.

D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1-34, 1997.

H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: discriminative nearest neighbor classification for visual category recognition. *Proc. of the IEEE Conf. o Computer Vision and Pattern Recognition*, pages 2126 - 2136, 2006.

Forced Information for Information-Theoretic Competitive Learning

Ryotaro Kamimura
IT Education Center, Information Technology Center, Tokai University, Japan

1. Introduction

We have proposed a new information-theoretic approach to competitive learning [1], [2], [3], [4], [5]. The information-theoretic method is a very flexible type of competitive learning, compared with conventional competitive learning. However, some problems have been pointed out concerning the information-theoretic method, for example, slow convergence. In this paper, we propose a new computational method to accelerate a process of information maximization. In addition, an information loss is introduced to detect the salient features of input patterns.

Competitive learning is one of the most important techniques in neural networks with many problems such as the dead neuron problem [6], [7]. Thus, many methods have been proposed to solve those problems, for example, conscience learning [8], frequency-sensitive learning [9], rival penalized competitive learning [10], lotto-type competitive learning [11] and entropy maximization [12]. We have so far developed information-theoretic competitive learning to solve those fundamental problems of competitive learning. In the information-theoretic learning, no dead neurons can be produced, because entropy of competitive units must be maximized. In addition, experimental results have shown that final connection weights are relatively independent of initial conditions.

However, one of the major problems is that it is sometimes slow in increasing information. As a problem becomes more complex, heavier computation is needed. Without solving this problem, it is impossible for the information-theoretic method to be applied to practical problems. To overcome this problem, we propose a new type of computational method to accelerate a process of information maximization. In this method, information is supposed to be maximized or sufficiently high at the beginning of learning. This supposed maximum information forces networks to converge to stable points very rapidly. This supposed maximum information is obtained by using the ordinary winner-take-all algorithm. Thus, this method is one in which the winter-takeall is combined with a process of information maximization.

We also present a new approach to detect the importance of a given variable, that is, information loss. Information loss is difference between information with all variables and information without a variable, and is used to represent the importance of a given variable. Forced information with information loss can be used to extract main features of input patterns. Connection weights can be interpreted as the main characteristics of classified groups. On the other hand, information loss is used to extract the features on which input

patterns or groups are classified. Thus, forced information and information loss has a possibility to show clearly main features of input patterns.
In Section 2, we present how to compute forced information as well as how to compute information loss. In Section 3 and 4, we present experimental results on a simple symmetric and Senate problem to show that one epoch is enough to reach stable points. In Section 5, we present experimental results on a student survey. In this section, we try to show that learning is accelerated more than sixty times faster, and explicit representations can be obtained.

2. Information maximization

We consider information content stored in competitive unit activation patterns. For this purpose, let us define information to be stored in a neural system. Information stored in a system is represented by decrease in uncertainty [13]. Uncertainty decrease, that is, information *I*, is defined by

$$I = -\sum_{\forall j} p(j) \log p(j) + \sum_{\forall s} \sum_{\forall j} p(s) p(j \mid s) \log p(j \mid s), \tag{1}$$

where *p*(*j*), *p*(*s*) and *p*(*j* | *s*) denote the probability of firing of the *j*th unit, the probability of the *s*th input pattern and the conditional probability of the *j*th unit, given the *s*th input pattern, respectively. When the conditional probability *p*(*j* | *s*) is independent of the occurrence of the *s*th input pattern, that is, *p*(*j* | *s*) = *p*(*j*), mutual information becomes zero.

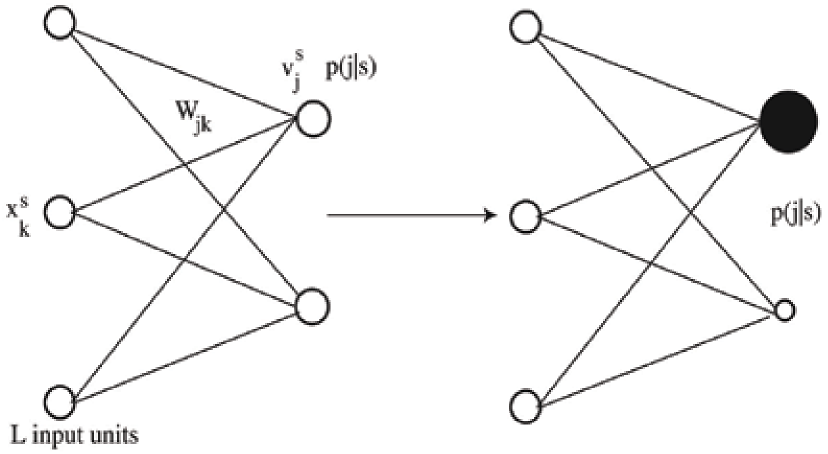


Fig. 1. A single-layered network architecture for information maximization.

Let us present update rules to maximize information content. As shown in Figure 2, a network is composed of input units x_k^s and competitive units v_j^s. We used as the output function the inverse of the square of the Euclidean distance between connection weights and outputs for facilitating the derivation. Thus, distance is defined by

$$d_j^s = \sum_{k=1}^{L} (x_k^s - w_{jk})^2. \tag{2}$$

An output from the *j*th competitive unit can be computed by

$$v_j^s = \frac{1}{d_j^s}, \tag{3}$$

where *L* is the number of input units, and w_{jk} denote connections from the *k*th input unit to the *j*th competitive unit. The output is increased as connection weights are closer to input patterns.

The conditional probability $p(j \mid s)$ is computed by

$$p(j \mid s) = \frac{v_j^s}{\sum_{m=1}^{M} v_m^s}, \tag{4}$$

where *M* denotes the number of competitive units. Since input patterns are supposed to be uniformly given to networks, the probability of the *j*th competitive unit is computed by

$$p(j) = \frac{1}{S}\sum_{s=1}^{S} p(j \mid s). \tag{5}$$

Information *I* is computed by

$$I = -\sum_{j=1}^{M} p(j)\log p(j) + \frac{1}{S}\sum_{s=1}^{S}\sum_{j=1}^{M} p(j \mid s)\log p(j \mid s). \tag{6}$$

Differentiating information with respect to input-competitive connections w_{jk}, we have

$$\begin{aligned} \Delta w_{jk} = & -\beta \sum_{s=1}^{S}\left(\log p(j) - \sum_{m=1}^{M} p(m \mid s)\log p(m)\right) Q_{jk}^s \\ & +\beta \sum_{s=1}^{S}\left(\log p(j \mid s) - \sum_{m=1}^{M} p(m \mid s)\log p(m \mid s)\right) Q_{jk}^s, \end{aligned} \tag{7}$$

where β is the learning parameter, and

$$Q_{jk}^s = \frac{1}{S} p(j \mid s) v_j^s (x_k^s - w_{jk}). \tag{8}$$

3. Maximum information-forced learning

One of the major shortcomings of information-theoretic competitive learning is that it is sometimes very slow in increasing information content to a sufficiently large level. We here present how to accelerate learning by supposing that information is already maximized before learning. Thus, we have a conditional probability $p(j \mid s)$ such that the probability is set to ϵ for a winner, and $1 - \epsilon$ for all the other units. We here suppose that ϵ ranges between zero and unity. For example, supposing that information is almost maximized with two competitive units, and this means that a conditional probability is close to unity, and all the other probabilities are close to zero. Thus, we should take the parameter ϵ as a value close to unity, say, 0.9. In this case, all the other cases are set to 0.1. Weights are updated so as to maximize usual information content. The conditional probability $p(j \mid s)$ is computed by

$$p(j \mid s) = \frac{v_j^s}{\sum_{m=1}^{M} v_m^s}, \tag{9}$$

where M denotes the number of competitive units.

$$p(j \mid s) = \begin{cases} \epsilon & \text{for a winner} \\ \frac{1-\epsilon}{M-1} & \text{otherwise} \end{cases} \tag{10}$$

At this place, we suppose that information is already close to a maximum value. This means that if the jth unit is a winner, the probability of the jth unit should be as large as possible, and close to unity, while all the other units' firing rates should be as small as possible.

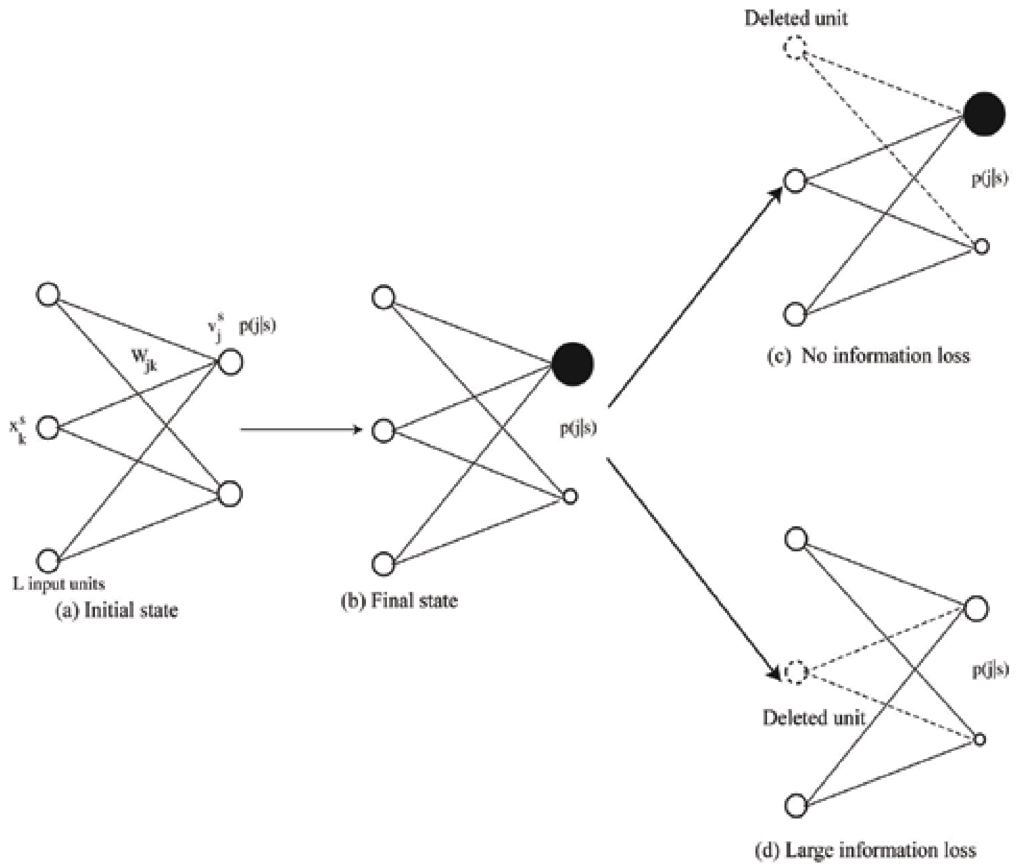


Fig. 2. A single-layered network architecture for information maximization.

This forced information is a method to include the winner-take-all algorithm inside information maximization. As already mentioned, the winner-take-all is a realization of forced information maximization, because information is supposed to be maximized.

4. Information loss

We now define information when a neuron is damaged by some reasons. In this case, distance without the *m*th unit is defined by

$$d_{jm}^s = \sum_{k \neq m} (x_k^s - w_{jk})^2, \tag{11}$$

where summation is over all input units except the *m*th unit. The output without the *m*th unit is defined by

$$v_{jm}^s = \frac{1}{d_{jm}^s}. \tag{12}$$

The normalized output is computed by

$$p^m(j \mid s) = \frac{v_{jm}^s}{\sum_{l=1}^M v_{lm}^s}. \tag{13}$$

Now, let us define mutual information without the *m*th input unit by

$$I_m = -\sum_{j=1}^M p^m(j) \log p^m(j) + \sum_{s=1}^S \sum_{j=1}^M p(s) p^m(j \mid s) \log p^m(j \mid s), \tag{14}$$

where p_m and $p_m(j|s)$ denote a probability and a conditional probability, given the *s*th input pattern. Information loss is defined by difference between original mutual information with full units and connections and mutual information without a unit. Thus, we have information loss

$$IL_m = I - I_m. \tag{15}$$

For each competitive unit, we compute conditional mutual information for each competitive unit.
For this, we transform mutual information as follows.

$$I = \sum_{s=1}^S \sum_{j=1}^M p(s) p(j \mid s) \log \frac{p(j \mid s)}{p(j)}. \tag{16}$$

Conditional mutual information for each competitive unit is defined by

$$I_j = \sum_{s=1}^S p(s) p(j \mid s) \log \frac{p(j \mid s)}{p(j)}. \tag{17}$$

Thus, conditional information loss is defined by

$$IL_{jm} = I_j - I_{jm} \tag{18}$$

We have the following relation:

$$IL_m = \sum_{j=1}^{M} IL_{jm} \tag{19}$$

5. Experiment No.1: symmetric data

In this experiment, we try to show that symmetric data can easily be classified by forced information. Figure 3 shows a network architecture where six input patterns are given into input units. These input patterns can naturally be classified into two classes. Figure 4 shows

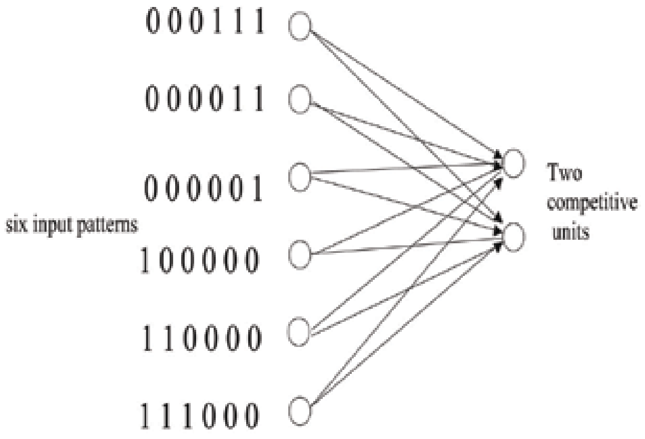


Fig. 3. A network architecture for the artificial data.

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Party	D	D	D	D	D	D	D	D	R	R	R	R	R	R	R
1	0	1	0	1	1	1	1	0.5	0	0	1	1	1	1	0
2	0.5	0	0	0	0	0	0	0	1	0	0	0	1	0	1
3	0	0.5	1	1	1	1	1	1	1	0.5	1	1	0.5	1	1
4	1	0.5	1	1	1	1	0	0	0	0	0	0	0	1	0
5	1	0.5	1	1	1	1	1	0	0	0.5	0	0	0	1	1
6	1	0.5	0	0	0	0	0	0	1	0.5	0	1	1	0	1
7	1	1	1	0.5	1	1	1	1	0	0	1	1	1	1	1
8	0	0	0.5	0	0	0	0	0	1	1	1	0	0	0	1
9	1	1	0.5	1	1	1	0.5	1	0	0	0	0	0	1	1
10	1	1	1	1	1	1	0	1	0	0	0	0	0	1	0
11	1	1	1	1	1	1	0	1	0	0	1	0	0	1	0
12	1	0.5	1	1	1	1	1	1	0	0.5	0	0	0	1	0
13	0	1	0	0	0	0	0	1	1	0.5	0.5	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0.5	0	0	1	0	0
15	0	0	0	0	0	0	0	0	1	0.5	1	0	1	0	1
16	0	0.5	0	0	0.5	0	0.5	0	0.5	0.5	1	1	1	0	1
17	0	0	0	0	0	0	0	0	0.5	0.5	1	0	1	0	0
18	1	1	1	1	1	1	1	1	1	1	0.5	0	1	1	0
19	1	1	1	1	0	0	0	1	0	0	0.5	0	0	1	0

Table 1: U.S. congressmen by their voting attitude on 19 environmental bills. The first 8 congressmen are Republicans, while the latter 7 (from 9 to 15) congressmen are Democrats. In the table, 1, 0 and 0.5 represent *yes*, *no* and undecided, respectively.

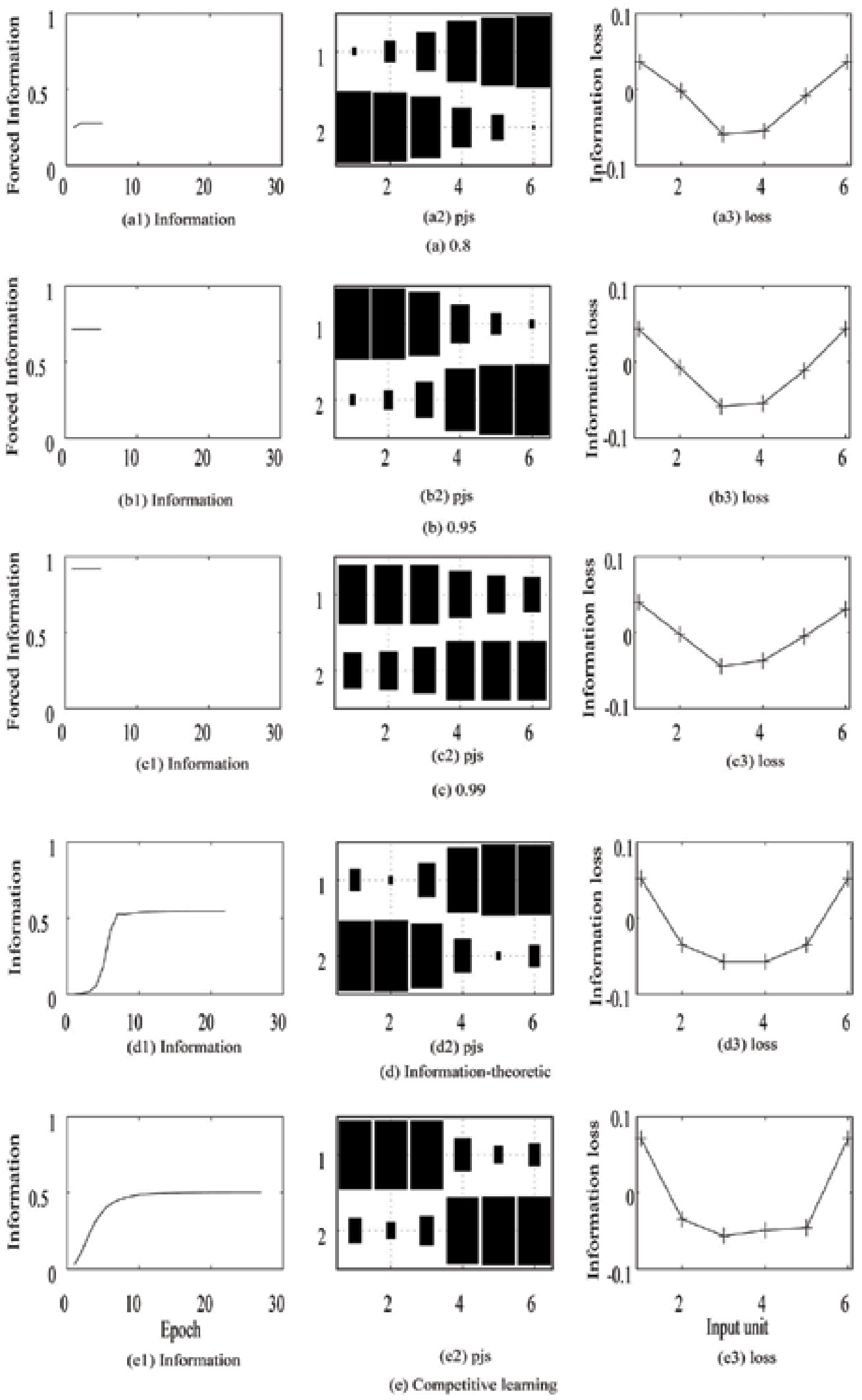


Fig. 4. Information, forced information, probabilities and information losses for the artificial data.

information, forced information, probabilities and information losses for the symmetric data. When the constant ϵ is set to 0.8, information reaches a stable point with eight epochs. When the constant is increased to 0.95, just one epoch is enough to reach that point. However, when information is further increased to 0.99, information reaches easily a stable point, but obtained probabilities show rather ambiguous patterns. Compared with forced information, information-theoretic learning needs more than 20 epochs and as many as 30 epochs are needed by competitive learning. We could obtain almost same probabilities $p(j|s)$ except ϵ = 0.99. For the information loss, the first and the sixth input patterns show large information loss, that is, important. This represents quite well symmetric input patterns.

6. Experiment No.2: senate problem

Table 1 shows the data of U.S. congressmen by their voting attitude on 19 environmental bills ??. The first 8 congressmen are Republicans, while the latter 7 (from 9 to 15) congressmen are Democrats. In the table, 1, 0 and 0.5 represent *yes*, *no* and undecided. Figure 5 shows information, forced information and information loss for the senate problem. When the constant ϵ is set to 0.8, information reaches a stable point with eight epochs. When the constant is increased to 0.95, just one epoch is enough to reach that point. However, when information is further increased to 0.99, obtained probabilities show rather ambiguous patterns. Compared with forced information, information-theoretic learning needs more than 25 epochs and as many as 15 epochs are needed by competitive learning. In addition, in almost all cases, the information loss shows the same pattern. The tenth, eleventh and twelfth input unit take large losses, meaning that these units play very important roles in learning. By examining Table 1, we can see that these units surely divide input patterns into two classes. Thus, the information captures the features in input patterns quite well.

7. Experiment 3: student survey

7.1 Two groups analysis

In the third experiment, we report an experimental result on a student survey. We did student survey about what subjects they are interested in. The number of students was 580, and the number of variables (questionnaires) was 58. Figure 6 shows a network architecture with two competitive units. The number of input units is 58 units, corresponding to 58 items such as *computer*, *internet* and so on. The students must respond to these items with four scales.

In the previous information-theoretic model, when the number of competitive units is large, it is sometimes impossible to attain the appropriate level of information. Figure 7 shows information as a function of the number of epochs. By using simple information maximization, we need as many as 500 epochs to be stabilized. On the other hand, by forced information, we need just eight epochs to finish learning. Almost same representations could be obtained. Thus, we can say that forced information maximization can accelerate learning almost seven times faster than the ordinary information maximization.

Figure 8 shows connection weights for two groups analysis. The first group represents a group with a higher interest in the items. The numbers of students in these groups are 256 and 324.

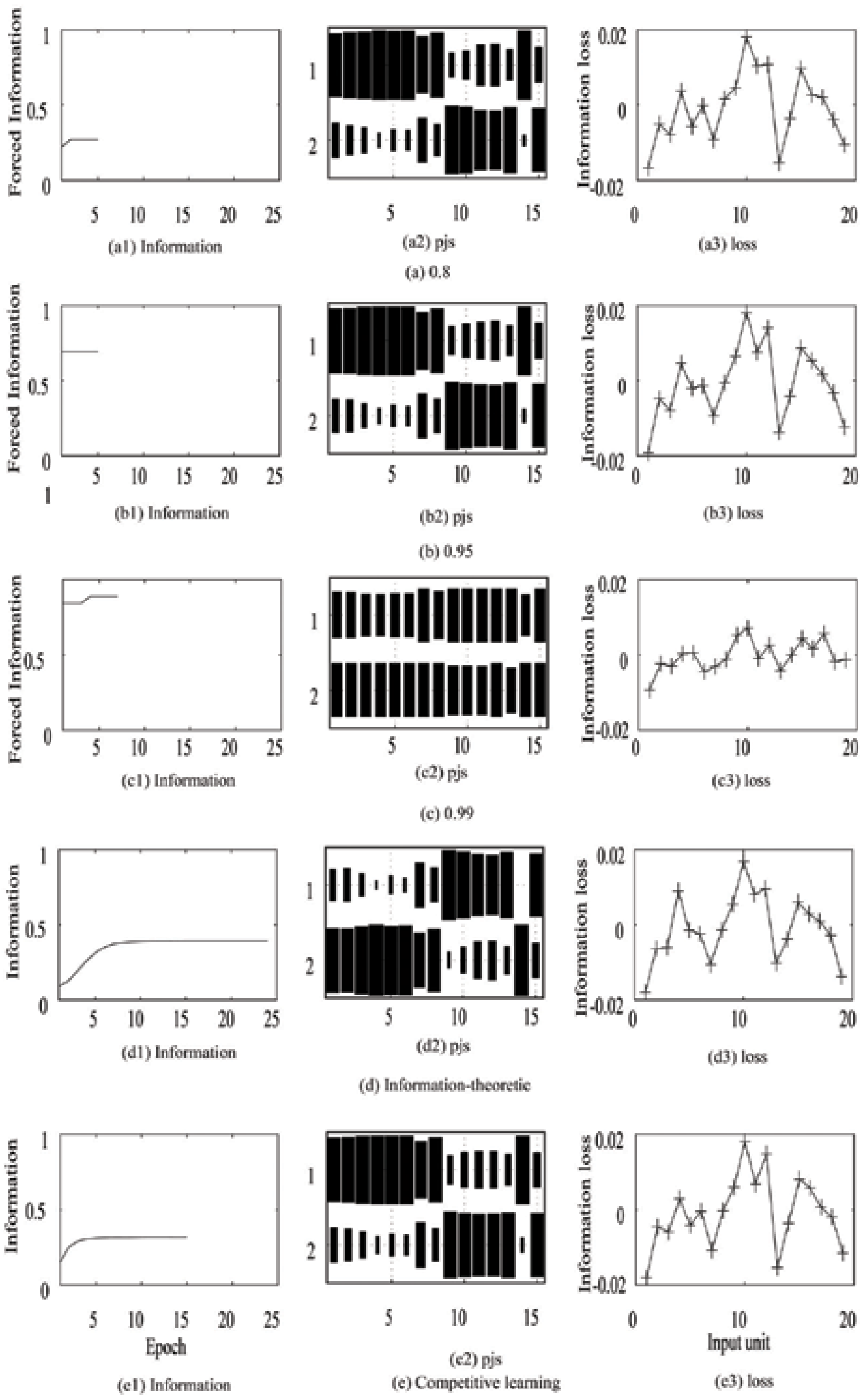


Fig. 5. Information, forced information, probabilities and information loss for the senate problem.

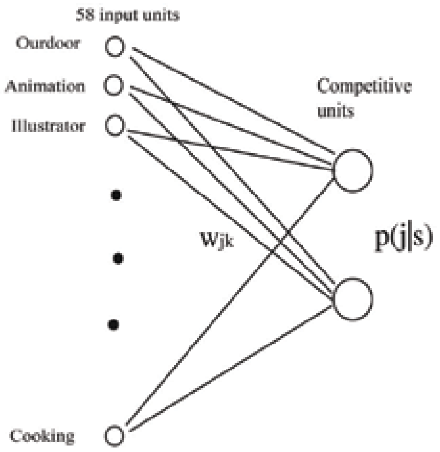


Fig. 6. Network architecture for a student analysis.

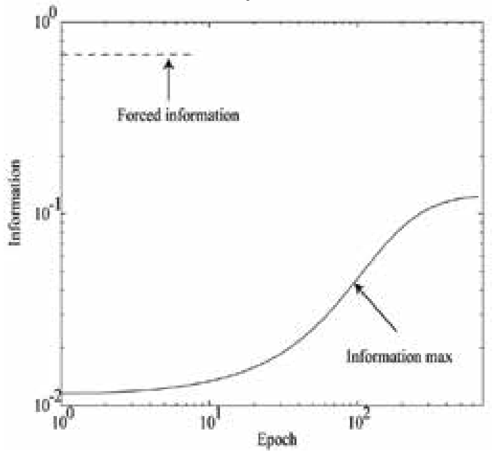


Fig. 7. Information and forced information as a function of the number of epochs by information-theoretic and forced-information method.

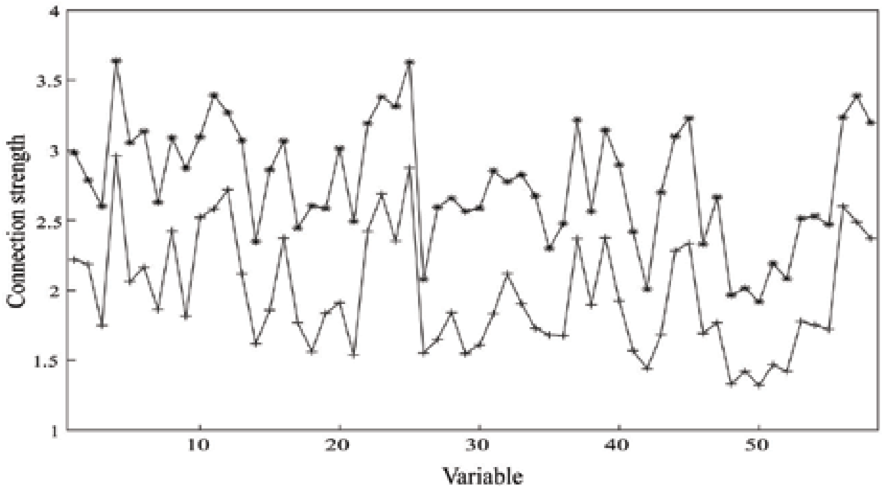


Fig. 8. Connection weights for two groups analysis.

This means that the method can classify 580 students by the magnitude of connection weights. Because connection weights try to imitate input patterns directly, we can see that two competitive units show students with high interest and low interest in the items in the questionnaire.
Table 2 represents the ranking of items for a group with a high interest in the items. As can be seen in the table, students respond highly to *internet* and *computer*, because we did this survey for the classes of information technology. Except these items, the majority is related to the so-called *entertainment* such as *music*, *travel*, *movie*. In addition, these students have some interest in *human relations* as well as *qualification*. On the other hand, these students have little interest in traditional and academic sciences such as *physics* and *mathematics*. Table 3 represents the ranking of items for a group with a low interest in the items. Except the difference of the strength, this group is similar to the first group. That is, students in this gropup respond highly to *internet* and *computer*, and they have keen interest in *entertainment*. On the other hand, these students have little interest in traditional and academic sciences such as *physics* and *mathematics*. Table 4 shows the information loss for the two groups. As can be seen in the table, two groups are separated by items such as *multimedia*, *business*. Especially, many terms concerning *business* appear in the table. This means that two groups are separated mainly based upon *business*. The most important thing to differentiate two groups is whether students have some interest in *buisiness* or *multimedia*. Let us see what the information loss represents in actual cases. Figure 9 shows the information loss (a) and difference between two connection weights (b). As can be seen in the figure, two figures are quite similar to each other. Only difference is the magnitude of two measures. Table 5 shows the ranking of items by difference between two connection weights. As can be seen in the table, the items in the list is quite similar to those in information loss. This means that the information loss in this case is based upon difference between two connection weights.

No.	No.(Figure)	Strength	Item
1	4	3.639	Internet
2	25	3.630	Music
3	11	3.393	Computer
4	57	3.389	Travel
5	23	3.381	Movie
6	24	3.314	Visual media
7	12	3.269	Sport
8	56	3.236	Comic
9	45	3.229	Human relations
10	37	3.217	Qualification
49	14	2.347	Trading
50	46	2.327	Mathematics
51	35	2.299	Archeology
52	51	2.193	Statistics
53	52	2.083	Physics
54	26	2.078	Chemistry
55	49	2.015	Earth science
56	42	2.009	Craftwork
57	48	1.966	Shipping
58	50	1.918	Railroad

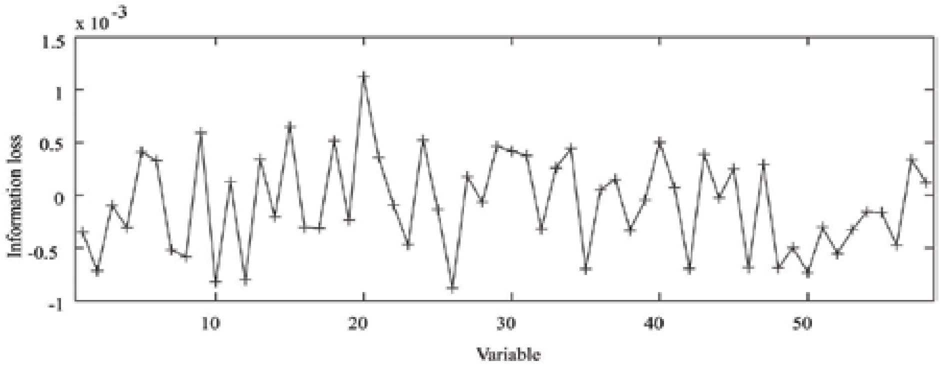
Table 2. Ranking of items for a group of students who responded to items with a low level of interest.

No.	No.(Figure)	Strength	Item
1	4	2.959	Internet
2	25	2.874	Music
3	12	2.717	Sport
4	23	2.687	Movie
5	56	2.599	Comic
6	11	2.580	Computer
7	10	2.519	Game
8	57	2.486	Travel
9	8	2.423	Entertainment
10	22	2.422	Eating and drinking
49	18	1.561	Marketing
50	26	1.552	Chemistry
51	29	1.549	Management sciences
52	21	1.539	Exchange
53	51	1.468	Statistics
54	42	1.440	Craftwork
55	52	1.421	Physics
56	49	1.421	Earth science
57	48	1.331	Shipping
58	50	1.321	Railroad

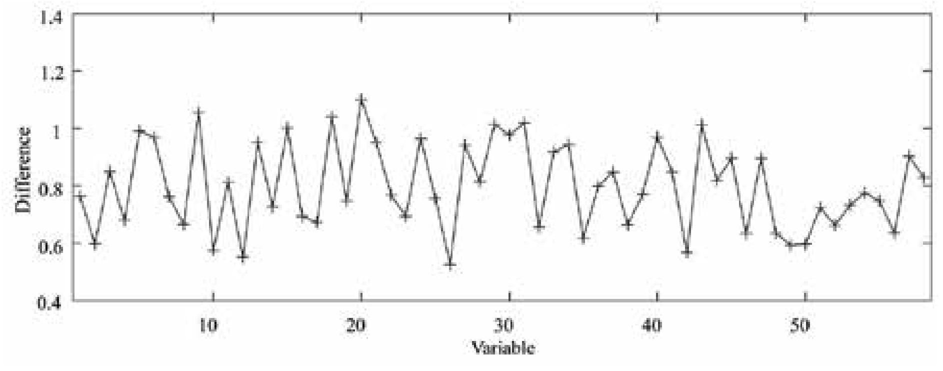
Table 3. Ranking of items for a group of students who responded to items with a low level of interest.

No.	No.(Figure)	Strength	Item
1	20	1.125	Multimedia
2	15	0.649	Buisiness
3	9	0.594	Creator
4	24	0.524	Visual Media
5	18	0.516	Marketing
6	40	0.501	Photograph
7	29	0.464	Business management
8	34	0.444	Publicity
9	30	0.420	Economics
10	5	0.410	Internet business
49	8	-0.581	Entertainment
50	46	-0.687	Mathematics
51	48	-0.690	Shipping
52	42	-0.695	Craftwork
53	35	-0.698	Archeology
54	2	-0.714	Animation
55	50	-0.732	Railroad
56	12	-0.802	Sport
57	10	-0.818	Game
58	26	-0.880	Chemistry

Table 4. Ranking of information loss for two groups analysis ($\times 10^{-3}$).



(a) Information loss



(b) Difference between two connection weights

Fig. 9. Information loss (a) and difference between two connection weights ($w2k - w1k$) (b).

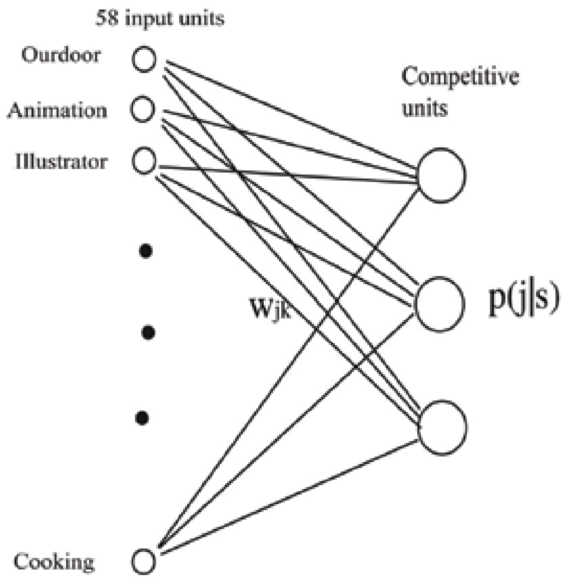


Fig. 10. Network architecture for three groups analysis.

No.	No.(Figure)	Strength	Item
1	20	1.100	Multimedia
2	9	1.055	Creator
3	18	1.040	Marketing
4	31	1.020	Arts
5	29	1.014	Business management
6	43	1.014	Information sciences
7	15	1.001	Business
8	5	0.991	Internet business
9	30	0.977	Economics
10	40	0.971	Photograph
49	48	0.636	Shipping
50	46	0.634	Mathematics
51	35	0.618	Archeology
52	2	0.598	Animation
53	50	0.597	Railroad
54	49	0.595	Earth science
55	10	0.575	Game
56	42	0.569	Craftwork
57	12	0.552	Sport
58	26	0.526	Chemistry

Table 5. Difference between two groups of students.

7.2 Three groups analysis

We increase the number of competitive units from two to three units as shown in Figure 10. Figure 11 shows connection weights for three groups. The third group detected at this time shows the lowest values of connection weights. The numbers of the first, the second and the third groups are 216, 341 and 23. Thus, the third group represents only a fraction of the data. Table 6 shows connection weights for students with strong interest in the items. Similar to a case with two groups, we can see that students have much interest in *entertainment*. Table 7 shows connection weights with moderate interest in the items. In the list, *qualification* and *human relations* disappear, and all the items expcet *computer* and *internet* are related to *entertainment*. Table 8 shows connection weights for the third group with low interest in the items. Though the scores are much lower than the other groups, this group also shows keen interest in *entertainment*. Table 9 shows conditional information losses for the first competitive unit. Table 10 shows information losses for the second competitive unit. Both tables show the same patterns of items in which business-related terms such as *economics*, *stock* show high values of information losses. Table 11shows a table of items for the third competitive units. Though the strength of information losses is small, more practical items such as *cooking* are detected.

7.3 Results by the principal component analysis

Figure 12 shows the contribution rates of principal components. As can be seen in the figure, the first principal component play a very important role in this case. Thus, we interpret the first principal component. Table 12 shows the ranking of items for the first principal component.

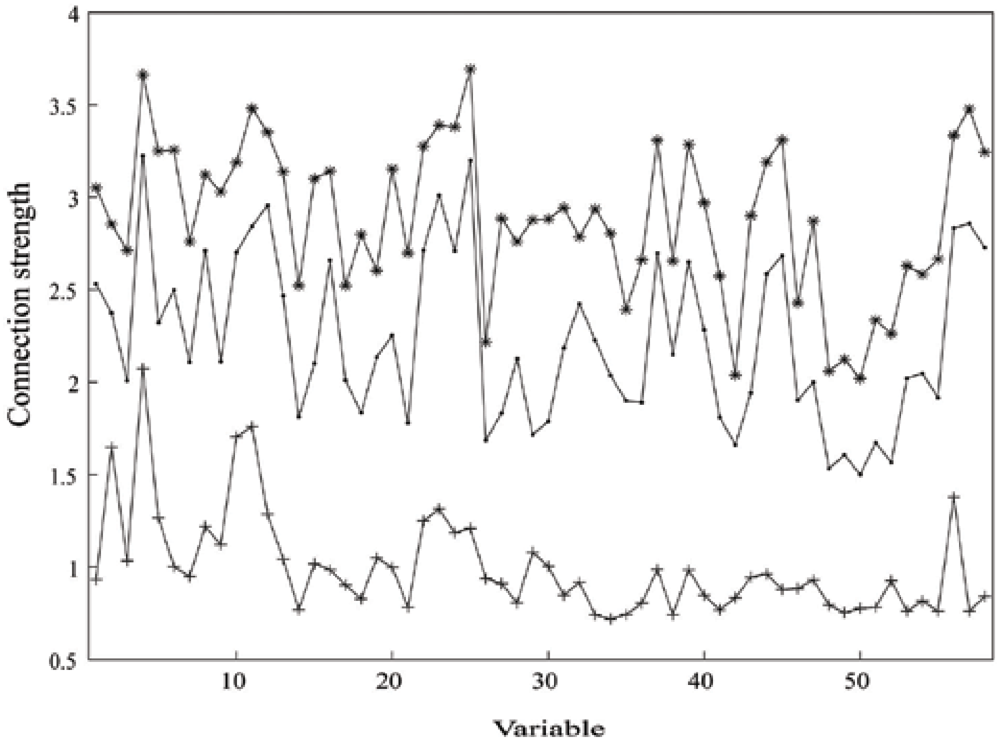


Fig. 11. Connection weights for three group analysis.

No.	No.(Figure)	Strength	Item
1	25	3.694	Music
2	4	3.663	Internet
3	11	3.481	Computer
4	57	3.478	Travel
5	23	3.390	Movie
6	24	3.379	Visual media
7	12	3.352	Sport
8	56	3.334	Comic
9	45	3.311	Human relations
10	37	3.309	Qualification
49	17	2.520	Volentier
50	46	2.427	Mathematics
51	35	2.391	Archeology
52	51	2.335	Statistics
53	52	2.261	Physics
54	26	2.216	Chemistry
55	49	2.122	Earth science
56	48	2.059	Shipping
57	42	2.037	Craftwork
58	50	2.019	Railroad

Table 6. Connection weights for students with strong interest in those items.

No.	No.(Figure)	Strength	Item
1	4	3.224	Internet
2	25	3.200	Music
3	23	3.012	Movie
4	12	2.956	Sport
5	57	2.856	Travel
6	11	2.843	Computer
7	56	2.834	Comic
8	58	2.727	Cooking
9	22	2.713	Eating and drinking
10	8	2.710	Entertainment
49	30	1.789	Economics
50	21	1.781	Exchange
51	29	1.717	Business management
52	26	1.687	Chemistry
53	51	1.674	Statistics
54	42	1.661	Craftwork
55	49	1.607	Earth science
56	52	1.566	Physics
57	48	1.533	Shipping
58	50	1.501	Railroad

Table 7. Connection weights for students with moderate interest in those items.

No.	No.(Figure)	Strength	Item
1	4	2.071	Internet
2	11	1.760	Computer
3	10	1.705	Game
4	2	1.648	Animation
5	56	1.377	Comic
6	23	1.314	Movie
7	12	1.286	Sport
8	5	1.267	Internet
9	22	1.250	Eating and drinking
10	8	1.219	Entertainment
49	14	0.770	Trading
50	41	0.769	Sociology
51	57	0.762	Travel
52	53	0.762	Literature
53	55	0.762	Law
54	49	0.754	Earth science
55	35	0.743	Archeology
56	33	0.741	Language
57	38	0.741	Bicycle
58	34	0.718	Publicity

Table 8. Connection weights for students with low interest in those items.

No.	No.(Figure)	Strength	Item
1	29	0.559	Business management
2	30	0.541	Economics
3	15	0.538	Business
4	20	0.510	Multimedia
5	27	0.445	Stock
6	18	0.400	Marketing
7	21	0.377	Exchange
8	9	0.334	Creator
9	43	0.320	Information sciences
10	47	0.314	Politics
49	56	-0.333	Comic
50	35	-0.353	Archeology
51	26	-0.355	Chemistry
52	4	-0.362	Internet business
53	42	-0.397	Craftwork
54	2	-0.422	Animation
55	8	-0.428	Entertainment
56	10	-0.429	Game
57	12	-0.435	Sport
58	23	-0.446	Movie

Table 9. Information loss No.1($\times 10^{-3}$).

No.	No.(Figure)	Strength	Item
1	30	0.335	Economics
2	15	0.317	Business
3	20	0.271	Multimedia
4	29	0.270	Business management
5	27	0.266	Stock
6	21	0.228	Exchange
7	18	0.227	Marketing
8	47	0.213	Politics
9	43	0.186	Information sciences
10	9	0.162	Creator
49	48	-0.226	Shipping
50	23	-0.229	Movie
51	50	-0.241	Railroad
52	26	-0.258	Chemistry
53	12	-0.271	Sport
54	42	-0.272	Craftwork
55	8	-0.289	Entertainment
56	10	-0.329	Game
57	2	-0.336	Animation
58	4	-0.392	Internet

Table 10. Information loss No.2($\times 10^{-3}$).

No.	No.(Figure)	Strength	Item
1	57	0.384	Travel
2	45	0.258	Human relations
3	34	0.249	Publicity
4	58	0.235	Cooking
5	33	0.211	Language
6	25	0.192	Music
7	44	0.176	Psychology
8	37	0.175	Qualification
9	40	0.175	Photograph
10	20	0.161	Multimedia
49	52	-0.218	Physics
50	3	-0.223	Illustrator
51	56	-0.234	Comic
52	26	-0.243	Chemistry
53	42	-0.268	Craftwork
54	29	-0.382	Business management
55	11	-0.447	Computer
56	4	-0.466	Internet business
57	10	-0.493	Game
58	2	-0.569	Animation

Table 11. Information loss No.3($\times 10^{-3}$).

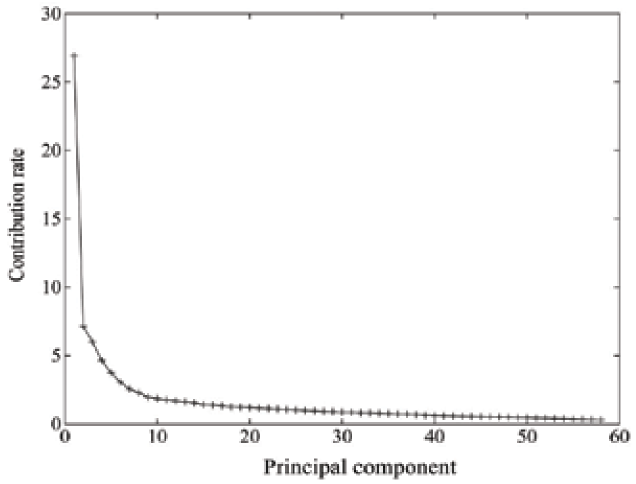


Fig. 12. Contribution rates for 58 variables.

The ranking seems to be quite similar to that obtained by the information loss. This means that the principal component seems to represent the main features by which different groups can be separated. On the other hand, connection weights by forced information represent the absolute magnitude of students' interest in the subjects. In forced-information maximization, we can see information loss as well as connection weights. The connection weights represent the absolute value of the importance. On the other hand, the information loss represents difference between several groups. This is a kind of relative importance of variables, because the importance of a variable in one group is measured in a relation to the other group.

No.	No.(Figure)	Strength	Item
1	20	0.1589	Multimedia
2	30	0.1539	Economics
3	15	0.1537	Business
4	29	0.1536	Business management
5	27	0.1514	Stock
6	18	0.1512	Marketing
7	34	0.1492	Publicity
8	47	0.1492	Politics
9	31	0.1482	Arts
10	33	0.1477	Language
49	52	0.1150	Physics
50	8	0.1115	Entertainment
51	49	0.1094	Illustrator
52	48	0.1085	Shipping
53	10	0.1070	Game
54	4	0.1049	Internet
55	42	0.1028	Craftwork
56	50	0.0986	Railroad
57	26	0.0981	Chemistry
58	2	0.0922	Animation

Table 12. The first principal component.

8. Conclusion

In this paper, we have proposed a new computational method to accelerate a process of information maximization. Information-theoretic competitive learning has been introduced to solve the fundamental problems of conventional competitive learning such as the dead neuron problem, dependency on initial conditions and so on. Though information theoretic competitive learning has demonstrated much better performance in solving these problems, we have observed that sometimes learning is very slow, especially when problems become very complex. To overcome this slow convergence, we have introduced forced information maximization. In this method, information is supposed to be maximized before learning. By using the WTA algorithm, we have introduced forced information in information-theoretic competitive learning. We have applied the method to several problems. In all problems, we have seen that learning is much accelerated, and for the student survey case, networks converge more than seventy times faster. Though we need to explore the exact mechanism of forced information maximization, the computational method proposed in this paper enables information theoretic learning to be applied to more large-scale problems.

9. Acknowledgment

The author is very grateful to Mitali Das for her valuable comments.

10. References

[1] R. Kamimura, T. Kamimura, and O. Uchida, "Flexible feature discovery and structural information," *Connection Science*, vol. 13, no. 4, pp. 323–347, 2001.

[2] R. Kamimura, T. Kamimura, and H. Takeuchi, “Greedy information acquisition algorithm: A new information theoretic approach to dynamic information acquisition in neural networks,” *Connection Science*, vol. 14, no. 2, pp. 137–162, 2002.

[3] R. Kamimura, “Information theoretic competitive learning in self-adaptive multi-layered networks,” *Connection Science*, vol. 13, no. 4, pp. 323–347, 2003.

[4] R. Kamimura, “Information-theoretic competitive learning with inverse euclidean distance,” *Neural Processing Letters*, vol. 18, pp. 163–184, 2003.

[5] R. Kamimura, “Unifying cost and information in information-theoretic competitive learning,” *Neural Networks*, vol. 18, pp. 711–718, 2006.

[6] D. E. Rumelhart and D. Zipser, “Feature discovery by competitive learning,” in *Parallel Distributed Processing* (D. E. Rumelhart and G. E. H. et al., eds.), vol. 1, pp. 151–193, Cambridge: MIT Press, 1986.

[7] S. Grossberg, “Competitive learning: from interactive activation to adaptive resonance,” *Cognitive Science*, vol. 11, pp. 23–63, 1987.

[8] D. DeSieno, “Adding a conscience to competitive learning,” in *Proceedings of IEEE International Conference on Neural Networks*, (San Diego), pp. 117–124, IEEE, 1988.

[9] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton, “Competitive learning algorithms for vector quantization,” *Neural Networks*, vol. 3, pp. 277–290, 1990.

[10] L. Xu, “Rival penalized competitive learning for clustering analysis, RBF net, and curve detection,” *IEEE Transaction on Neural Networks*, vol. 4, no. 4, pp. 636–649, 1993.

[11] A. Luk and S. Lien, “Properties of the generalized lotto-type competitive learning,” in *Proceedings of International conference on neural information processing*, (San Mateo: CA), pp. 1180–1185, Morgan Kaufmann Publishers, 2000.

[12] M. M. V. Hulle, “The formation of topographic maps that maximize the average mutual information of the output responses to noiseless input signals,” *Neural Computation*, vol. 9, no. 3, pp. 595–606, 1997.

[13] L. L. Gatlin, *Information Theory and Living Systems*. Columbia University Press, 1972.

7

Learning to Build a Semantic Thesaurus from Free Text Corpora without External Help

Katia Lida Kermanidis
Ionian University
Greece

1. Introduction

The automatic extraction and representation of domain knowledge has been attracting the interest of researchers significantly during the last years. The plethora of available information, the need for intelligent information retrieval, as well as the rise of the semantic web, have motivated information scientists to develop numerous approaches to building thesauri, like dictionaries and Ontologies that are specific to a given domain.
Ontologies are hierarchical structures of domain concepts that are enriched with semantic relations linking the concepts together, as well as concept properties. Domain terms populate the ontology, as they are assigned to belong to one or more concepts, and enable the communication and information exchange between domain experts. Furthermore, domain Ontologies enable information retrieval, data mining, intelligent search, automatic translation, question answering within the domain.
Building Ontologies automatically to the largest extent possible, i.e. keeping manual intervention to a minimum, has first the advantage of an easily updateable extracted ontology, and second of largely avoiding the subjective, i.e. biased, impact of domain experts, which is inevitable in manually-based approaches.
This chapter describes the knowledge-poor process of extracting ontological information in the economic domain mostly automatically from Modern Greek text using statistical filters and machine learning techniques. Fig. 1 shows the various stages of the process. In a first stage, the text corpora are being pre-processed. Pre-processing includes tokenization, basic morphological tagging and recognition of named and other semantic entities, that are

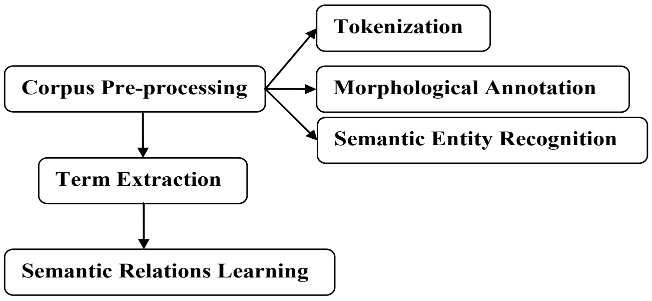


Fig. 1. System overview

related to the economic domain (e.g. values, amounts, percentages etc), and that would be useful in future data-mining applications. In a second stage, content-words in the text are categorized into domain terms and non-terms, i.e. words that are economic terms and words that aren't. Finally, domain terms are linked together with various types of semantic relations, such as hyponymy/hyperonymy (is-a), meronymy (part-of), and other relations of economic nature that don't fit the typical profile of *is-a* or *part-of* relations.

2. Comparison to related work

As mentioned earlier, significant research effort has been put into the automatic extraction of domain-specific knowledge. This section describes the most characteristic approaches for every stage in the process, and compares the proposed process to them.

Regarding named entity recognition, Hendrickx and Van den Bosch (2003) employ manually tagged and chunked English and German datasets, and use memory-based learning to learn new named entities that belong to four categories. They perform iterative deepening to optimize their algorithmic parameter and feature selection, and extend the learning strategy by adding seed list (gazetteer) information, by performing stacking and by making use of unannotated data. They report an average f-score on all four categories of 78.20% on the English test set. Another approach that makes use of external gazetteers is described in (Ciaramita & Altun, 2005), where a Hidden Markov Model and Semi-Markov Model is applied to the CoNLL 2003 dataset. The authors report a mean f-score of 90%. Multiple stacking is also employed in (Tsukamoto et al., 2002) on Spanish and Dutch data and the authors report 71.49% and 60.93% mean f-score respectively. The work in (Sporleder et al., 2006) focuses on the Natural History domain. They employ a Dutch zoological database to learn three different named-entity classes, and use the contents of specific fields of the database to bootstrap the named entity tagger. In order to learn new entities they, too, train a memory-based learner. Their reported average f-measure reaches 68.65% for all three entity classes. Other approaches (Radu et al., 2003; Wu et al., 2006) utilize combinations of classifiers in order to tag new named entities by ensemble learning.

For the automatic extraction of domain terms, various approaches have been proposed in the literature. Regarding the linguistic pre-processing of the text corpora, approaches vary from simple tokenization and part-of-speech tagging (Drouin, 2004; Frantzi et al., 2000), to the use of shallow parsers and higher-level linguistic processors (Hulth, 2003; Navigli & Velardi, 2004). The latter aim at identifying syntactic patterns, like noun phrases, and their structure (e.g. head-modifier), in order to rule out tokens that are grammatically impossible to constitute terms (e.g. adverbs, verbs, pronouns, articles, etc). The statistical filters, that have been employed in previous work to filter out non-terms, also vary. Using corpus comparison, the techniques try to identify words/phrases that present a different statistical behaviour in the corpus of the target domain, compared to their behaviour in the rest of the corpora. Such words/phrases are considered to be terms of the domain in question. In the simplest case, the observed frequencies of the candidate terms are compared (Drouin, 2004). Kilgarriff (2001) experiments with various other metrics, like the χ^2 score, the t-test, mutual information, the Mann-Whitney rank test, the Log Likelihood, Fisher's exact test and the TF.IDF (term frequency-inverse document frequency). Frantzi et al. (2000) present a metric that combines statistical (frequencies of compound terms and their nested sub-terms) and linguistic (context words are assigned a weight of importance) information.

In the field of taxonomy learning, previous approaches have varied from supervised to unsupervised clustering techniques, and from methodologies that make use of external taxonomic thesauri, to those that rely on no external resources. Regarding previous approaches that employ clustering techniques, Cimiano et al. (2004) describe a conceptual clustering method that is based on the Formal Concept Analysis for automatic taxonomy construction from text and compares it to similarity-based clustering (agglomerative and Bi-Section-KMeans clustering). The automatically generated ontology is compared against a hand-crafted gold standard ontology for the tourism domain and report a maximum lexical recall of 44.6%. Other clustering approaches are described in (Faure & Nedellec, 1998) and (Pereira et al., 1993). The former uses a syntactically parsed text (verb-subcategorization examples) and utilize iterative clustering to form new concept graphs. The latter also makes use of verb-object dependencies, and relative frequencies and relative entropy as similarity metrics for clustering. Pekar and Staab (2002) take advantage of a taxonomic thesaurus (a tourism-domain ontology) to improve the accuracy of classifying new words into its classes. Their classification algorithm is an extension of the *k-NN* method, which takes into account the taxonomic similarity between nearest neighbors. They report a maximum overall accuracy of 43.2%. Lendvai (2005) identifies taxonomic relations between two sections of a medical document using memory-based learning. Binary vectors represent overlap between the two sections, and the tests are run on parts of two Dutch medical encyclopedias. A best overall accuracy value of 88% is reported. Witschel (2005) proposes a methodology for extending lexical taxonomies by first identifying domain-specific concepts, then calculating semantic similarities between concepts, and finally using decision trees to insert new concepts to the right position in the taxonomy tree. The classifier is evaluated against two subtrees from GermaNet. Navigli and Velardi (2004) interpret semantically the set of complex terms that they extract, based on simple string inclusion. They make use of a variety of external resources in order to generate a semantic graph of senses. Another approach that makes use of external hierarchically structured textual resources is (Makagonov et al., 2005). The authors map an already existing hierarchical structure of technical documents to the structure of a domain-specific technical ontology. Words are clustered into concepts, and concepts into topics. They evaluate their ontology against the structure of existing textbooks in the given domain. Maedche and Volz (2001) make use of clustering, as well as pattern-based (regular expressions) approaches in order to extract taxonomies from domain-specific German texts. Degeratu and Hatzivassiloglou (2004) also make use of syntactic patterns to extract hierarchical relations, and measure the dissimilarity between the attributes of the terms using the Lance and Williams coefficient. They evaluate their methodology on a collection of forms provided by the state agencies and report a precision value of 73% and 85% for *is-a* and attributive relations respectively.

Compared to previous approaches, the work described in this chapter includes some interesting novel aspects. The whole process is based on the effort to utilize as limited external linguistic resources as possible, in order to render the methodology easily portable to other languages and other thematic domains. To this purpose no semantic networks like WordNet, grammars, hierarchically structured corpora, or pre-existing Ontologies are utilized, only two unstructured corpora of free Modern Greek text: one balanced in domain and genre, and one domain-specific.

Another interesting aspect of the present work is the language itself. Modern Greek is a relatively free-word-order language, i.e. the ordering of the constituents of a sentence is not strictly fixed, like it is in English. Therefore, it is primarily the rich morphology and not the position of a word in a sentence that determines its syntactic and semantic role. As a result, the extraction of compound terms, as well as the identification of nested terms, are not straightforward and cannot be treated as cases of simple string concatenation. The grammatical case of nouns and adjectives affects their semantic labelling. Still, the language-dependent features of the process are not so binding to not allow it to be applicable to other inflectional languages with relative easiness.

Looking at each stage of the process in more detail, there are further application-specific interesting features to be noted. As mentioned earlier in this section, classical approaches to named-entity recognition are limited to names of organizations, persons and locations. The semantic entities in the present work, however, also cover names of stocks and bonds, as well as names of newspapers (due to the newswire genre of the used corpus). Furthermore, there are other semantic types that are important for economic information retrieval, like quantitative units (e.g. denoting stock and fund quantities, monetary amounts, stock values), percentages etc. Temporal words and expressions are also identified due to their importance for data mining tasks.

Traditionally, approaches to terminology extraction make use of a domain-specific corpus that is to a large extent restricted in the vocabulary it contains and in the variety of syntactic structures it presents. The economic corpus in this work does not consist of syntactically standardized taglines of economic news. On the contrary, it presents a very rich variety in vocabulary, syntactic formulations, idiomatic expressions, sentence length, making the process of term extraction an interesting challenge.

Finally, regarding semantic relation learning, related work focuses mostly on hyperonymy/hyponymy and meronymy, in the process described here *attribute* relations are also detected, i.e. more 'abstract' relations that are specific to the economic domain. For example, *rise* and *drop* are two attributes of the concept *value*, a *stockholder* is an attribute of the concept *company*.

3. Advanced learning schemata

The lack of sophisticated resources leads unavoidably to the presence of noise in the data. Noise is examples of useless data that not only do not help the learning of useful, interesting linguistic information, but they also mislead the learning algorithm, harming its performance. In machine learning terms, noise appears in the form of class imbalance. Positive class instances (instances of the class of interest that needs to be learned) in the data are underrepresented compared to negative instances (null class instances). Class imbalance has been dealt with in previous work in various ways: oversampling of the minority class until it consists of as many examples as the majority class (Japkowicz, 2000), undersampling of the majority class (random or focused), the use of cost-sensitive classifiers (Domingos, 1999), the ROC convex hull method (Provost & Fawcett, 2001).

3.1 One-sided sampling

In the present methodology, One-sided sampling (Kubat & Matwin, 1997; Laurikkala, 2001) has been chosen to deal with the noise when learning taxonomy relations as it generally leads to better classification performance than oversampling, and it avoids the problem of arbitrarily assigning initial costs to instances that arises with cost-sensitive classifiers. One-sided sampling prunes out redundant and misleading negative examples while keeping all the positive examples. Instances of the majority class can be categorized into four groups:

Noisy are instances that appear within a cluster of examples of the opposite class; *borderline* are instances close to the boundary region between two classes; *redundant* are instances that can be already described by other examples of the same class; *safe* are instances crucial for determining the class. Instances belonging to one of the first three groups need to be eliminated as they do not contribute to class prediction. Noisy and borderline examples can be detected using *Tomek links*: two examples, x and y, of opposite classes have a distance of $\delta(x,y)$. This pair of instances constitutes a Tomek link if no other example z exists, such that $\delta(x,z) < \delta(x,y)$ or $\delta(y,z) < \delta(x,y)$. Redundant instances may be removed by creating a *consistent subset* of the initial training set. A subset C of training set T is consistent with T, if, when using the nearest neighbor (*1-NN*) algorithm, it correctly classifies all the instances in T. To this end we start with a subset C of the initial dataset T, consisting of all positive examples and a few (e.g. 20) negative examples. We train a learner with C and try to classify the rest of the instances of the initial training set. All misclassified instances are added to C, which is the final reduced dataset. The normalized Euclidean distance function is used to detect noisy and borderline examples. One-sided sampling has been used in the past in several domains such as image processing (Kubat & Matwin, 1997), medicine (Laurikkala, 2001), text categorization (Lewis & Gale, 1994).

3.2 Ensemble learning

Ensemble learning schemata have also been experimented with to deal with the noise and help the learner to disregard the useless foggy examples and focus on the useful content data. An ensemble of classifiers is a set of individual (base) classifiers whose output is combined in order to classify new instances. The construction of good ensembles of classifiers is one of the most active areas of research in supervised learning, aiming mainly at discovering ensembles that are more accurate than the individual classifiers that make them up (Dietterich, 2002). Various schemes have been proposed for combining the predictions of the base classifiers into a unique output. The most important are *bagging*, *boosting* and *stacking*. *Bagging* entails the random partitioning of the dataset in equally sized subsets (bags) using resampling (Breiman, 1996). Each subset trains the same base classifier and produces a classification model (hypothesis). The class of every new test instance is predicted by every model, and the class label with the majority vote is assigned to the test instance. Unlike bagging, where the models are created separately, *boosting* works iteratively, i.e. each new model is influenced by the performance of those built previously (Freund & Schapire, 1996; Schapire et al., 2002). In other words, new models are forced, by appropriate weighting, to focus on instances that have been handled incorrectly by older ones. Finally, *stacking* usually combines the models created by different base classifiers, unlike bagging and stacking where all base models are constructed by the same classifier (Dietterich, 2002). After constructing the different base models, a new instance is fed into them, and each model predicts a class label. These predictions form the input to another, higher-level classifier (the so-called *meta-learner*), that combines them into a final prediction.

4. The corpora

The corpora used in our experiments were:

1. The ILSP/ELEFTHEROTYPIA (Hatzigeorgiu et al., 2000) and ESPRIT 860 (Partners of ESPRIT-291/820, 1986) Corpora (a total of 300,000 words). Both these corpora are

balanced in genre and domain and manually annotated with complete morphological information. Further (phrase structure) information is obtained automatically.

2. The DELOS Corpus (Kermanidis et al., 2002) is a collection of economic domain texts of approximately five million words and of varying genre. It has been automatically annotated from the ground up. Morphological tagging on DELOS was performed by the analyzer of (Sgarbas et al., 2000). Accuracy in part-of-speech and case tagging reaches 98% and 94% accuracy respectively. Further (phrase structure) information is again obtained automatically.

All of the above corpora (including DELOS) are collections of newspaper and journal articles. More specifically, regarding DELOS, the collection consists of texts taken from the financial newspaper EXPRESS, reports from the Foundation for Economic and Industrial Research, research papers from the Athens University of Economics and several reports from the Bank of Greece. The documents are of varying genre like press reportage, news, articles, interviews and scientific studies and cover all the basic areas of the economic domain, i.e. microeconomics, macroeconomics, international economics, finance, business administration, economic history, economic law, public economics etc. Therefore, it presents richness in vocabulary, in linguistic structure, in the use of idiomatic expressions and colloquialisms, which is not encountered in the highly domain- and language-restricted texts used normally for term extraction (e.g. medical records, technical articles, tourist site descriptions). To indicate the linguistic complexity of the corpus, we mention that the length of noun phrases varies from 1 to 53 word tokens.

All the corpora have been phrase-analyzed by the chunker described in detail in (Stamatatos et al., 2000). Noun, verb, prepositional, adverbial phrases and conjunctions are detected via multi-pass parsing. From the above phrases, noun and prepositional phrases only are taken into account for the present task, as they are the only types of phrases that may include terms. Regarding the phrases of interest, precision and recall reach 85.6% and 94.5% for noun phrases, and 99.1% and 93.9% for prepositional phrases respectively. The robustness of the chunker and its independence on extravagant information makes it suitable to deal with a style-varying and complicated in linguistic structure corpus like DELOS.

It should be noted that phrases are non-overlapping. Embedded phrased are flatly split into distinct phrases. Nominal modifiers in the genitive case are included in the same phrase with the noun they modify; nouns joined by a coordinating conjunction are grouped into one phrase. The chunker identifies basic phrase constructions during the first passes (e.g. adjective-nouns, article nouns), and combines smaller phrases into lon ger ones in later passes (e.g. coordination, inclusion of genitive modifiers, compound phrases). As a result, named entities, proper nouns, compound nominal constructions are identified during chunking among the rest of the noun phrases.

5. Learning semantic entities

The tagging of semantic entities in written text is an important subtask for information retrieval and data mining and refers to the task of identifying the entities and assigning them to the appropriate semantic category. In the present work, each token in the economic corpus constitutes a candidate semantic entity. Each candidate entity is represented by a feature-value vector, suitable for learning. The features forming the vector are:

1. The token lemma. In the case where automatic lemmatization was not able to produce the token lemma, the token itself is the value of this feature.
2. The part-of-speech category of the token.

3. The morphological tag of the token. The morphological tag is a string of 3 characters encoding the case, number, and gender of the token, if it is nominal (noun, adjective or article).
4. The case tag of the token. The case tag is one of three characters denoting the token case.
5. Capitalization. A Boolean feature encodes whether the first letter of the token is capitalized or not.

For each candidate entity, context information was included in the feature-value vector, by taking into account the two tokens preceding and the two tokens following it. Each of these tokens was represented in the vector by the five features described above. As a result, a total of 25 (5x5) features are used to form the instance vectors.

The class label assigns a semantic tag to each candidate token. These tags represent the entity boundaries (whether the candidate token is the start, the end or inside an entity) as well as the semantic identity of the token. A total of 40,000 tokens were manually tagged with their class value. Table 1 shows the various values of the class feature, as well as their frequency among the total number of tokens.

Tag	Description	Percentage
AE	Start of company/organization/bank name	1.4%
ME	Middle of company/organization/bank name	0.74%
TE	End of company/organization/bank name	1.4%
E	Company/organization/bank 1-word name	1.1%
AP	Start of monetary amount/price/value	0.88%
MP	Middle of monetary amount/price/value	0.63%
TP	End of monetary amount/price/value	0.88%
AAM	Start of number of stocks/bonds	0.3%
MAM	Middle of number of stocks/bonds	0.42%
TAM	End of number of stocks/bonds	0.3%
AT	Start of percentage value	0.73%
MT	Middle of percentage value	0.08%
TT	End of percentage value	0.73%
AX	Start of temporal expression	1%
MX	Middle of temporal expression	0.75%
TX	End of temporal expression	1%
X	1-word temporal expression	0.55%
AO	Start of stock/bond name	0.16%
MO	Middle of stock/bond name	0.17%
TO	End of stock/bond name	0.16%
ON	1-word stock/bond name	0.05%
AL	Start of location name	0.21%
ML	Middle of location name	0.48%
TL	End of location name	0.21%
L	1-word location name	0.33%
F	1-word newspaper/journal name	0.14%
AN	Start of person name	0.18%
MN	Middle of person name	0.02%
TN	End of person name	0.18%
N	1-word person name	0.06%

Table 1. Values of the semantic entities class label

Unlike most previous approaches that focus on labelling three or four semantic categories of named entities, the present work deals with a total of 30 class values plus the non-entity (NULL) value, as can be seen in the previous table.
Another important piece of information provided disclosed by the previous table is the imbalance between the populations of the positive instances (entities) in the dataset, that form only 15% of the total number of instances, and the negative instances (non-entities). This imbalance leads to serious classification problems when trying to classify instances that belong to one of the minority classes (Kubat & Matwin, 1997). By removing negative examples, so that their number reaches that of the positive examples (Laurikkala, 2001), the imbalance is attacked and the results prove that classification accuracy of the positive instances improves considerably.

5.1 Experimental setup and results

Instance-based learning (*1-NN*) was the algorithm selected to classify the candidate semantic entities. *1-NN* was chosen because, due to storing all examples in memory, it is able to deal competently with exceptions and low-frequency events, which are important in language learning tasks (Daelemans et al., 1999), and are ignored by other learning algorithms.
Several experiments were conducted for determining the optimal context window size of the candidate entities. Sizes (-2, +2) - two tokens preceding and two following the candidate entity - and (-1, +1) - one token preceding and one following the candidate entity - were experimented with, and comparative performance results were obtained. When decreasing the size from (-2, +2) to (-1, +1), the number of features forming the instance vectors drops from 25 to 15. The results are shown in the second and third column of Table 2.
Another set of experiments focused on comparing classification in one stage and in two stages, i.e. stacking. In the first stage, the Instance-based learner predicts the class labels of the test instances. In the second stage, the predictions of the first phase are added to the set of features that are described in the previous section. The total number of features in the second stage, when experimenting with the (-2, +2) context window, is 30. The results of learning in two stages with window size (-1, +1) are shown in the fourth column of Table 2.
Comparative experiments were also performed with and without the removal of negative examples, in order to prove the increase in performance after applying random undersampling to the data. With random undersampling, random instances of the majority class are removed from the dataset in order for their number to reach that of the positive classes. The classification results, after applying the undersampling procedure and for context window size (-1, +1), are presented in the last column of Table 2. Testing of the algorithm was performed using 10-fold cross validation.
For a qualitative analysis of the results, a set of graphs follows that groups them together into clusters. Fig. 2 shows the impact of the selected context window size on the classification process to the various classes in the initial dataset. The bars represent the average f-score for every semantic entity type, e.g. *Stock/bond name* is the average value of the AO, MO, TO and ON classes. Certain types of entities require a larger window for their accurate detection, while larger context is misleading for other types. To the former category belong multi-word entities like stock names, person and location names. Entities that consist normally of two words at the most, or one word and a symbol (like amounts, prices, etc.) belong to the second category.
Fig. 3 shows the grouped results for the start, middle, end and 1–word labels in the initial dataset. For example, the Start bar is the average f-score over all the start labels. The Middle

class group presents the lowest results, especially when a small context window size is used. This may be attributed to the fact that tokens in the inside of an entity are normally neither preceded nor followed by characteristic keywords or symbols. Therefore, their detection is harder than that of the entity borders, as the environment surrounding the entity helps the classification decision for the borders.

Class	F-score (-1,+1)	F-score (-2,+2)	F-score Stacking	F-score Undersampling
NULL	0.969	0.96	0.981	0.939
AE	0.728	0.683	0.882	0.899
ME	0.557	0.64	0.831	0.808
TE	0.768	0.74	0.871	0.903
AP	0.851	0.767	0.96	0.96
MP	0.865	0.852	0.957	0.963
TP	0.84	0.774	0.932	0.932
E	0.667	0.621	0.721	0.803
AAM	0.754	0.675	0.895	0.895
MAM	0.769	0.708	0.944	0.911
TAM	0.611	0.643	0.865	0.838
AO	0.353	0.465	0.81	0.85
MO	0.194	0.293	0.55	0.5
TO	0.143	0.35	0.629	0.611
AT	0.911	0.802	0.985	0.98
MT	0.588	0.857	0.952	0.952
TT	0.939	0.818	0.954	0.96
AX	0.585	0.558	0.755	0.806
TX	0.588	0.492	0.736	0.774
AL	0.421	0.449	0.651	0.571
ML	0.059	0.17	0.562	0.632
TL	0.278	0.293	0.524	0.465
X	0.452	0.457	0.567	0.694
F	0.889	0.947	0.944	1
AN	0.286	0.364	0.65	0.756
TN	0.378	0.632	0.65	0.579
MX	0.524	0.561	0.802	0.8
MN	0	0	0	0
ON	0	0	0	0
N	0.667	0.571	0.533	0.571
L	0.519	0.506	0.55	0.565

Table 2. Detailed experimental results

As can be seen in Table 2, classification for certain types reaches a poor score. Looking more closely at Table 1, this can be attributed without a doubt to the sparseness that characterizes

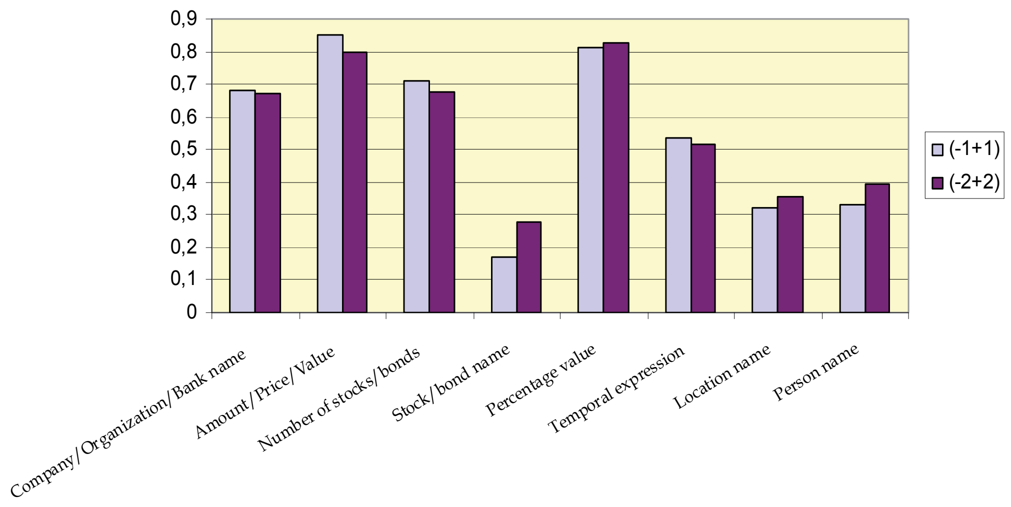


Fig. 2. The impact of the context window size

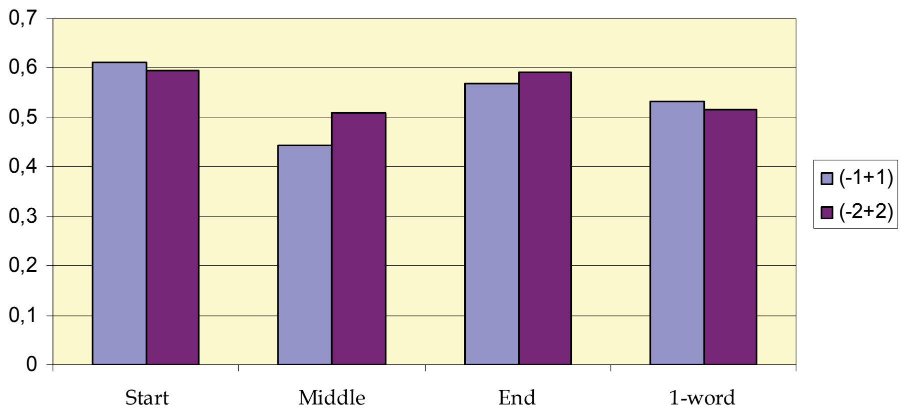


Fig. 3. The average results for the Start, Middle, End and 1-word class groups

these types (multi-word person names, multi-word stock/bond names, multi-word locations). An interesting exception to this rule is newspaper/journal names, that reach very high scores, despite their low frequency, because they are normally introduced by specific words like *'εφημερίδα'* (newspaper) or *'περιοδικό'* (journal).
Table 2 also shows the high f-score achieved for the negative (NULL) class compared to that of the positive classes, due to its high over-representation in the dataset.
The fourth column of Table 2 shows the positive effects of stacking on the task at hand. The f-score increases up to more than 50% after applying two-phase learning. This improvement is due to two reasons: first, the sequential nature of the class label tags (start, middle, end). The class of one entity depends largely on the class of the preceding and the following entities. Second, the inclusion of the predicted class of the candidate entity (from the

previous learning stage) in the feature vector of the second stage forces the classifier to focus on the mistakes it made, and try to correct them. Difficult cases like multi-word locations and multi-word names are now dealt with satisfactorily.
Random undersampling also proved highly beneficial for the majority of the entity categories. It forces the learner to pay more attention to the minority classes. The random nature of the undersampling process is the reason that the results for certain entity types were not improved, as certain useful negative examples may have been removed.
The positive effects of stacking and undersampling are shown clearly in Figure 4.

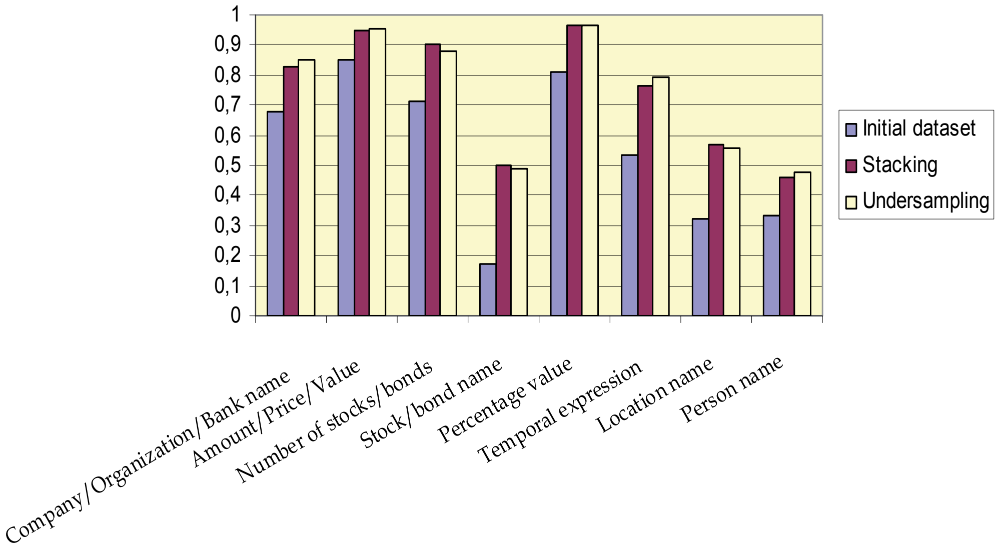


Fig. 4. The average results for all semantic entity types using Stacking and Undersampling.

One-word stock/bond names (ON) occur extremely seldom in the corpus. Person names consisting of more than two words (MN), are even more rare. The learner has not been able to detect these classes due to the sparseness.
Given, however, the nature and complexity of the corpus, the low level of pre-processing (compared to previous approaches that use phrase-chunked input), and the large number of class labels, the results of Table 2 are very impressive when compared to the ones reported in the literature.

6. Extracting economic terms

The next step of the procedure is the automatic extraction of economic terms, following the methodology described in (Thanopoulos et al., 2006). Corpora comparison was employed for the extraction of economic terms. Corpora comparison detects the difference in statistical behavior that a term presents in a balanced and in a domain-specific corpus.
Noun and prepositional phrases of the two corpora are selected to constitute candidate terms, as only these phrase types are likely to contain terms. The occurrences of words and multi-word units (n-grams), pure as well as nested, are counted. Longer candidate terms are split into smaller units (tri-grams into bi-grams and uni-grams, bi-grams into uni-grams).

Due to the relative freedom in the word ordering in Modern Greek sentences, bi-gram A B (A and B being the two lemmata forming the bi-gram) is considered to be identical to bi-gram B A, if the bi-gram is not a semantic entity. Their joint count in the corpora is calculated and taken into account. The resulting uni-grams and bi-grams are the candidate terms. The candidate term counts in the corpora are then used in statistical filters.
Statistical filtering is performed in two stages: First the relative frequencies are calculated for each candidate term. Then, for those candidate terms that present a relative frequency value greater than 1, the Log Likelihood ratio (LLR) is calculated. The LLR metric detects how surprising (or not) it is for a candidate term to appear in the domain-specific or in the balanced corpus (compared to its expected appearance count), and therefore constitute an economic domain term (or not).

Rank	Word	Translation	Count 1	Count 2	RF	LLR
1	εταιρία	company	5396	0	1845.9	852.0
2	δρχ	drachmas	3003	1	342.5	465.5
3	μετοχή	stock	2827	6	74.4	414.0
4	αγορά	buy	2330	33	11.9	257.2
5	αύξηση	growth, rise	2746	66	7.1	247.6
6	κέρδος	profit	1820	15	20.1	228.2
7	τράπεζα	bank	1367	11	20.3	171.8
8	επιχείρηση	enterprise	1969	56	6.0	162.1
9	κεφάλαιο	capital	1325	14	15.6	157.3
10	σημαντικός	important	1872	56	5.7	149.3
11	πώληση	sale	1203	11	17.9	147.3
12	προϊόν	product	1282	16	13.3	146.0
13	όμιλος	company, group	1036	5	32.2	140.0
14	A.E.	INC	820	0	280.7	126. 4
15	μετοχικός	stocking	790	2	54.1	112.8
16	τιμή	price	1722	70	4.2	110.9
17	επιτόκιο	interest	821	4	31.2	110.0
18	υψηλός	high	711	0	243.4	109.2
19	κόστος	cost	1031	19	9.0	103.4
20	κλάδος	branch	833	7	19.0	103.2

Table 3. The 20 most highly ranked terms

Table 3 shows the relative frequency (*RF*) and *LLR* scores of the 20 most highly ranked economic terms, ordered by their *LLR* value. *Count 1* and *Count 2* are the term counts in the domain-specific and the balanced corpus respectively. An interesting term is '*υψηλός*', the ancient Greek form for '*high*', used today almost exclusively in the context of the degree of performance, growth, rise, profit, cost, drop (i.e. the appropriate form in economic context), as opposed to its modern form '*ψηλός*', which is used in the concept of the degree of actual height.
A particularity of the present work is that, unlike in most previous approaches to term extraction, the domain-specific corpus available to us is quite large compared to the

balanced corpus. As a result, several terms that appear in DELOS do not appear in the balanced corpus, making it impossible for the LLR statistic to detect them. In other words, these terms cannot be identified by traditional corpora comparison. Lidstone's law (Manning & Schuetze, 1999) was applied to the candidate terms, i.e. each candidate term count was augmented by a value of λ=0.5 in both corpora. Thereby, terms that actually do not appear in the balanced corpus at all, end up having a *Count 2* = 0.5. This value was chosen for λ because, due to the small size of the balanced corpus, the probability of coming across a previously unseen word is significant.

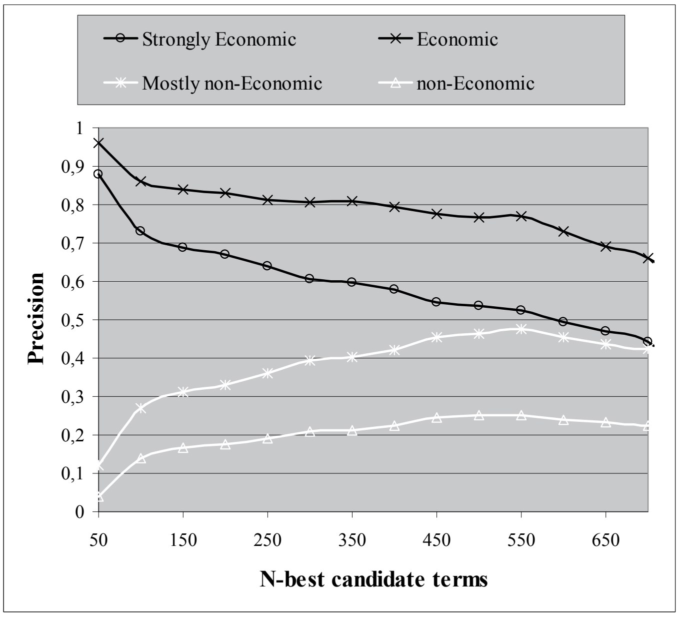


Fig. 5. Precision (y-axis) for the N-best candidate terms (x-axis) that appear in both corpora

As can be seen in Fig. 5, the term extraction methodology reaches a precision of 82% for the 200 N-best candidate terms. In this figure, *strongly economic* are terms that are characteristic of the domain and necessary for understanding domain texts. *Economic* are terms that function as economic within a context of this domain, but may also have a different meaning outside this domain. *Mostly non-economic* are words that are connected to the specific domain only indirectly, or more general terms that normally appear outside the economic domain, but may carry an economic sense in certain limited cases. *Non-economic* are terms that never appear in an economic sense or can be related to the domain in any way.

7. Learning semantic relations

The final step of the proposed methodology focuses on the identification of the taxonomic relations between the terms that were extracted in the previous phase. From the previous phase, the 250 most highly ranked terms (according to the LLR metric) were selected, and each one was paired with the rest. Syntactic and semantic information regarding the term pair has been encoded in a set of attributes that form a feature-value vector for each pair of terms. The proposed syntactic/semantic attributes are empirical and are described in the next sections. The term lemmata, their frequencies, and their part-of-speech tags were also included in the feature set. The semantic relations of a total of 6000 term pairs were manually annotated by economy and finance experts with one of the four class label values: *is-a*, *part-of*, *attribute relation* and no relation (*null*).

7.1 Semantic context vectors

The sense of a term is strongly linked to the context the term appears in. To this end, for each extracted term semantic context vectors have been constructed, that are comprised by the ten most frequent words the term co occurs with in the domain-specific corpus. A context window of two words preceding and two words following the term for every occurrence of the term in the corpus is formed. All non-content words (prepositions, articles, pronouns, particles, conjunctions) are disregarded, while acronyms, abbreviations, and certain symbols (e.g. %, €) are taken into account because of their importance for determining the semantic profile of the term in the given domain. Bi-grams (pairs of the term with each word within the con-text window) are generated and their frequency is recorded. The ten words that present the highest bi-gram frequency scores are chosen to form the context vector of the term.

7.2 Semantic similarity

For each pair of terms, their semantic similarity is calculated, based on their semantic context vectors. The smaller the distance between the context vectors, the more similar the terms' semantics. The value of semantic similarity is an integer with a value ranging from 0 to 10, which denotes the number of common words two context vectors share.

7.3 Semantic diversity

Another important semantic feature that is taken into account is how 'diverse' the semantic properties of a term are, i.e. the number of other terms that a term shares semantic properties with. This property is important when creating taxonomic hierarchies, because, the more 'shared' the semantic behaviour of a term is, the more likely it is for the term to have a higher place in the hierarchy. The notion of 'semantic diversity' is included in the feature set by calculating the percentage of the total number of terms whose semantic similarity with the focus term (one of the two terms whose taxonomic relation is to be determined) is at least 1.

7.4 Syntactic patterns

Syntactic information, regarding the linguistic patterns that govern the co occurrence of two terms, is significant for extracting taxonomic information. For languages with a relatively strict sentence structure, like English, such patterns are easier to detect (Hearst, 1992), and their impact on taxonomy learning more straightforward.

As mentioned earlier, Modern Greek presents a larger degree of freedom in the ordering of the constituents of a sentence, due to its rich morphology and its complex declination system. This freedom makes it difficult to detect syntactic patterns, and, even if they are detected, their contribution to the present task is not that easily observable.

However, two Modern Greek syntactic schemata prove very useful for learning taxonomies. They are the attributive modification schema and the genitive modification schema. The first, known in many languages, is the pattern where (usually) an adjective modifies the following noun. The second is typical for Modern Greek, and it is formed by two nominal expressions, one of which (usually following the other) appears in the genitive case and modifies the preceding nominal, denoting possession, property, origin, quantity, quality. The following phrases show examples of the first (example 1) and the second (examples 2, 3 and 4) schemata respectively.

(1)	το	μετοχικό[ADJ]	κεφάλαιο[NOUN]
	the	stock	capital

(2)	η	κατάθεση[NOUN]	επιταγής[NOUN-GEN]
	the	deposit	check
	(the deposit of the check)		

(3)	πρόεδρος[NOUN]	του	συμβουλίου[NOUN-GEN]
	head	the	council
	(head of the council)		

(4)	αύξηση[NOUN]	του	κεφαλαίου[NOUN-GEN]
	increase	the	capital
	(capital increase)		

Both these schemata enclose the notion of taxonomic relations: hyponymy relations (a *check deposit* is a type of deposit, a *stock capital* is a type of capital), as well as meronymy relations (the head is part of a council). The fourth example incorporates an attribute relation. The distinction among the types of relations is not always clear. In the check deposit example, the deposit may also be considered an attribute of check, constituting thereby an attribute relation. For each pair of terms, the number of times they occur in one of the two schemata in the domain-specific corpus is calculated. This information is basically the only language-dependent feature that is included in the methodology.

7.5 Experimental setup and results

9% of the term pairs belong to the *is-a* class, 17% belong to the *attribute* class and only 0.5% belong to the *part-of* class. The instances that belong to one of the first three classes are called positive, while those that belong to the null class are called negative.

Different classifiers lead to different results. Preliminary experiments have been run using various classification algorithms. *C4.5* is Quinlan's decision tree induction algorithm without pruning (Quinlan, 1993). Decision trees were chosen because of their high representational power, which is very significant for understanding the impact of each feature on the classification accuracy, and because of the knowledge that can be extracted from the resulting tree itself. The *1-NN* instanced-based learning algorithm is chosen to constitute a reference to a baseline classification performance. *SVM* is the Support Vector Machines classifier with a linear kernel. SVM cope well with the sparse data problem, and also with noise in the data (an inevitable phenomenon due to the automatic nature of the procedure described so far). A first degree polynomial kernel function was selected and the

Sequential Minimal Optimization algorithm was chosen to train the Support Vector classifier (Platt, 1998). *BN* is a Bayesian Network classifier, using a hill climbing search algorithm, and the conditional probability tables are estimated directly from the data.

	C4.5	1-NN	Naïve Bayes	SVM	BN
Is-a	0.808	0.694	0.419	0.728	0.762
Part-of	0.4	0	0	0	0
Attribute	0.769	0.765	0.77	0.788	0.775
Null	0.938	0.904	0.892	0.907	0.917

Table 4. Class f-score for various classifiers

Table 4 shows the f-score for each class achieved when trying to classify new term pairs using 10-fold cross validation. The poor results for the part-of relation are attributed mainly to its extremely rare occurrence in the data. The economic domain is more 'abstract' and is governed to a large extent by other relation types.
To overcome this problem of performance instability among the various classifiers, the application of ensemble learning is proposed. The combination of various disagreeing classifiers leads to a resulting classifier with better overall predictions (Dietterich, 2002). Experiments have been conducted using the aforementioned classifiers in various combination schemes using bagging, boosting and stacking.
Table 5 shows the results using bagging. Experiments were run using several base classifiers and several bag sizes as a percentage of the dataset size. A 50% bag size leads to the best classification results. 50% bag size means that half of the dataset instances were randomly chosen to form the first training set, another random half is used to form the second training set etc. After repeating the process ten times (10 iterations), the datasets are used to train the same base learner. Majority voting determines the class label for the test instances. The best results are achieved with a decision tree base classifier.

	C4.5	1-NN	SVM	BN
Is-a	0.856	0.736	0.728	0.766
Part-of	0	0	0	0
Attribute	0.809	0.765	0.786	0.783
Null	0.962	0.912	0.908	0.909

Table 5. Results with bagging

Table 6 shows the results using boosting. Again, various experiments were conducted with different base learners. The best results are again obtained with a decision tree base learner. It is interesting to observe the detection of some *part-of* relations using boosting.
Table 7 shows the results with stacking. Different base classifiers were combined, and their predictions were given as input to the higher level meta-learner. The combined classifiers are the 1-NN instance based-learner, the C4.5 decision tree learner, the Naïve Bayes learner, the Bayes Network classifier and the Support Vector Machine classifier. After running experiments with several combinations, it became obvious, that the greater the number and the diversity of the base classifiers, the better the achieved results. Using the same base

learner combination, numerous experiments were run to compare meta-learners (shown in Table 7). The best results are achieved using SVM as a meta-learner, but the results are very satisfactory with the other meta-learners as well. It is interesting to observe that even the simple lazy meta-learner, IB1, reaches an f-score higher than 81% for all three classes. This is attributed to the predictive power of the combination of base learners. In other words, the sophisticated base learners do all the hard work, deal with the difficult cases, and the remaining work for the meta-learner is simple.

	C4.5	1-NN	SVM	BN
Is-a	0.772	0.719	0.611	0.826
Part-of	0.286	0	0	0
Attribute	0.762	0.744	0.732	0.798
Null	0.922	0.903	0.92	0.944

Table 6. Results with boosting

Meta-learner	C4.5	1-NN	Naïve Bayes	SVM
Is-a	0.761	0.848	0.827	0.853
Part-of	0	0	0	0
Attribute	0.756	0.818	0.793	0.835
Null	0.94	0.952	0.947	0.957

Table 7. Results with stacking

A further set of experiments was performed, after applying One-sided sampling to the dataset. Approximately 9% of the negative examples were removed (37.5% of which were noisy or borderline, and the remaining 62.5% were redundant). The positive effect of balancing the dataset is clearer especially when experimenting with the 'simpler' classification algorithms (IB1or C4.5), as they are more sensitive to class distribution imbalances, compared to the more 'sophisticated' classification schemata (SVM, boosting). After balancing, both sophisticated learners are able to detect part-of relations. Table 8 shows the classification results for every class.

Meta-learner	C4.5	1-NN	Naïve Bayes	SVM
Is-a	0.805	0.776	0.781	0.789
Part-of	0	0	0.25	0.33
Attribute	0.805	0.71	0.811	0.794
Null	0.931	0.913	0.915	0.927

Table 8. Results with One-sided sampling

Comparing the results with ensemble learning (Tables 5, 6 and 7) and simple learning (Table 4), the positive impact of combining multiple classifiers into a single prediction scheme becomes apparent. Mistakes made by one single classifier are amended through the iterative process and the majority voting in bagging, through instance weighting, according to how difficult an instance is to predict, in boosting, and through combining the strengths of several distinct classifiers in stacking.

Among the several ensemble schemes, stacking achieves the highest results. As mentioned earlier, class prediction performance benefits significantly from combining different base learners, because, roughly speaking, the weaknesses of one classifier are 'overshadowed' by the strengths of another, leading to a significant improvement in overall prediction.

The *part-of* relation proves to be very problematic, even with meta-learning. This is not surprising, however, taking into account that only 0.5% of the data instances were labeled as *part-of* relations. This rare occurrence leads all learning algorithms to disregard these instances, except for the unpruned decision tree learner, either as a stand-alone classifier or as base classifier in a boosting scheme. When no pruning on the decision tree is performed, overlooking tree paths that might be important for classification is avoided, and, thereby, even very low frequency events may be taken into account.

8. Discussion and future research

This chapter described the process of extracting economic knowledge automatically from Modern Greek corpora, using statistical and supervised learning techniques. The knowledge includes semantic entities, economic terminology, and semantic taxonomic relations between the extracted terms. The presented methodology makes use of no external resources in order for it to be easily portable to other domains. The language-dependent features of the described approach are kept to a minimum, so that it can be easily adapted to other languages. The lack of sophisticated resources allows for 'noise' to penetrate the dataset, leading to an imbalance between the distribution of the positive (useful for learning) and the negative (useless and misleading) class instances. Advanced sampling and ensemble learning techniques were applied, in order to remove noisy and redundant examples of the majority class, or focus on the interesting, rare instances. Despite the use of minimal resources and the highly automated nature of the process, classification performance is very promising, compared to results reported in previous work.

The extracted relations are useful in many ways. They form a generic semantic thesaurus that can be further used in several applications. First, the knowledge is important for economy/finance experts for a better understanding and usage of domain concepts. Moreover, the thesaurus facilitates intelligent search. Looking for semantically related terms improves the quality of the search results. The same holds for information retrieval and data mining applications. Intelligent question/answering systems that take into account terms that are semantically related to the terms appearing in queries return information that is more relevant, more accurate and more complete.

The economic domain is governed by semantic relations that are characteristic of the domain (buy/sell, monetary/percentage, rise/drop relations etc.), and that have been included under the attribute relation label in this work. A more fine-grained distinction between these types of attribute relations is a challenging future research direction, providing information that is very useful for data mining applications in the particular domain.

Employing other learning algorithms, that are also able to deal with the class imbalance barrier, such as neural networks, and discovering the differences in their performance compared to the algorithms presented in this chapter, promises to be another future research challenge.

Finally, another future research perspective is building an integrated ontological thesaurus from the learned taxonomic relations. Organizing the extracted terms into a hierarchical structure, e.g. a semantic network will render the extracted knowledge even more useful.

9. References

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, Vol. 24, pp. 123-140

Ciaramita, M. & Altun, Y. (2005). Named Entity Recognition in Novel Domains with External Lexical Knowledge. *Proceedings of the Workshop on Advances in Structured Learning for Text and Speech Processing* (NIPS)

Cimiano, P.; Hotho, A. & Staab., S. (2004). Comparing Conceptual, Divisive and Agglomerative Clustering for Learning Taxonomies from Text. *Proceedings of the European Conference on Artificial Intelligence* (ECAI), Valencia, Spain

Daelemans, W.; van den Bosch, A. & Zavrel, J. (1999). Forgetting Exceptions is Harmful in Language Learning. *Machine Learning*, Vol. 34, pp. 11-41

Degeratu, M. & Hatzivassiloglou, V. (2004). An Automatic Model for Constructing Domain-Specific Ontology Resources. *Proceedings of the International Conference on Language Resources and Evaluation* (LREC), pp. 2001-2004, Lisbon, Portugal

Dietterich, T. (2002). *Ensemble Learning. The Handbook of Brain Theory and Neural Networks*. Second Edition. The MIT Press, Cambridge, Massachusetts, USA

Domingos, P. (1999). Metacost: A General Method for Making Classifiers Cost-sensitive. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 155-164, San Diego, California, USA

Drouin, P. (2004). Detection of Domain Specific Terminology Using Corpora Comparison. *Proceedings of the 4th International Conference on Language Resources and Evaluation* (LREC), pp. 79−82, Lisbon, Portugal

Faure, D. & Nedellec., C. (1998). A Corpus-based Conceptual Clustering Method for Verb Frames and Ontology. *Proceedings of the LREC Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications,* Granada, Spain

Frantzi, K.; Ananiadou, S. & Mima, H. (2000). Automatic Recognition of Multi-word Terms: the C-value/NC-value Method. *International Journal on Digital Libraries*, Vol. 3, No. 2, pp. 117−132

Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the International Conference on Machine Learning*, pp. 148-156, San Francisco, USA

Hatzigeorgiu, N.; Gavrilidou, M.; Piperidis, S.; Carayannis, G.; Papakostopoulou, A.; Spiliotopoulou, A.; Vacalopoulou, A.; Labropoulou, P.; Mantzari, E.; Papageorgiou, H.; & Demiros, I. (2000). Design and Implementation of the online ILSP Greek Corpus. *Proceedings of the 2nd International Conference on Language Resources and Evaluation* (LREC), pp. 1737-1742, Athens, Greece

Hearst, M. A. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. *Proceedings of the International Conference on Computational Linguistics*, pp. 539-545, Nantes, France

Hendrickx, I. & van den Bosch, A. (2003). Memory-based One-step Named-entity Recognition: Effects of Seed List Features, Classifier Stacking and Unannotated Data. *Proceedings of the 7th Conference on Computational Natural Language Learning* (CoNLL), Edmonton, Canada

Hulth, A. (2003). Improved Automatic Keyword Extraction Given More Linguistic Knowledge. *Proceedings of the International Conference on Empirical Methods in Natural Language Processing* (EMNLP), pp. 216-223, Sapporo

Japkowicz, N. (2000). The Class Imbalance Problem: Significance and Strategies. *Proceedings of the International Conference on Artificial Intelligence*, Las Vegas, USA

Kermanidis, K.; Fakotakis, N. & Kokkinakis, G. (2002). DELOS: An Automatically Tagged Economic Corpus for Modern Greek. *Proceedings of the 3rd International Conference on Language Resources and Evaluation* (LREC), pp. 93-100, Las Palmas de Gran Canaria, Spain

Kilgarriff, A. (2001). Comparing Corpora. *International Journal of Corpus Linguistics*, Vol. 6, No. 1, pp. 1-37

Kubat, M. & Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets. *Proceedings of the International Conference on Machine Learning*, pp. 179- 186, Nashville, Tennessee, USA

Laurikkala, J. (2001). Improving Identification of Difficult Small Classes by Balancing Class Distribution. *Proceedings of the 8th Conference on Artificial Intelligence in Medicine in Europe*, pp. 63-66, Cascais, Portugal

Lendvai, P. (2005). Conceptual Taxonomy Identification in Medical Documents. *Proceedings of the Second International Workshop on Knowledge Discovery and Ontologies,* pp. 31-38. Porto, Portugal

Lewis, D. & Gale, W. (1994). Training Text Classifiers by Uncertainty Sampling. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3-12, Dublin, Ireland

Maedche, A. & Volz, R. (2001). The Ontology Extraction and Maintenance Framework Text-To-Onto. *Proceedings of the Workshop on Integrating Data Mining and Knowledge Mining,* San Jose, California, USA

Makagonov, P.; Figueroa, A. R.; Sboychakov, K. & Gelbukh, A. (2005). Learning a Domain Ontology from Hierarchically Structured Texts. *Proceedings of the 22nd International Conference on Machine Learning* (ICML), Bonn, Germany

Manning, C. & Schuetze, H. (1999). *Foundations of Statistical Natural Language Processing*, MIT Press

Navigli, R. & Velardi, P. (2004). Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. *Computational Linguistics*, Vol. 30, No. 2, pp. 151−179, MIT Press, ISSN: 0891-2017

Partners of ESPRIT-291/860. (1986). Unification of the Word Classes of the ESPRIT Project 860. Internal Report BU-WKL-0376.

Pekar, V. & Staab, S. (2002). Taxonomy Learning –Factoring the Structure of a Taxonomy into a Semantic Classification Decision. *Proceedings of the International Conference on Computational Linguistics*, Taipei, Taiwan

Pereira, F.; Tishby, N. & Lee, L. (1993). Distributional Clustering of English Words. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*

Platt, J. (1998). Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, & A. Smola, Eds. MIT Press.

Provost, F. & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, Vol. 42, No. 3, pp. 203-231

Quinlan, R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA

Radu, F.; Ittycheriah A.; Jing H. & Zhang T. (2003). Named Entity Recognition through Classifier Combination. *Proceedings of the 7th Conference on Computational Natural Language Learning* (CoNLL), pp. 168-171, Edmonton, Canada

Schapire, R. E.; Rochery, M.; Rahim, M. & Gupta, N. (2002). Incorporating Prior Knowledge into Boosting. *Proceedings of the Nineteenth International Conference on Machine Learning*

Sgarbas, K.; Fakotakis, N. & Kokkinakis, G. (2000). A Straightforward Approach to Morphological Analysis and Synthesis. *Proceedings of the Workshop on Computational Lexicography and Multimedia Dictionaries* (COMLEX), pp. 31-34, Kato Achaia, Greece

Sporleder, C.; van Erp, M.; Porcelijn, T.; van den Bosch, A. & Arntzen, P. (2006). Identifying Named Entities in Text Databases from the Natural History Domain. *Proceedings of the 5th International Conference on Language Resources and Evaluation* (LREC)

Stamatatos, E.; Fakotakis, N. & Kokkinakis, G. (2000). A Practical Chunker for Unrestricted Text. *Proceedings of the Conference on Natural Language Processing* (NLP), pp. 139-150, Patras, Greece

Thanopoulos, A.; Kermanidis, K. & Fakotakis, N. (2006). Challenges in Extracting Terminology from Modern Greek Texts. *Proceedings of the Workshop on Text-based Information Retrieval*, (TIR), Riva del Garda, Italy

Tsukamoto, K.; Mitsuishi, Y. & Sassano, M. (2002). Learning with Multiple Stacking for Named Entity Recognition. *Proceedings of the 6th Conference on Natural Language Learning* (CoNLL), pp. 1-4,, Taipei, Taiwan

Witschel, H. F. (2005). Using Decision Trees and Text Mining Techniques for Extending Taxonomies. *Proceedings of the Workshop on Learning and Extending Lexical Ontologies by Using Machine Learning Methods*

Wu. C.; Jan, S.; Tsai, T. & Hsu, W. (2006). On Using Ensemble Methods for Chinese Named Entity Recognition. *Proceedings of the 5th SIGHAN Workshop on Chinese Language Processing*, pp. 142-145, Sydney, Australia

Machine Learning Methods for Spoken Dialogue Simulation and Optimization

Olivier Pietquin
Ecole Supérieure d'Electricité (Supélec)
France

1. Introduction

Computers and electronic devices are becoming more and more present in our day-to-day life. This can of course be partly explained by their ability to ease the achievement of complex and boring tasks, the important decrease of prices or the new entertainment styles they offer. Yet, this real incursion in everybody's life would not have been possible without an important improvement of Human-Computer Interfaces (HCI). This is why HCI are now widely studied and become a major trend of research among the scientific community. Designing "user-friendly" interfaces usually requires multidisciplinary skills in fields such as computer science, ergonomics, psychology, signal processing etc. In this chapter, we argue that machine learning methods can help in designing efficient speech-based human-computer interfaces.

Speech is often considered as the most convenient and natural way for humans to communicate and interact with each other. For this reason, speech and natural language processing have been intensively studied for more than 60 years. It has now reached a maturity level that should enable the design of efficient voice-based interfaces such as Spoken Dialogue Systems (SDS). Still, designing and optimizing a SDS is not only a matter of putting together speech and language processing systems such as Automatic Speech Recognition (ASR) (Rabiner & Juang 1993), Spoken Language Understanding (SLU) (Allen 1998), Natural Language Generation (NLG) (Reiter & Dale 2000), and Text-to-Speech (TTS) synthesis (Dutoit 1997) systems. It also requires the development of dialogue strategies taking at least into account the performances of these subsystems (and others), the nature of the task (e.g. form filling (Pietquin & Dutoit 2006a), tutoring (Graesser *et al* 2001), robot control, or database querying (Pietquin 2006b)), and the user's behaviour (e.g. cooperativeness, expertise (Pietquin 2004)). The great variability of these factors makes rapid design of dialogue strategies and reusability across tasks of previous work very complex. For these reasons, human experts are generally in charge of tailoring dialogue strategies which is costly and time-consuming. In addition, there is also no objective way to compare strategies designed by different experts or to objectively qualify their performance. Like for most software engineering tasks, such a design is a cyclic process. Strategy hand-coding, prototype releases and user tests are required making this process expensive and time-consuming.

In the purpose of obtaining automatic data-driven methods and objective performances measures for SDS strategy optimisation, statistical learning of optimal dialogue strategies became a leading domain of research (Lemon & Pietquin, 2007). The goal of such approaches is to reduce the number of design cycles (Fig.1).

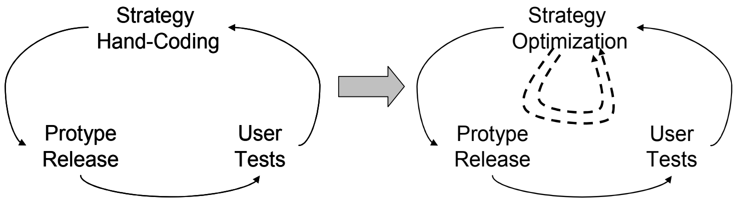


Fig. 1.Optimization for minimizing the number of design cycles

Supervised learning for such an optimization problem would require examples of ideal (sub)strategies which are typically unknown. Indeed, no one can actually provide an example of what would have objectively been the perfect sequencing of exchanges after having participated to a dialogue. Humans have a greater propensity to criticize what is wrong than to provide positive proposals. In this context, reinforcement learning using Markov Decision Processes (MDPs) (Levin *et al* 1998, Singh *et al* 1999, Scheffler & Young 2001, Pietquin & Dutoit 2006a, Frampton & Lemon 2006) and Partially Observable MDP (POMDPs) (Poupart *et al* 2005, Young 2006) has become a particular focus.
Such machine learning methods are very data demanding and sufficient amounts of annotated dialogue data are often not available for training. Different standard methods have therefore been investigated to deal with the data sparsity that can be split into two classes: statistical generation of new data by means of simulation (Schatzmann *et al*, 2007a) or generalization to unseen situations (Henderson *et al*, 2005).
In this chapter, we propose to provide an overview of the state of the art in machine learning for spoken dialogue systems optimization. This will be illustrated on a simple train ticket booking application.

2. Definitions and formalisms

2.1 Definitions

In this text, a *dialogue* will be describing an interaction between two *agents* based on sequential turn taking. We will only treat the special case of *goal-directed* dialogs where both agents cooperate in order to achieve an aim (or accomplish a task), like obtaining a train ticket for example. *Social* dialogues are out of the scope of this chapter. We will consider *man-machine* dialogs where one of the agents is a human *user* while the other is a computer (or *system*). In the particular case of a speech-based communication, the computer implements a *Spoken Dialogue System* (SDS). When one of the agents is an SDS, the dialogue consists of a sequence of *utterances* exchanged at each turn. A *spoken utterance* is the acoustic realisation of the *intentions* or *concepts* (or *dialog acts*, *communicative acts*) one of the agents wants to communicate to the other and is expressed as a *word* sequence. The amount of time between one communication and the other can be of variable length and is called a *turn*.

2.2 Formal description of man-machine spoken dialog

So as to use statistical machine learning for SDS strategy optimization, one needs to describe a spoken dialogue in terms of a finite number of variables. A man-machine spoken dialog will therefore be considered as a sequential (turn-taking) process in which a human user and a Dialog Manager (DM) communicate using spoken utterances passing through speech and language processing modules (Fig.2). A Knowledge Base (KB) is usually connected to the DM and contains information about the task addressed by the system (i.e. a list of songs).

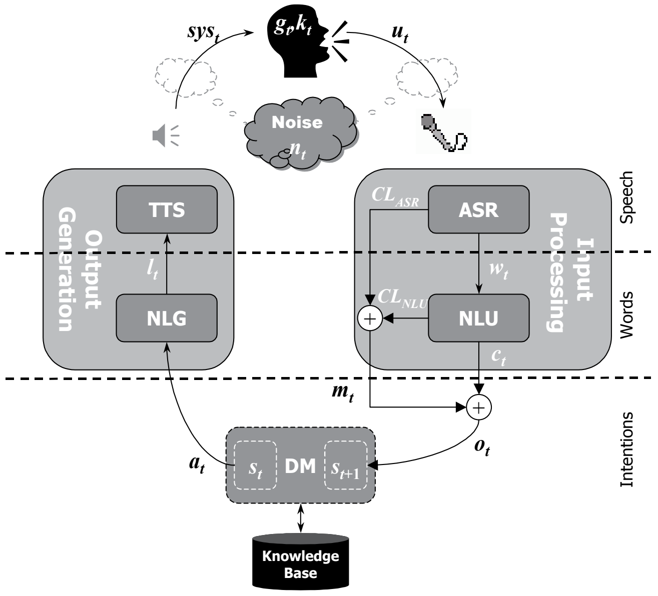


Fig. 2. Man-Machine Spoken Communication

The DM implements the SDS strategy or *policy* π defining a mapping between the DM internal state and dialogue acts (the way to build the DM internal state will be discussed later). It thus takes decisions about what to *say* at a given time. Dialogue acts can be of different kinds: providing information to the user, asking for more information, closing the dialogue, etc. The DM decisions (so its policy) are of course of a major importance since they make the interaction going in one direction or another. Adopting the system's point of view, the information exchange typically starts at turn t with the generation of a communicative act $\boldsymbol{a_t}$ by the DM. This act is generated according to the DM's strategy π_t and internal state $\boldsymbol{s_t}$ at turn t, and has to be transformed in a spoken output. A Natural Language Generation (NLG) module converts this act into a linguistic representation $\boldsymbol{l_t}$ (generally a text) which in turn serves as an input to a Text-to-Speech (TTS) system. The output of the TTS module is a spoken utterance $\boldsymbol{sys_t}$ addressed to the user. From this, the human user produces a new spoken utterance $\boldsymbol{u_t}$ taking into account his/her understanding of sys_t but also to his/her background knowledge $\boldsymbol{k_t}$ (about the task, the interaction history, the world in general) and finally to the goal $\boldsymbol{g_t}$ s/he is pursuing while interacting with the system. Both utterances sys_t and u_t can be mixed with some additional environmental noise $\boldsymbol{n_t}$. This potentially noisy user utterance is then processed by an ASR system which output is a sequence of words $\boldsymbol{w_t}$ as well as a confidence level $\boldsymbol{CL_{ASR}}$ associated to this result. The sequence w_t is usually taken out of a so called "Nbest list" ranking the best hypotheses the system can make about what the user said given the speech signal. The confidence level is usually a number between 0 and 1 providing information about the confidence the systems in the result of its processing.

It can also be a real number, depending on the system. Finally, the Natural Language Understanding (NLU) module generates a set of concepts (or communicative acts) $\boldsymbol{c_t}$ also picked from a "Nbest list" derived from $\boldsymbol{w_t}$ and again with a confidence level $\boldsymbol{CL_{NLU}}$. The observation $\boldsymbol{o_t}$ passed to the DM is actually the set $\{c_t, CL_{ASR}, CL_{NLU}\}$. From this observation, a new internal state is computed by the DM which will be used to generate a new dialog act $\boldsymbol{a_{t+1}}$. A new cycle is then started again until the end of the dialogue. This can occur when the user reached his/her goal or whenever the user or the system wants to stop the interaction for any reason (dissatisfaction, looping dialogue etc.)

2.3 Reinforcement learning and Markov decision processes

From the former description of a spoken dialogue system, it is clear that optimizing a SDS is about implementing an optimal strategy into the dialogue manager. Adopting a machine learning point of view, automatic optimization of a strategy is addressed by Reinforcement Learning (RL). The general purpose of a RL agent is to optimally control a stochastic dynamic system. The control problem is then described in terms of states, actions and rewards. In this framework, an artificial agent tries to learn an optimal control policy through real interactions with the system. It observes the state s of the system through an observation o and chooses an action a to apply on it accordingly to a current internal policy π mapping states to actions. A feedback signal r is provided to the agent after each interaction as a reward information, which is a local hint about the quality of the control. This reward is used by the agent to incrementally learn the optimal policy, simply by maximizing a function of the cumulative rewards.

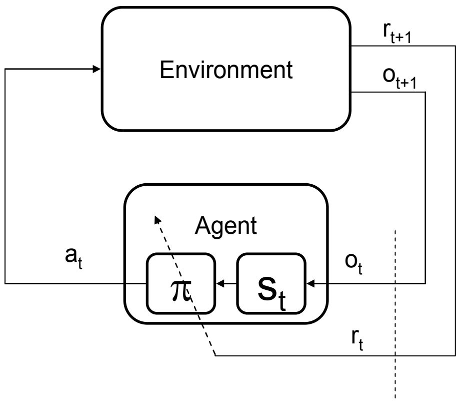


Fig. 3. Reinforcement Learning paradigm

This can be put into the formalism of Markov Decision Processes (MDP), where a discrete-time system interacting with its stochastic environment through actions is described by a finite or infinite number of states $\{s_i\}$ in which a given number of actions $\{a_j\}$ can be performed. To each state-action pair is associated a transition probability $\mathcal{T}$ giving the probability of stepping from state s at time t to state s' at time t+1 after having performed action a when in state s. To this transition is also associated a reinforcement signal (or reward) r_{t+1} describing how good was the result of action a when performed in state s. Formally, an MDP is thus completely defined by a 4-tuple $\{S, A, \mathcal{T}, \mathcal{R}\}$ where S is the state space, A is the action set, $\mathcal{T}$ is a transition probability distribution over the state space and $\mathcal{R}$ is the expected reward distribution. The couple $\{\mathcal{T}, \mathcal{R}\}$ defines the dynamics of the system:

$$\mathcal{T}_{ss'}^{a} = P\left(s_{t+1} = s' \mid s_t = s, a_t = a\right) \tag{1}$$

$$\mathcal{R}_{ss'}^{a} = E\left[r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right] \tag{2}$$

These expressions assume that the Markov property is met, which means that the system's functioning is fully defined by its one-step dynamics and that its behavior from state *s* will be identical whatever the path followed before reaching *s*. To control a system described as an MDP (choosing actions to perform in each state), one would need a *strategy* or *policy* π mapping states to actions: $\pi(s) = P(a \mid s)$ (or $\pi(s) = a$ if the strategy is deterministic).
In this framework, a RL *agent* is a system aiming at optimally mapping states to actions, that is finding the best strategy π^* so as to maximize, for each state, an overall return *R* which is a function (most often a discounted return is used i.e. a weighted sum of immediate rewards) of all the immediate rewards r_t.

$$R^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_t = \pi(s_t)\right] \tag{3}$$

$$\pi^*(s) = \underset{\pi}{\arg\max}\left[E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_t = \pi(s_t)\right]\right] \tag{4}$$

where γ is a discount factor ($0 < \gamma \leq 1$). If the probabilities of equations (1) and (2) are known, an analytical solution can be computed by resolving the Bellman equations using dynamic programming (Bertsekas 1995), otherwise the system has to learn the optimal strategy by a trial-and-error process.
To do so, a standard approach is to model the knowledge of the agent as a so-called Q-function mapping state-action pairs to an estimate of the expected cumulative reward. The optimal Q-function maps each state-action pair to its maximum expected cumulative rewards and the role of the agent can therefore be summarized as learning this function through interactions.

$$Q^{\pi}(s,a) = \int_{S} p(z|s,a)\Big(r(s,a,z) + \gamma Q^{\pi}(z,\pi(z))\Big)\,dz \tag{5}$$

Different techniques are described in the literature and in the following the Watkin's Q(λ) algorithm (Watkin 1989) will be used. This algorithm performs the following update after each interaction:

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha\left(r + \gamma \max_{b \in A} \hat{Q}(s',b) - \hat{Q}(s,a)\right) \tag{6}$$

where α is a learning rate ($0 < \alpha \leq 1$). This algorithm has been proven to converge towards the optimal solution.

3. Human-machine dialogue and Markov decision process

A first requirement to use machine learning methods such as reinforcement learning for SDS optimization is to describe a man-machine dialogue in terms of random variables and probabilities. To do so, given the description of section 2.2, we adopt the dialogue manager point of view from which the interaction can probabilistically be described by the joint probability of the signals a_t, o_t and s_{t+1} given the history of the interaction (Pietquin 2005):

$$\begin{aligned} P(s_{t+1}, o_t, a_t \mid s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0) &= \underbrace{P(s_{t+1} \mid o_t, a_t, s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0)}_{\text{Task Model}} \cdot \\ \underbrace{P(o_t \mid a_t, s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0)}_{\text{Environment}} &\cdot \underbrace{P(a_t \mid s_t, n_t, a_{t-1}, s_{t-1}, n_{t-1}, \ldots, a_0, s_0, n_0)}_{\text{DM}} \end{aligned} \quad (7)$$

In (7), the *task model* term aims at describing the way the dialogue manager builds its internal state thanks to the perceived observation, the second term stands for the environment's response to the dialogue manager's stimulation, and the last stands for the dialogue manager decision process or strategy.

3.1 Markov property and random noise

As said in section 2.3, the Markov property has to be met so as to apply standard reinforcement learning methods for strategy optimization. In the case of a SDS, the Markov property implies that the dialogue manager choice about the communicative act a_t to choose at time t and the according transition probability for stepping to internal state s_{t+1} at time t+1 are only a function of the state s_t at time t and not of the history of interactions. It can easily be met by a judicious choice of the DM state representation, which should embed enough *information* about the history of the interaction into the current state description. Such a state representation is said *informational*.

This can be easily illustrated on a simple train ticket booking system. Using such a system, a customer can book a ticket by providing orally information about the cities of departure and arrival and a desired time of departure. Three bits of information (sometimes called *attributes*) have therefore to be transferred from the human user (or caller) to the system. The problem can be seen as filling a 3-slot form. From this, a very simple way to build the state space is to represent the dialogue state as a vector of three Boolean values (e.g. [dep arr time]) set to *true* if the corresponding attribute is considered as transferred to the system and to *false* otherwise. Table 1 shows an ideal dialogue for such an application with the associated dialogue state evolution.

Speaker	Spoken Utterance	Dialogue state
System	Hello, how may I help you?	[*false false false*]
User	I'd like to go to Edinburgh.	
System	What's your departure city?	[*false true false*]
User	I want to leave from Glasgow.	
System	When do you want to go from Glasgow to Edinburgh?	[*true true false*]
User	On Saturday morning.	
System	Ok, seats are available in train n° xxx …	[*true true true*]

Table 1. Ideal dialogue in a train ticket booking application

To assume the Markov property is met using this state representation, one have make the assumption that the system adopts the same behaviour whatever the order in which the slots where filled (and by the way, whatever the values of the attributes). The Markov assumption is also made about the environment; that is the user behaves the same whatever the filling order as well. These are of course strong assumptions but we will see later that they lead to satisfactory results.
Finally, most often the noise is considered as being random so as to have independence between n_t and n_{t-1}. Eq. (5) then simplifies as follow:

$$P(s_{t+1}, o_t, a_t \mid s_t, n_t) = \underbrace{P(s_{t+1} \mid o_t, a_t, s_t, n_t)}_{\text{Task Model}} \cdot \underbrace{P(o_t \mid a_t, s_t, n_t)}_{\text{Environment.}} \cdot \underbrace{P(a_t \mid s_t, n_t)}_{\text{DM}} \tag{8}$$

3.2 Dialogue management as an MDP

From paragraph 2.2, the observation o_t can be regarded as the result of the processing of the DM dialog act a_t by its *environment*. This point of view helps putting dialogue management optimization into the MDP framework. As depicted on Fig. 2, a task-oriented (or goal-directed) man-machine dialogue can be regarded as a turn-taking process in which a user and a dialogue manager exchange information through different channels processing speech inputs and outputs (ASR, TTS ...). The dialogue manager's action (or dialogue act) selection *strategy* has to be optimized; the dialogue manager should thus be the learning *agent*.

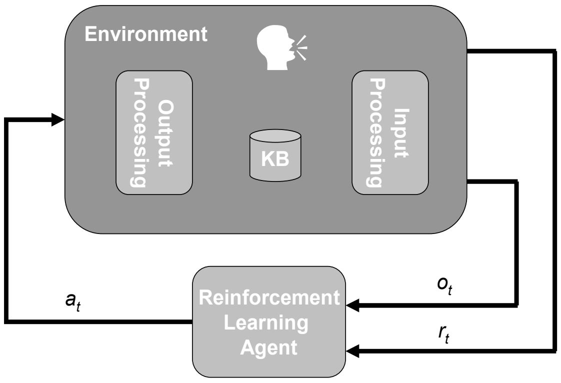


Fig. 4. Dialogue management as an MDP

The *environment* modeled by the RL agent as an MDP includes everything but the dialogue manager (see Fig. 4), i.e. the human user, the communication channels (ASR, TTS …), and any external information source (database, sensors etc.). In this context, at each turn *t* the dialogue manager has to choose an *action* a_t according to its interaction *strategy* so as to complete the task it has been designed for. The RL agent has therefore to choose an action among greetings, spoken utterances (constraining questions, confirmations, relaxation, data presentation etc.), database queries, dialogue closure etc. They result in a response from the DM environment (user speech input, database records etc.), considered as an observation o_t, which usually leads to a DM *internal state* update according to the task model (Eq. 8).

3.3 Reward function

To entirely fit to the Reinforcement Learning formalism, the previous description is still missing a *reward signal* r_t. Different ideas could lead to the building of this signal such as the amount of money saved by using a SDS instead of having human operators or the number of people hanging off before the end of the interaction etc. Singh *et al* in 1999 proposed to use the contribution of an action to the user's satisfaction. Although this seems very subjective, some studies have shown that such a reward could be approximated by a linear combination of the task completion (TC) and objective measures c_i related to the system performances. It is the PARADISE paradigm proposed in Walker *et al* 1997:

$$r_t = \alpha \cdot \mathcal{N}(TC) - \sum_i w_i \cdot \mathcal{N}(c_i), \tag{9}$$

where N is a Z-score normalization function that normalises the results to have mean 0 and standard deviation 1 and w_i are non-zero weights. Each weight (α and w_i) thus expresses the relative importance of each term of the sum in the performance of the system. There are various ways to associate an objective measure to the task completion. For example the kappa (κ) coefficient (Carletta 1996) is defined as:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}, \tag{10}$$

where $P(A)$ is the proportion of correct interpretations of user's utterances by the system and $P(E)$ is the proportion of correct interpretations occurring by chance. One can see that $\kappa = 1$ when the system performs perfect interpretation ($P(A) = 1$) and $\kappa = 0$ when the all the correct interpretations were obtained by chance ($P(A) = P(E)$).
The weights α and w_i are obtain by asking a large number of users to use a prototype system and to answer a satisfaction survey containing around 9 statements on a five-point Likert scale. The overall satisfaction is computed as the mean value of collected ratings. The objective costs c_i are measured during the interaction. A Multivariate Linear Regression is then applied using the results of the survey as the dependent variable and the weights as independent variables. In practice, the significant performance measures c_i are mainly the duration of the dialogue and the ASR and NLU performances.

3.4 Partial observability

When a direct mapping between states and observations exists, building the task model (eq. 8) is straightforward. Yet, it is rarely the case that the observations can directly be translated into dialogue states. Indeed, the real dialogue state (which we have chosen informational) at time t is related to the information the user *intended* to transmit to the system until time t during the interaction. The statistical speech recognition and understanding systems processing the user speech inputs are error prone and it can occur that the observation doesn't contain only the information meant by the user but a probability distribution over a set of possible bits of information. Indeed, as said before, the output of a speech recognition system can be a list of N word sequences (named N-best list), each of them being associated with a confidence level. This can be considered as a probability of the word sequence being correct given the spoken utterance (and maybe the context). This N-bests list serves as an input to the natural language understanding module which in turn provides a list of concept sequences associated to confidence levels.
This is typically what happens in partially observable environments where a probability distribution is drawn over possible states given the observations. An observation model is

therefore usually required. It is what we have called the task model in eq. 8 which can be a real probability distribution. For this reason, emerging research is focused on the optimization of spoken dialogue systems in the framework of Partially Observable Markov Decision Processes (POMDPs) (Poupart *et al* 2005, Young 2006)

4. Learning dialogue policies using simulation

Using the framework described previously, it is theoretically possible to automatically learn spoken dialogue policies allowing natural conversation between human users and computers. This learning process should be realised online, through real interactions with users. One could even imagine building the reinforcement signal from direct queries to the user about his/her satisfaction after each interaction (Fig. 5).

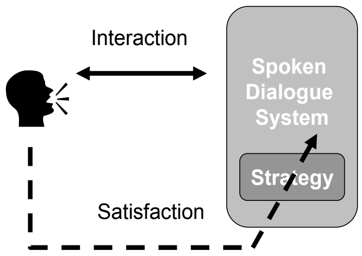


Fig. 5. Ideal learning process

For several reasons, direct learning through interactions is made difficult. First, a human user would probably react badly to some of the exploratory actions the system would choose since they might be completely incoherent. Anyway a very large number of interactions are required (typically tens of thousands of dialogues for standard dialogue systems) to train such a system. This is why data driven learning as been proposed so as to take advantage of existing databases for bootstrapping the learning process. Two methods were initially investigated: learning the state transition probabilities and the reward distribution from data (Singh *et al*, 1999) or learning parameters of a simulation environment mainly reproducing the behaviour of the user (Levin *et al* 2000). The second method is today preferred (Fig. 6). Indeed, whatever the data set available, it is unlikely that it contains every possible state transitions and it allows exploring the entire spaces. Dialogue simulation is therefore necessary for expanding the existing data sets and learning optimal policies. Another track of research is dealing with generalization to unseen situation. In this case, instead of simulating unseen situations, machine learning generalization methods are used to compute a Q-function over the entire state space with only a finite set of samples (Henderson *et al* 2005).

Most often, the dialogue is simulated at the intention level rather than at the word sequence or speech signal level, as it would be in the real world. An exception can be found in (Lopez Cozar *et al* 2003). Here, we regard an intention as the minimal unit of information that a dialogue participant can express independently. Intentions are closely related to concepts, speech acts or dialogue acts. For example, the sentence "I'd like go to Edinburgh" is based on the concept go(Edinburgh). It is considered as unnecessary to model environment behavior at a lower level, because strategy optimization is a high level concept. Additionally, concept-based communication allows error modeling of all the parts of the system, including natural language understanding (Pietquin & Renals 2002, Pietquin & Dutoit 2006b). More pragmatically, it is simpler to automatically generate concepts compared with word sequences (and certainly speech signals), as a large number of utterances can express the same intention while it should not influence the dialogue manager strategy. Table 2 describes such a simulation process. The intentions have been expanded in the last column for comprehensiveness purposes. The signals column refers to notations of section 2.2.

Signals	**Intentions**	**Expanded Intentions**
$\mathbf{sys}_0$	greeting	*Hello! How may I help you?*
$\mathbf{u}_0$	arr_city = 'Paris'	I'd like to go to Paris.
$\mathbf{sys}_1$	const(arr_time)	*When do you prefer to arrive?*
$\mathbf{u}_1$	arr_time = '1.00 PM'	I want to arrive around 1 PM.
$\mathbf{sys}_2$	rel(arr_time)	*Don't you prefer to arrive later?*
$\mathbf{u}_2$	rel = *false*	No.
$\mathbf{sys}_3$	conf(arr_city)	*Can you confirm you want to go to Paris?*
$\mathbf{u}_3$	conf = *true*	Yes !
…	…	…
…	…	…

Table 2. Simulated dialogue at the intention level ('const' stands for constraining question, 'rel' for relaxation and 'conf' for confirmation)

This approach requires modelling the environment of the dialogue manager as a stochastic system and to learn the parameters of this model from data. It has been a topic of research since the early 2000's (Levin *et al* 2000, Scheffler & Young 2001, Pietquin 2004). Most of the research is now focused on simulating the user (Georgila *et al* 2005, Pietquin 2006a, Schatzmann *et al* 2007a) and assessing the quality of a user model for training a reinforcement learning agent is an important track (Schatzmann *et al 2005,* Rieser & Lemon 2006, Georgila *et al* 2006). Modelling the errors introduced by the ASR and NLU systems is also a major topic of research (Scheffler & Young 2001, Lopez Cozar *et al* 2003, Pietquin & Beaufort 2005, Pietquin & Dutoit 2006b).

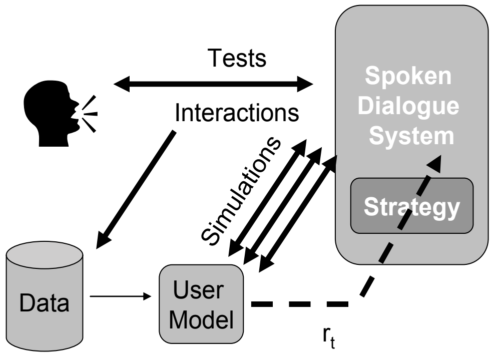


Fig. 6. Learning via simulation

4.1 Probabilistic user simulation

According to the conventions of Fig. 2 and omitting the t indices, the user behavior is ruled by the following joined probability that can be factored and simplified:

$$\begin{aligned} P(u,g,k|sys,a,s,n) &= \underbrace{P(k \mid sys,a,s,n)}_{\text{Knowledge Update}} \cdot \underbrace{P(g \mid k,sys,a,s,n)}_{\text{Goal Modification}} \cdot \underbrace{P(u \mid g,k,sys,a,s,n)}_{\text{User Output}} \\ &= \underbrace{P(k \mid sys,s,n)}_{\text{Knowledge Update}} \cdot \underbrace{P(g \mid k)}_{\text{Goal Modification}} \cdot \underbrace{P(u \mid g,k,sys,n)}_{\text{User Output}} \end{aligned}$$

These terms emphasize on the relation existing between the user's utterance production process and his/her goal and knowledge, themselves linked together. The knowledge can be modified during the interaction through the speech outputs produced by the system. Yet, this modification of the knowledge is incremental (it is an update) and takes into account the last system utterance (which might be misunderstood, and especially in presence of noise) and the previous user's knowledge state. This can be written as follow with $\boldsymbol{k}^-$ standing for k_{t-1}:

$$\begin{aligned} P(k \mid sys,s,n) &= \sum_{k^-} P(k \mid k^-,sys,s,n) \cdot P(k^- \mid sys,s,n) \\ &= \sum_{k^-} P(k \mid k^-,sys,n) \cdot P(k^- \mid s) \end{aligned}$$

The parameter of this model can be learnt from data. In (Pietquin & Dutoit, 2006b), this model serves as a basis to define a Dynamic Bayesian Network (DBN) (Fig. 7). This allows using standard DBN tools to simulate a user model and to learn the parameters from data.
Although the user's knowledge k^- is not directly dependent of the system state s, we kept this dependency in our description so as to be able to introduce a mechanism for user knowledge inference from system state because it is supposed to contain information about the history of the dialogue. This mechanism can actually be used to introduce grounding (Clarck et Shaefler, 1989) subdialogs in the interaction so as to obtain a good connection between the user's understanding of the interaction and the system view of the same interaction (Pietquin, 2007).

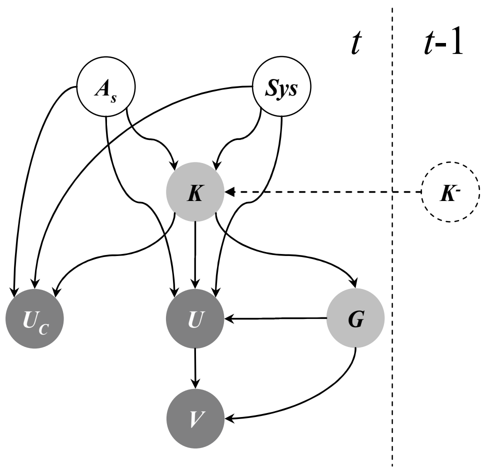


Fig. 7. DBN-based user model

4.2 Attribute-Value variable representation

It is quite unclear how to model each variable present in this description (such as u_t, sys_t, g_t etc.) for computer-based HMD simulation. As said before, it is often argued that *intention-based* communication is sufficient to internally model dialogs. Variables can then be regarded as finite sets of abstract concepts, related to the specific task, that have to be manipulated along the interactions by the SDS and the user. For this reason, we opted for a variable representation based on Attribute-Value (AV) pairs. This representation allows very high-level considerations (attributes are regarded as concepts) while values (particular values for the concepts) allow to some extent to come back to lower levels of communication. This variable description is founded on an Attribute-Value-Matrix (AVM) representation of the task (Walker *et al*, 1999)

Each communicative act is then symbolized by a set of AV pairs. From now on, we will denote $\boldsymbol{A}$ the set of possible attributes (concepts) according to the task, and by $\boldsymbol{V}$ the set of all possible values. The system utterances *sys* are then modeled as sets of AV pairs in which the attribute set will be denoted $\boldsymbol{Sys}=\{sys^{\sigma}\} \subset A$ and the set of possible values for each attribute sys^{σ} will be denoted $\boldsymbol{V^{\sigma}} = \{\mathrm{v}_i^{\sigma}\} \subset V$. The system utterance attribute set contains a special attribute $\boldsymbol{A_S}$ which values define the type of the embedded act. Allowed types can be constraining questions, relaxing prompts, greeting prompts, assertions, confirmation queries, etc. The user's utterance u is modeled as a set of AV pairs (transmitted to the ASR model) in which attributes belong to $\boldsymbol{U} = \{u^{\upsilon}\} \subset A$ and the set of possible values for u^{υ} is $\boldsymbol{V^{\upsilon}} = \{\mathrm{v}_i^{\upsilon}\} \subset V$. The user's utterance attribute set contains a special attribute $\boldsymbol{C_U}$ which value is a Boolean indicating whether the user wants to close the dialog or not. The ASR process results in an error-prone set of AV pairs w which is in turn processed and possibly modified by the NLU model. This process provides a new AV pair set c, which is part of the observation o. The user's goal $\boldsymbol{G} = \{[g^{\gamma}, \mathrm{gv}_i^{\gamma}]\}$ and the user's knowledge $\boldsymbol{K} = \{[k^{\kappa}, \mathrm{kv}_i^{\kappa}]\}$ are also AV pair sets where g^{γ} and k^{κ} are attributes and where gv_i^{γ} and kv_i^{κ} are values.

5. Experiment

This model was developed in the aim of being used in an optimal dialog strategy learning process. We therefore show here a use case of dialog simulation for Reinforcement-Learning (RL) agent training on a simple form-filling dialog task. To do so, a reward function (or reinforcement signal) $\boldsymbol{r_t}$ has to be defined. This reward provides information about the quality of each DM decision of performing an action a when in state s at time t. It is generally considered that the contribution of each action to the user's satisfaction is the most suitable reward function (Singh *et al,* 1999). According to (Walker *et al*, 1997), the major contributors to user's satisfaction are the dialog time duration (which can be approximated by the number of dialog turns $\boldsymbol{N}$), the ASR performances (which we will approximate by a confidence level $\boldsymbol{CL}$ as in (Pietquin & Renals, 2002) and the task completion ($\boldsymbol{TC}$). For this reason, we chose a reward function of the form:

$$r_t = w_{TC} \cdot TC + w_{CL} \cdot CL - w_N \cdot N$$

where w_x are positive tunable weights.

The task is a simplified version of a train ticket booking system that aims at delivering train tickets corresponding to a particular travel. Users are invited to provide information about the departure city (over 50 possible options) and time (over 24 possible options) as well as the destination city and time. The desired class (2 options) is also requested. Table 3 shows the task structure, the user's goal structure (AV pairs) and the knowledge structure which will be simply a set of counters associated to each goal AV pair and incremented each time the user answers to a question related to a given attribute during the dialog. The task completion is therefore measured as a ratio between the common values in the goal and the values retrieved by the system after the dialog session. The simulation environment includes the DBN user model, and an ASR model like in (Pietquin & Renals, 2002).
The RL paradigm requires the definition of a state space. It will be defined by a set of state variables which are 5 Booleans (one for each attribute in the task) set to *true* when the corresponding value is known, 5 status Booleans set to *true* if the corresponding value is confirmed and 5 binary values indicating whether the Confidence Level (CL) associated to the corresponding value is *high* or *low*. Every combination is not possible and the state space size is therefore of 52 states. The DM will be allowed 5 action types: greeting, open question (about more than 1 attribute), closed question (about only 1 attribute), explicit confirmation, closing.

Task		User Goal (*G*)		Knowledge (*K*)	
Attributes (A)	#V	Att.	Value	Count	init
dep	50	g^{dep}	Glasgow	k^{dep}	0
dest	50	g^{dest}	Edinburgh	k^{dest}	0
t_dep	24	g^{t_dep}	8	k^{t_dep}	0
t_dest	24	g^{t_dest}	12	k^{t_dest}	0
class	2	g^{class}	1	k^{class}	0

Table 3. AV representation of the task

Performance	
N_U	TC
5.39	0.81

Strategy				
greet	constQ	openQ	expC	close
1.0	0.85	1.23	1.31	1.0

Table 4. Experimental results

The results of the learning process on 10^5 dialogs shown in Table 4 can be interpreted as follow. This experiment shows that, in our model, the user's satisfaction relies as much on the duration time as on the task completion. Thus dialogues are short, but task completion is not optimal since one attribute is often missing in the presented data (one of the cities in general). There are more open-ended questions than constraining questions. Actually, constraining questions are present because sometimes only one argument is missing and there is no need of an open-ended question to retrieve it. Yet, there are explicit confirmations because the task completion is a factor of user satisfaction. It actually illustrates well the trade-off between task completion and time duration. This behaviour can be tuned by changing the parameters of our user model for example.

6. Conclusion

In this chapter, a formal probabilistic description of human-machine dialogues was described. This description allowed putting the optimization of spoken dialogue strategies in the framework of reinforcement learning. Reinforcement learning designates a very data-demanding class of machine learning methods. This is a major problem for SDS optimization since collecting and annotating data is very difficult. To solve this problem of data sparsity, dialogue simulation techniques are commonly used. A specific simulation framework based on a probabilistic description of the user's behavior has been described. It can easily be translated into a dynamic Bayesian network and use the standard parameter learning and inference tools. The reinforcement learning framework also requires the definition of a reward function associating a numerical number to each system action. To do so, the PARADISE framework using multivariate regression has been described. To summarize, this chapter has shown that a large number of machine learning methods can be used in the context of spoken dialogue optimization. Among these techniques, reinforcement learning, Bayesian inference and multivariate regression are very common.

7. Future works

Statistical machine learning for spoken dialogue strategies optimization is an emerging area of research and lots of issues still remain. One of the first, which is common to a lot of reinforcement learning applications, is to find tractable algorithms for real size dialogue systems. The standard RL algorithms are indeed suitable for small tasks such as described in section 5. Yet real applications can exhibit up to several million of states, possibly with continuous observations (Williams *et al* 2005). Supervised learning (Henderson *et al* 2005) and hierarchical learning (Cuayáhuitl *et al* 2007) have been recently proposed to tackle this problem.

In this chapter, we have essentially considered the problem of completely observable systems. But as said in paragraph 3.4, a spoken dialogue system should be considered as partially observable, because of error prone speech processing sub-systems. Research on POMDP for SDS optimization are reported in (Poupart *et al* 2005, Young 2006), yet a lot of work is still necessary to anchor SDS in real life.

Spoken dialogue simulation is also the topic of ongoing research. Different approaches are being studied such as the recently proposed agenda-based user model (Schatzmann *et al* 2007b) that can be trained by an Expectation-Maximisation algorithm from data, or user models based on dynamic Bayesian networks (Pietquin & Dutoit 2006a) such as those presented in this chapter. One of the major argument against the current simulation methods is the lack of assessment methods even though some work can be cited (Schatzmann *et al* 2005, Georgila *et al* 2006, Rieser & Lemon 2006).

On another hand, it might be interesting to see how to use learned strategies to help human developers to design optimal strategies. Indeed, the solution may be in computer-aided design more than fully automated design (Pietquin & Dutoit 2003).

The ultimate aim of this research area is to design a complete data-driven dialogue system using an end-to-end probabilistic framework, from speech recognition to speech synthesis systems automatically trained on real data, is probably the next step (Lemon & Pietquin 2007).

8. Acknowledgement

The research presented in this chapter has been funded by the 'First Europe' program of the Belgian Walloon Region, the SIMILAR European Network of Excellence and the French Lorraine Region.

9. References

Allen, J. (1994) *Natural Language Understanding*, Benjamin Cummings, 1987, Second Edition, 1994.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific, 3rd edition.

Carletta J. (1996), Assessing Agreement on Classification Tasks: the Kappa Statistic. *Computational Linguistics*, 22(2), 1996, 249-254.

Clarck H. and Schaefer E., "Contributing to discourse," Cognitive Science, vol. 13, pp. 259–294, 1989.

Cuayáhuitl, H.; Renals, S.; Lemon, O. and Shimodaira, H. (2007) Hierarchical Dialogue Optimization Using Semi-Markov Decision Processes, in *Proceedings of International Conference on Speech Communication (Interspeech'07)*, Anvers (Belgium), 2007.

Dutoit, T., *An Introduction to Text-To-Speech Synthesis*. Kluwer Academic Publishers, Dordrecht, ISBN 0-7923-4498-7, 1997.

Frampton, M. & Lemon O. (2006). Learning more effective dialogue strategies using limited dialogue move features, in *Proceedings of ACM*, 2006.

Georgila, K.; Henderson, J. and Lemon, O. (2005). Learning User Simulations for Information State Update Dialogue Systems, in *Proceedings of International Conference on Speech Communication (Interspeech'05)*, Lisbon (Portugal) 2005.

Georgila, K.; Henderson, J. and Lemon, O. (2006) User simulation for spoken dialogue systems: Learning and evaluation, in *Proceedings of International Conference on Speech Communication (Interspeech'06)*, Pittsburgh, 2006.

Graesser, A.; VanLehn, K.; Rosé, C.; Jordan, P. & Harter, D. (2001) Intelligent Tutoring Systems with Conversational Dialogue. in *AI Magazine* vol. 22(4) , 2001, pp. 39-52.

Henderson, J.; Lemon, O. and Georgila, K. (2005) Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data, in *Proceedings of the IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2005, pp. 68–75.

Lemon, O. & Pietquin, O. (2007). Machine learning for spoken dialogue systems, in *Proceedings of the European Conference on Speech Communication and Technologies (Interspeech'07)*, Anvers (Belgium), August 2007.

Levin, E.; Pieraccini, R. & Eckert, W. (1997). Learning dialogue strategies within the Markov decision process framework, in *Proceedings of the International Workshop on Automatic Speech Recognition and Understanding* (ASRU'97), December 1997.

Levin, E.; Pieraccini, R. and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies, in *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

Lopez-Cozar, R.; de la Torre, A.; Segura, J. and Rubio, A. (2003) Assesment of dialogue systems by means of a new simulation technique, in *Speech Communication*, vol. 40, no. 3, pp. 387–407, May 2003.

Pietquin, O. and Renals, S. (2002). Asr system modelling for automatic evaluation and optimization of dialogue systems, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'02)*, Orlando, (USA, FL), May 2002.

Pietquin, O. and Dutoit, T. (2003). Aided Design of Finite-State Dialogue Management Systems, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2003)*, Baltimore (USA, MA), 2003.

Pietquin, O. (2004). *A Framework for Unsupervised Learning of Dialogue Strategies*, Presses Universitaires de Louvain, ISBN : 2-930344-63-6, 2004.

Pietquin, O. (2005). A probabilistic description of man-machine spoken communication, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'05)*, Amsterdam (The Netherlands), July 2005.

Pietquin, O., Beaufort, R. (2005). Comparing ASR Modeling Methods for Spoken Dialogue Simulation and Optimal Strategy Learning. In *Proceedings of Interspeech/Eurospeech 2005*, Lisbon, Portugal (2005)

Pietquin, O. (2006a) Consistent goal-directed user model for realistic man-machine task-oriented spoken dialogue simulation, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'06)*, Toronto, Canada, July 2006.

Pietquin, O. (2006b). Machine learning for spoken dialogue management : an experiment with speech-based database querying, in *Artificial Intelligence : Methodology, Systems and Applications*, J. Euzenat and J. Domingue, Eds., vol. 4183 of Lecture Notes in Artificial Intelligence, pp. 172–180. Springer Verlag, 2006.

Pietquin, O. & Dutoit, T. (2006a). A probabilistic framework for dialog simulation and optimal strategy learning, in *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 2, pp. 589–599, March 2006.

Pietquin, O. and Dutoit, T. (2006b). Dynamic Bayesian networks for NLU simulation with applications to dialog optimal strategy learning, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'06)*, May 2006.

Pietquin, O. (2007), Learning to Ground in Spoken Dialogue Systems. *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, IV:165-168, Hawaii (USA), 2007

Poupart, P.; Williams, J. & Young, S. (2006). Partially observable Markov decision processes with continuous observations for dialogue management, in *Proceedings of the SigDial Workshop (SigDial'06)*, 2006.

Rabiner, L. & Juang, B.H. (1993). *Fundamentals of Speech Recognition*, Prentice Hall, Signal Processing Series, 1993.

Reiter, E. & Dale, R. (2000) *Building Natural Language Generation Systems*, Cambridge University Press, Cambridge, 2000.

Rieser, V. and Lemon, O. (2006) Cluster-based user simulations for learning dialogue strategies and the super evaluation metric, in *Proceedings of Interspeech/ICSLP*, 2006.

Schatzmann, J.; Georgila, K. and Young, S. (2005) Quantitative evaluation of user simulation techniques for spoken dialogue systems, in *Proceedings of the SIGdial'05 Workshop*, September 2005.

Schatzmann, J.; Weilhammer, K.; Stuttle, M. and Young, S. (2007a) A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies, in *Knowledge Engineering Review* 21(2): 97-126, 2007.

Schatzmann, J.; Thomson, B. and Young., S. (2007b). Statistical User Simulation with a Hidden Agenda. In *Proceedings of the 8th SigDIAL Workshop*, Antwerp, 2007.

Scheffler, K. & Young, S. (2001). Corpus-based dialogue simulation for automatic strategy learning and evaluation, in *Proc. NAACL Workshop on Adaptation in Dialogue Systems*, 2001.

Singh, S.; Kearns, M.; Litman, D. & Walker, M. (1999), Reinforcement learning for spoken dialogue systems, in *Proceedings of NIPS'99*, 1999.

Young, S. (2006). Using POMDPs for dialog management, in *Proceedings of the 1st IEEE/ACL Workshop on Spoken Language Technologies (SLT'06)*, 2006.

Young, S.; Schatzmann, J.; Weilhammer, K. & Ye, H. (2007). The hidden information state approach to dialog management, in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'07)*, April 2007.

Walker, M.; Litman, D.; Kamm, C. & Abella, A. (1997). PARADISE: A Framework for Evaluating Spoken Dialogue Agents. in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain (1997) 271-280.

Watkins, C. (1989). *Learning from delayed rewards*. PhD thesis, Psychology Department, Cambridge University, Cambridge, England, 1989.

Williams, J. & Young, S. (2005). Scaling up POMDPs for dialogue management: the summary POMDP method, in *Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU'05)*, 2005.

Williams, J.; Poupart, P. and Young, S. (2005). Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management, in *Proceedings of the 6th SigDial Workshop*, Lisbon (Portugal), 2005.

9

Hardening Email Security via Bayesian Additive Regression Trees

Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang and Suku Nair
SMU HACNet Lab, Southern Methodist University Dallas, TX, USA

1. Introduction

The changeable structures and variability of email attacks render current email filtering solutions useless. Consequently, the need for new techniques to harden the protection of users' security and privacy becomes a necessity. The variety of email attacks, namely *spam*, damages networks' infrastructure and exposes users to new attack vectors daily. Spam is unsolicited email which targets users with different types of commercial messages or advertisements. Porn-related content that contains explicit material or commercials of exploited children is a major trend in these messages as well. The waste of network bandwidth due to the numerous number of spam messages sent and the requirement of complex hardware, software, network resources, and human power are other problems associated with these attacks. Recently, security researchers have noticed an increase in malicious content delivered by these messages, which arises security concerns due to their attack potential. More seriously, *phishing* attacks have been on the rise for the past couple of years. Phishing is the act of sending a forged e-mail to a recipient, falsely mimicking a legitimate establishment in an attempt to scam the recipient into divulging private information such as credit card numbers or bank account passwords (James, 2005). Recently phishing attacks have become a major concern to financial institutions and law enforcement due to the heavy monetary losses involved. According to a survey by Gartner group, in 2006 approximately 3.25 million victims were spoofed by phishing attacks and in 2007 the number increased by almost 1.3 million victims. Furthermore, in 2007, monetary losses, related to phishing attacks, were estimated by $3.2 billion. All the aforementioned concerns raise the need for new detection mechanisms to subvert email attacks in their various forms.
Despite the abundance of applications available for phishing detection, unlike spam classification, there are only few studies that compare machine learning techniques in predicting phishing emails (Abu-Nimeh et al., 2007). We describe a new version of Bayesian Additive Regression Trees (BART) and apply it to phishing detection. A phishing dataset is constructed from 1409 raw phishing emails and 5152 legitimate emails, where 71 features (variables) are used in classifiers' training and testing. The variables consist of both textual and structural features that are extracted from raw emails. The performance of six classifiers, on this dataset, is compared using the area under the curve (AUC) (Huang & Ling, 2005). The classifiers include Logistic Regression (LR), Classification and Regression Trees (CART), Bayesian Additive Regression Trees (BART), Support Vector Machines (SVM), Random

Forests (RF), and Neural Networks (NNet). In addition to the AUC, additional measures are used to gauge the classifiers' performance, such as the error rate, false positive, and false negative rates.

1.1 Motivation to Bayesian methodology

We start by providing a discussion on Bayesian learning and the reasons behind choosing BART among another classifiers, then we illustrate the technical details of BART. BART is a Bayesian approach, thus it inherits all the advantages from Bayesian learning. There are various advantages of Bayesian learning when compared to other statistical approaches. As opposed to the frequentist approach for defining the probability of an uncertain event, here one needs a record of past information of an event. Yet, if this information is not available, then the frequentist approach cannot be used to define the degree of belief in the uncertain event. On the other hand, the Bayesian approach allows us to reason about beliefs under conditions of uncertainty. Thus, it helps in modeling uncertainty in a probabilistic way (Neal, 1995). In addition, Bayesian inference is regarded as a good approach to tackle the problem of data modeling (Kandola, 2001). A model is designed for a particular application and adapted according to the data while the data arrives from the application. The model then provides a representation of the prior beliefs about the application and the information derived from the data (Bishop, 1995).

Another advantage of the Bayesian approach is that one does not need to update the model entirely when acquiring new knowledge. Yet, the new data can be used to update the current model instead of re-fitting the entire model. This feature comes handy especially in phishing detection since phishing attacks change frequently and vastly to lure filters and detection mechanisms. Assuming that one needs to re-fit the entire model when new batch of emails arrives, the procedure becomes very computationally extensive and time consuming, thus impractical.

Furthermore, BART is a model-based Bayesian approach. As opposed to to those algorithm-based learning methods, model-based approaches can provide full and accurate assessment of uncertainty in predictions, while remaining highly competitive in terms of predictive accuracy. In addition, model-based approaches are considered non-greedy, hence opposed to selecting the best solution at the time being and not worrying about the future (i.e. whether the solution is efficient or not), the solution is interchangeable accordingly. Also, model-based approaches are non-adhoc, hence the provided solution is not only selected to a particular problem; however, it can be used as a general case.

1.2 Why Bayesian additive regression trees?

BART automatically selects variables from a large pool of input predictors, while searching for models with highest posterior probabilities for future prediction, via a backfitting Markov chain Monte Carlo (MCMC) algorithm (see section 3.1 for further details). Compared to other Bayesian methods, such as Naive Bayes and Bayesian Networks, the latter approaches require variable selection to be done separately, otherwise they use all the variables supplied for training, thus the performance of the classifier will be very poor. Also, it is well known that variable selection in a high dimensional space is a very difficult problem that often requires intensive computations. As we mentioned earlier, phishing emails change regularly and vastly to lure detection mechanisms and the variables may change over time as well. Yet, the above nice feature of BART comes handy when training on newly arriving emails on a regular basis. With no additional requirements to perform variable selection, BART simultaneously accomplishes variable selection during the training phase.

In addition, in phishing detection hundreds of potential features are extracted from raw emails. Only an unknown subset of them are useful for prediction and others are irrelevant. Blindly including all the variables in the step of training often leads to overfitting, and hence predicting new attacks may be poor. However, with the automatic variable selection feature in BART this problem is solved.

Further, many other Bayesian learning approaches require very careful prior specification, and hence extra effort and operational cost in training models. However, BART, as shown in (Chipman et al., 2006), appears to be relatively insensitive to small changes in the prior specification and the choice of the number of trees. According to (Chipman et al., 2006), the default priors work very well, which enables BART to be an objective and automatic training procedure. This is desirable in situations in which there is no prior information available or no human intervention is preferred.

Furthermore, BART is a class of Bayesian additive models with multivariate components of binary trees. Using binary trees as model components makes BART more exible in practice, as opposed to common regression approaches, since the structure of binary trees has been proved to approximate well nonlinear and nonsmooth functional forms in many applications (Hastie et al., 2001).

Also, BART uses a sum-of-trees-model which is more exible than any single tree model that can hardly account for additive effects. Each tree component is regarded as a *weak learner*, which explains a small and different part of the unknown relationship between the input and output. In addition, multivariate components of BART can easily incorporate high-order interaction effects among three or more input variables, which can be dificult to capture by other additive models.

Moreover, the rich structure of BART leads to its excellent learning ability, even in the presence of a very complicated structure embedded in data. Since phishing emails look very similar to legitimate emails, actually they are duplicates of legitimate emails with some changes, learning is a challenging problem and perhaps involves discovering an elaborate and subtle relationship from data.

2. Related work

(Chandrasekaran et al., 2006) proposed a technique to classify phishing based on structural properties of phishing emails. They used a total of 25 features mixed between style markers (e.g. the words suspended, account, and security) and structural attributes, such as the structure of the subject line of the email and the structure of the greeting in the body. They tested 200 emails (100 phishing and 100 legitimate). They applied simulated annealing as an algorithm for feature selection. After a feature set was chosen, they used information gain (IG) to rank these features based on their relevance. They applied one-class SVM to classify phishing emails based on the selected features. Their results claim a detection rate of 95% of phishing emails with a low false positive rate.

(Fette et al., 2007) compared a number of commonly-used learning methods through their performance in phishing detection on a past phishing data set, and finally Random Forests were implemented in their algorithm PILFER. Their methods can be used to detect phishing websites as well. They tested 860 phishing emails and 6950 legitimate emails. The proposed method detected correctly 96% of the phishing emails with a false positive rate of 0.1%. They used ten features handpicked for training and their phishing dataset was collected in 2002 and 2003. As pointed out by the authors themselves, their implementation is not optimal and further work in this area is warranted.

(Abu-Nimeh et al., 2007) compared six machine learning techniques to classify phishing emails. Their phishing corpus consisted of a total of 2889 emails and they used 43 features (variables). They showed that, by merely using a *bag-of-words* approach, the studied classifiers could successfully predict more than 92% of the phishing emails. In addition, the study showed that Random Forests achieved the maximum predictive accuracy and Logistic Regression achieved the minimum false positives on the studied corpus.

3. Machine learning approaches for binary classification

In the literature, there exist several machine learning techniques for binary classification, e.g., logistic regression, neural networks (NNet), binary trees and their derivatives, discriminant analysis (DA), Bayesian networks (BN), nearest neighbor (NN), support vector machines (SVM), boosting, bagging, etc. The interested reader can refer to (Hastie et al., 2001) and the references therein for a detailed overview. Here we describe the application of Bayesian Additive Regression Trees (BART) for learning from data, combined with a probit setup for binary responses, to detect phishing emails.

Most of the machine learning algorithms discussed here are categorized as *supervised* machine learning, where an algorithm (classifier) is used to map inputs to desired outputs using a specific function. In classification problems a classifier tries to learn several features (variables or inputs) to predict an output (response). Specifically in phishing classification, a classifier will try to classify an email to phishing or legitimate (response) by learning certain characteristics (features) in the email.

Applying any supervised machine learning algorithm to phishing detection consists of two steps: training and classification. During the *training* step a set of compiled phishing and non-phishing messages (with known status) is provided as training dataset to the classifier. Emails are first transformed into a representation that is understood by the algorithms. Specifically, raw emails are converted to vectors using the vector space model (VSM) (Salton & McGill, 1983), where the vector represents a set of features that each phishing and non-phishing email carries. Then the learning algorithm is run over the training data to create a classifier. The *classification* step follows the training (learning) phase. During classification, the classifier is applied to the vector representation of real data (i.e. test dataset) to produce a prediction, based on learned experience.

3.1 Bayesian additive regression trees

Bayesian Additive Regression Trees (BART) is a new learning technique, proposed by (Chipman et al., 2006), to discover the unknown relationship between a continuous output and a dimensional vector of inputs. The original model of BART was not designed for classification problems, therefore, a modified version, hereafter CBART, which is applicable to classification problems in general and phishing classification in particular is used. Note that BART is a learner to predict quantitative outcomes from observations via regression. There is a distinction between regression and classification problems. Regression is the process of predicting quantitative outputs. However, when predicting qualitative (categorical) outputs this is called a classification problem. Phishing prediction is a binary classification problem, since we measure two outputs of email either phishing =1 or legitimate =0 (Hastie et al., 2001).

BART discovers the unknown relationship f between a continuous output Y and a p dimensional vector of inputs $x = (x_1, ..., x_p)$. Assume $Y = f(x) + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$ is the random error. Motivated by ensemble methods in general, and boosting algorithms in particular, the basic idea of BART is to model or at least approximate $f(x)$ by a sum of regression trees,

$$f(x) = \sum_{i=1}^{m} g_i(x); \qquad (1)$$

each g_i denotes a binary tree with arbitrary structure, and contributes a small amount to the overall model as a *weak learner*, when m is chosen large. An example of a binary tree structure is given in Figure 1, in which a is the root node, c is an internal node, and b, d and e are three terminal nodes that are associated with parameter μ_1, μ_2 and μ_3, respectively. Also, each of the interior (i.e., non-terminal) nodes is associated with a binary splitting rule based on some x variable. By moving downwards from the root, an observation with given x will be assigned to a unique terminal node, according to the splitting rules associated with the nodes included in its path. In consequence, the corresponding parameter of the terminal node will be the value of g for this observation.

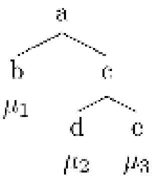


Fig. 1. A binary tree structure

Let T_i be the i^{th} binary tree in the model (1), consisting of a set of decision rules (associated with its interior nodes) and a set of terminal nodes, for i = 1, ..., m. Let M_i be the vector containing all terminal node parameters of T_i such that $M = \{M_1, ..., M_{b_i}\}$ and b_i is the number of terminal nodes that T_i has. Now we can explicitly write

$$Y = g(x; T_1, M_1) + \ldots + g(x; T_m, M_m) + \epsilon. \qquad (2)$$

Figure 2 depicts an example of a binary tree in the BART model. Note that the BART contains multiple binary trees, since it is an additive model. Each node in the tree represents a feature in the dataset and the terminal nodes represent the probability that a specific email is phishing, given that it contains certain features. For example, if an email contains HTML code, contains javascript, and the javascript contains form validation, then the probability that this email is phishing is 80% (refer to Figure 2). These features are discussed in more details in Section 4.1.1.

BART is fully model-based and Bayesian in the sense that a *prior* is specified, a *likelihood* is defined using the data, and then a sequence of draws from the *posterior* using Markov chain Monte Carlo (MCMC) is obtained. Specifically, a *prior* distribution is needed for T, M, and σ, respectively. Each draw represents a fitted model f^* of the form (1).

To specify a *prior* distribution $P(T)$ on T, one needs three pieces of information; (i) determining how likely a node will be split when a tree is created; (ii) determining which variable will be chosen to split the node; (iii) determining the rule that will be used for splitting. The main goal here is to generate small trees or *"weak learners"*, hence each tree plays a small share in the overall fit, but when combined all produce a powerful "committee".

For the *prior* distribution on terminal node parameters $P(\mu)$, the parameters of the terminal nodes are assumed independent *a priori*, hence the prior mean $E(Y|x) = \sum_{i=1}^{m} \mu_i$. Lastly, for the variance of noise σ^2, a prior $P(\sigma)$ is needed. The parameters of the prior on σ can be specified from a *least square linear regression* of Y on the original x's.

Now given the *prior* distributions a *backfitting MCMC Gibbs sampler* is used to sample from the *posterior* distribution as shown below.

Repeat $i = 1$ to I (say $I = 1000$, where I is the number of simulations):

- Sample T_j conditional on Y, all Ts but T_j, all μs, and σ.
- Sample M_j given all Ts, all Ms but M_j, and σ.
- Repeat the above steps m times for $j = 1, ., m$, where j is the total number of trees available.
- Sample σ given Y and all Ts, all Ms and σ.

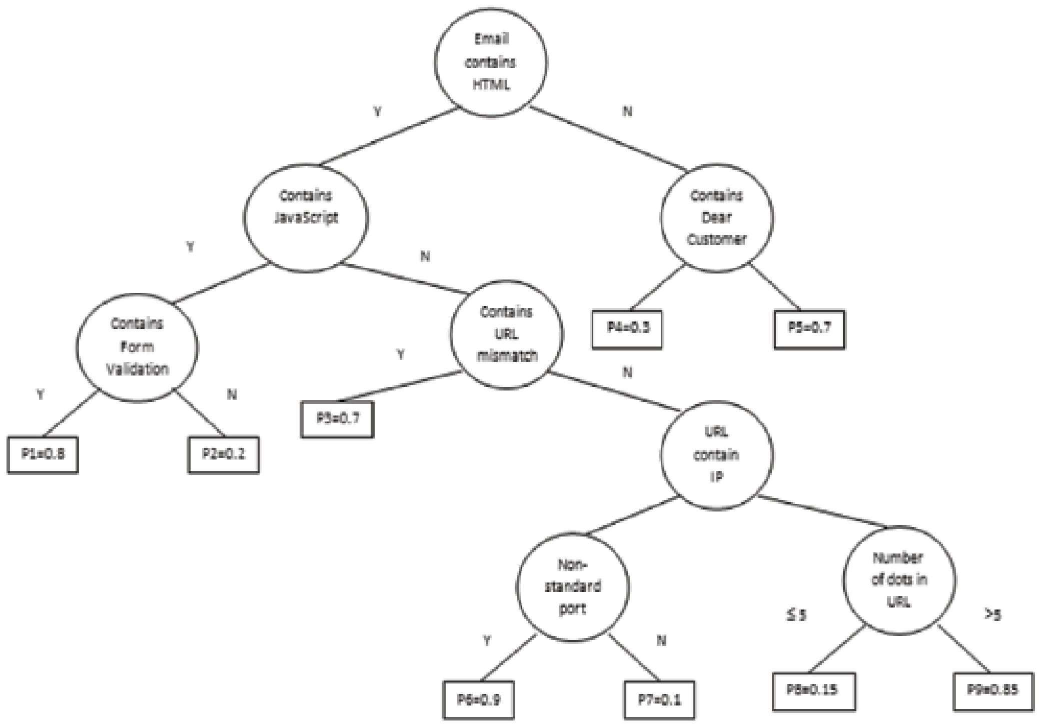


Fig. 2. Example of a binary tree.

Since this is a Markov chain, simulation i depends on simulation i - 1. The MCMC simulation changes tree structures based on a stochastic tree generating process. The structures can be changed by randomly using any of the following four actions. *Grow* can be applied to grow a new pair of terminal nodes from a terminal node and make it become an interior one. *Prune* can be applied to prune a pair of terminal nodes and make their parent node become a terminal one. *Change* is to change a splitting rule of a non-terminal node. *Swap* is to swap rules between a parent node and a child. By these changes, MCMC generates different tree structures and chooses the tree structure that provides the "best" sum-of-trees model according to posterior probabilities of trees.

It is worth mentioning that BART has several appealing features, which make it competitive compared to other learning methods and motivate our study as well. Rather than using a single regression tree, BART uses a sum- of-trees model that can account for additive effects. Also, the binary tree structure helps in approximating well nonlinear and non-smooth relationships (Hastie et al., 2001). Furthermore, BART can conduct automatic variable selection of inputs while searching for models with highest posterior probabilities during MCMC simulation. In addition, by applying Bayesian learning, BART can use newly coming data to update the current model instead of re-fitting the entire model.

Despite the advantages mentioned earlier, it is well known that a Bayesian approach usually brings heavy computation time due to its nature. Predicting the *posterior* probabilities via MCMC is usually time consuming and requires complex computations.

3.1.1 BART for classification (CBART)

As mentioned in Section 3.1, BART requires the output variable to be continuous, instead of binary. Let $Y = 1$ if an email is phishing; otherwise $Y = 0$. To use BART with binary outputs, we introduce a latent variable Z in connection with Y in spirit of (Albert & Chib, 1993), by defining

$$Z = f(x) + \epsilon, \quad \epsilon \sim N(0,1);$$
$$Y = \begin{cases} 1 & \text{if } Z > 0; \\ 0 & \text{if } Z \leq 0. \end{cases} \tag{3}$$

where $f(x)$ is the sum-of-trees model in (1). Note here, we fix σ at 1, due to the simple binary nature of Y . This yields the probit link function between the phishing probability p and $f(x)$,

$$p \equiv P(Y = 1|x) = P(Z > 0|x) = \Phi(f(x)), \tag{4}$$

where $\Phi(\cdot)$ is the cumulative density function of $N(0, 1)$.

Under the above setup of the latent variable Z, we can use BART to learn $f(x)$ from data, after appropriately modifying the prior distribution on M and the MCMC algorithm proposed in (Chipman et al., 2006) for posterior computation. Then we can estimate $Y = 1$ if the fitted $f^*(x) > 0$, otherwise estimate $Y = 0$. Further, we can obtain the estimate of p through equation (4).

Before we describe the algorithm, let T denote a binary tree consisting of a set of interior node decision rules and a set of terminal nodes, and let $M = \{\mu_1, \mu_2, ..., \mu_b\}$ denote a set of parameter values associated with each of the b terminal nodes of T. Now we explicitly denote the ith component of the model $g_i(x)$ by $g_i(x; T_i, M_i)$. Also, let $T_{(j)}$ be the set of all trees in the sum (1) except T_j , and $M_{(j)}$ the associated terminal node parameters. Let y denote the observed phishing status of emails in the training data. The algorithm will generate draws from the posterior distribution

$$p((T_1, M_1), ..., (T_m, M_m), Z|y) \tag{5}$$

rather than drawing from

$$p((T_1, M_1), ..., (T_m, M_m), \sigma|y)$$

in the original algorithm. A typical draw from the new posterior (5) entails m successive draws of tree component(T_j, M_j) conditionally on $(T_{(j)}, M_{(j)}, Z)$:

$$\begin{array}{c} (T_1, M_1)|T_{(1)}, M_{(1)}, y, Z \\ (T_2, M_2)|T_{(2)}, M_{(2)}, y, Z \\ \vdots \\ (T_m, M_m)|T_{(m)}, M_{(m)}, y, Z \end{array} \tag{6}$$

followed by a draw of Z from the full conditional:

$$Z|(T_1, M_1), ..., (T_m, M_m), y. \tag{7}$$

Note that there is no need to draw σ in our new algorithm since it is set to 1.
We proceed to discuss how to implement (6) and (7). First, we claim that the first step is essentially the same as in the original algorithm. This is because in (6), no extra information is given by y when Z is given, since y can be completely determined by Z through (3). Hence we can remove the redundant y in (6), and use the original algorithm (substitute y by Z and set $\sigma = 1$) to draw from (6). Since Z is latent, we need an extra step to draw values of Z from (7). It can be verified that for the jth email in the training data, $Z_j \mid (T_1,M_1), ..., (T_m,M_m), y$ is distributed as $N(\sum_{i=1}^{m} g_i(x; T_i,M_i), 1)$ truncated at the left by 0 if $y_j = 1$, and distributed as $N(\sum_{i=1}^{m} g_i(x; T_i,M_i), 1)$ truncated at the right by 0 if $y_j = 0$. Thus, drawing from (7) can be easily done by drawing values from the normal distributions and then truncating them by 0 either from the right or from the left based on the value of y.
As shown above, BART is well suited for binary classification, under the probit setup with the use of the latent variable. In this case, it is even easier than before because σ is no longer an unknown parameter and the draws of Z are extremely easy to obtain.
We now briey discuss how to use BART for prediction. In an MCMC run, we can simply pick up the “best” f^* (according to posterior probabilities or Bayes factor or other criteria) from the sequence of visited models, and save it for future prediction. Note that the selected f^* perhaps involves a much less number of input variables than p since BART automatically screens input variables. This would allow prediction for a new email to be quickly done since much less information needs to be extracted from the email. A better way is to use the posterior mean of f for prediction, approximated by averaging the f^* over the multiple draws from (5), and further gauge the uncertainty of our prediction by the variation across the draws. However, this involves saving multiple models in a physical place for future use. A more realistic approach is to use the best B fitted models for prediction that account for the 95% posterior probabilities over the space of sum-of-tree models. Usually, B is a number less than 20 and again, when predicting a new email is or not, a much less number of input variables than p are expected to be used (Abu-Nimeh et al., 2008).

3.2 Classification and regression trees

CART or Classification and Regression Trees (Breiman et al., 1984) is a model that describes the conditional distribution of y given x. The model consists of two components; a tree T

with b terminal nodes, and a parameter vector $\Theta = (\theta_1, \theta_2, \dots, \theta_b)$ where θ_i is associated with the i^{th} terminal node. The model can be considered a classification tree if the response y is discrete or a regression tree if y is continuous. A binary tree is used to partition the predictor space recursively into distinct homogenous regions, where the terminal nodes of the tree correspond to the distinct regions. The binary tree structure can approximate well non-standard relationships (e.g. non-linear and non-smooth). In addition, the partition is determined by splitting rules associated with the internal nodes of the binary tree. Should the splitting variable be continuous, a splitting rule in the form $\{x_i \in C\}$ and $\{x_i \notin C\}$ is assigned to the left and the right of the split node respectively. However, should the splitting variable be discrete, a splitting rule in the form $\{x_i \leq s\}$ and $\{x_i > s\}$ is assigned to the right and the left of the splitting node respectively (Chipman et al., 1998).
CART is exible in practice in the sense that it can easily model nonlinear or nonsmooth relationships. It has the ability of interpreting interactions among predictors. It also has great interpretability due to its binary structure. However, CART has several drawbacks such as it tends to overfit the data. In addition, since one big tree is grown, it is hard to account for additive effects.

3.3 Logistic regression

Logistic regression is the most widely used statistical model in many fields for binary data (0/1 response) prediction, due to its simplicity and great interpretability. As a member of generalized linear models it typically uses the *logit* function. That is

$$log\frac{P(x;\beta)}{1-P(x;\beta)} = \beta^T x$$

where x is a vector of p predictors $x = (x_1, x_2, \dots, x_p)$, y is the binary response variable, and β is a $p \times 1$ vector of regression parameters.
Logistic regression performs well when the relationship in the data is approximately linear. However, it performs poorly if complex nonlinear relationships exist between the variables. In addition, it requires more statistical assumptions before being applied than other techniques. Also, the prediction rate gets affected if there is missing data in the data set.

3.4 Neural networks

A neural network is structured as a set of interconnected identical units (neurons). The interconnections are used to send signals from one neuron to the other. In addition, the interconnections have weights to enhance the delivery among neurons (Marques de Sa, 2001). The neurons are not powerful by themselves, however, when connected to others they can perform complex computations. Weights on the interconnections are updated when the network is trained, hence significant interconnection play more role during the testing phase. Figure 3 depicts an example of neural network. The neural network in the figure consists of one input layer, one hidden layer, and one output layer. Since interconnections do not loop back or skip other neurons, the network is called *feedforward*. The power of neural networks comes from the nonlinearity of the hidden neurons. In consequence, it is signi_cant to introduce nonlinearity in the network to be able to learn complex mappings. The commonly used function in neural network research is the *sigmoid* function, which has the form (Massey et al., 2003)

$$a(x) = \frac{1}{1 + e^{-x}}$$

Although competitive in learning ability, the fitting of neural network models requires some experience, since multiple local minima are standard and delicate regularization is required.

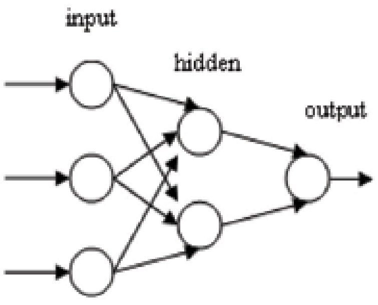


Fig. 3. Neural Network.

3.5 Random forests

Random forests are classi_ers that combine many tree predictors, where each tree depends on the values of a random vector sampled independently. Furthermore, all trees in the forest have the same distribution (Breiman, 2001). In order to construct a tree we assume that n is the number of training observations and p is the number of variables (features) in a training set. In order to determine the decision node at a tree we choose $k << p$ as the number of variables to be selected. We select a *bootstrap* sample from the n observations in the training set and use the rest of the observations to estimate the error of the tree in the testing phase. Thus, we randomly choose k variables as a decision at a certain node in the tree and calculate the best split based on the k variables in the training set. Trees are always grown and never pruned compared to other tree algorithms.

Random forests can handle large numbers of variables in a data set. Also, during the forest building process they generate an internal unbiased estimate of the generalization error. In addition, they can estimate missing data well. A major drawback of random forests is the lack of reproducibility, as the process of building the forest is random. Further, interpreting the final model and subsequent results is difficult, as it contains many independent decisions trees.

3.6 Support vector machines

Support Vector Machines (SVM) are one of the most popular classifiers these days. The idea here is to find the optimal separating hyperplane between two classes by maximizing the margin between the classes closest points. Assume that we have a linear discriminating function and two linearly separable classes with target values +1 and -1. A discriminating hyperplane will satisfy:

$$w' x_i + w_0 \geq 0 \text{ if } t_i = +1;$$
$$w' x_i + w_0 < 0 \text{ if } t_i = -1$$

Now the distance of any point x to a hyperplane is $| w'x_i + w_0 | / \| w \|$ and the distance to the origin is $| w_0 | / \| w \|$. As shown in Figure 4 the points lying on the boundaries are

called support vectors, and the middle of the margin is the optimal separating hyperplane that maximizes the margin of separation (Marques de Sa, 2001).
Though SVMs are very powerful and commonly used in classification, they suffer from several drawbacks. They require high computations to train the data. Also, they are sensitive to noisy data and hence prone to overfitting.

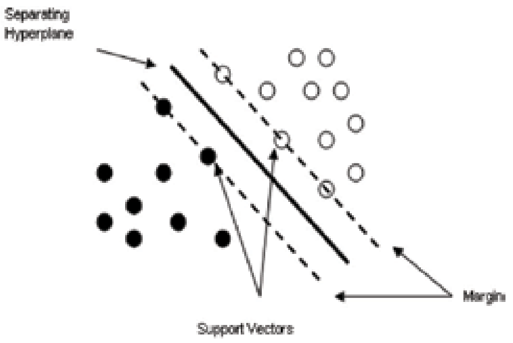


Fig. 4. Support Vector Machines.

4. Quantitative evaluation

4.1 Phishing dataset

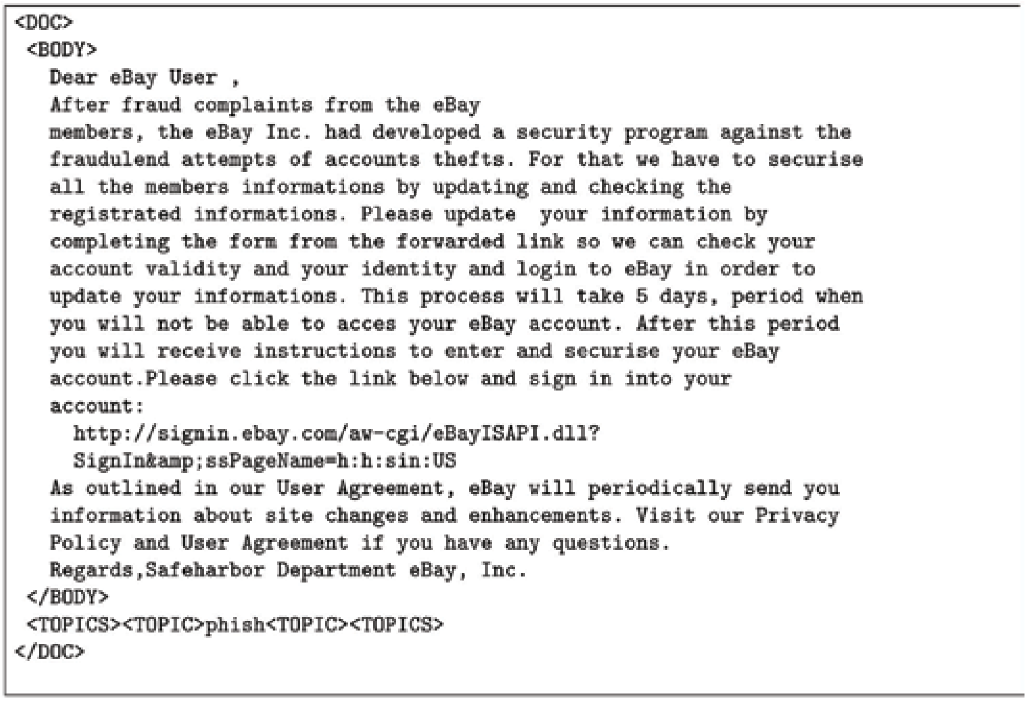
The phishing dataset constitutes of 6561 raw emails. The total number of phishing emails in the dataset is 1409 emails. These emails are donated by (Nazario, 2007) covering many of the new trends in phishing and collected between August 7, 2006 and August 7, 2007. The total number of legitimate email is 5152 emails. These emails are a combination of financial-related and other regular communication emails. The financial-related emails are received from financial institutions such as Bank of America, eBay, PayPal, American Express, Chase, Amazon, AT&T, and many others. As shown in Table 1, the percentage of these emails is 3% of the complete dataset. The other part of the legitimate set is collected from the authors' mailboxes. These emails represent regular communications, emails about conferences and academic events, and emails from several mailing lists.

Corpus	No. of Emails	Percentage (%)
Phishing	1409	21%
Legitimate (financial)	178	3%
Legitimate (other)	4974	76%
Total	6561	100%

Table 1. Corpus description.

4.1.1 Data standardization, cleansing, and transformation

The analysis of emails consists of two steps: First, *textual analysis*, where text mining is performed on all emails. In order to get consistent results from the analysis, one needs to standardize the studied data. Therefore, we convert all emails into XML documents after stripping all HTML tags and email header information. Figure 5 shows an example of a phishing email after the conversions. Text mining is performed using the text-miner software kit (TMSK) provided by (Weiss et al., 2004). Second, *structural analysis*. In this step we analyze the structure of emails. Specifically, we analyze links, images, forms, javascript code and other components in the emails.



```
<DOC>
 <BODY>
   Dear eBay User ,
   After fraud complaints from the eBay
   members, the eBay Inc. had developed a security program against the
   fraudulend attempts of accounts thefts. For that we have to securise
   all the members informations by updating and checking the
   registrated informations. Please update  your information by
   completing the form from the forwarded link so we can check your
   account validity and your identity and login to eBay in order to
   update your informations. This process will take 5 days, period when
   you will not be able to acces your eBay account. After this period
   you will receive instructions to enter and securise your eBay
   account.Please click the link below and sign in into your
   account:
     http://signin.ebay.com/aw-cgi/eBayISAPI.dll?
     SignIn&amp;ssPageName=h:h:sin:US
   As outlined in our User Agreement, eBay will periodically send you
   information about site changes and enhancements. Visit our Privacy
   Policy and User Agreement if you have any questions.
   Regards,Safeharbor Department eBay, Inc.
 </BODY>
 <TOPICS><TOPIC>phish<TOPIC><TOPICS>
</DOC>
```

Fig. 5. Phishing email after conversion to XML.

Afterwards, each email is converted into a vector $\vec{x} = \langle x_1, x_2, ..., x_p \rangle$, where $x_1, ..., x_p$ are the values corre- sponding to a specific feature we are interested in studying (Salton & McGill, 1983). Our dataset consists of 70 continuous and binary features (variables) and one binary response variable, which indicates that email is phishing=1 or legitimate=0. The first 60 features represent the frequency of the most frequent terms that appear in phishing emails. Choosing words (terms) as features is widely applied in the text mining literature and is referred to as "bag-of-words". In Table 2 we list both textual and structural features used in the dataset. As shown in Figure 6, we start by striping all *attachments* from emails in order to facilitate the analysis of emails. The following subsections illustrate the textual and structural analysis in further details.

4.1.2 Textual analysis

As we mentioned earlier we start by stripping all *attachments* from email messages. Then, we extract the *header* information of all emails keeping the email body. Afterwards, we extract the *html tags* and *elements* from the body of the emails, leaving out the body as plain text. Now, we standardize all emails in a form of XML documents. The `<DOC> </DOC>` tags indicate the beginning and ending of a document respectively. The `<BODY> </BODY>` tags indicate the starting and ending of an email body respectively. The `<TOPICS> </TOPICS>` tags indicate the class of the email, whether it is phish or legit (see Figure 5).

Feature	Binary/Continuous	Value	Description
1	binary	0/1	class of email, phishing or legitimate
2-61	continuous	TF/IDF	frequency of terms
62	binary	0/1	link mismatch
63	binary	0/1	URL contains IP
64	binary	0/1	email contains javascript
65	binary	0/1	email contains images that link to external server
66	binary	0/1	email contains form validation
67	binary	0/1	URL contains non-standard ports
68	continuous	maximum total	total number of dots in URL
69	continuous	total	total number of links in email
70	binary	0/1	email contains URL redirection
71	binary	0/1	email contains URL encoding

Table 2. Feature description.

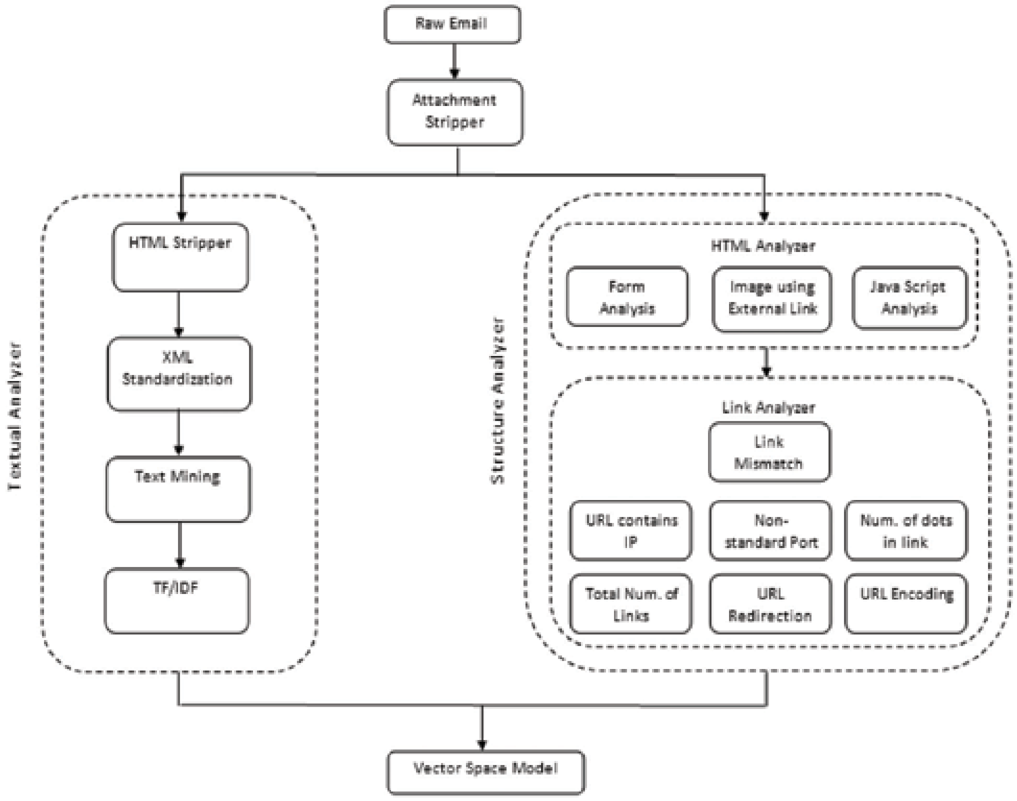
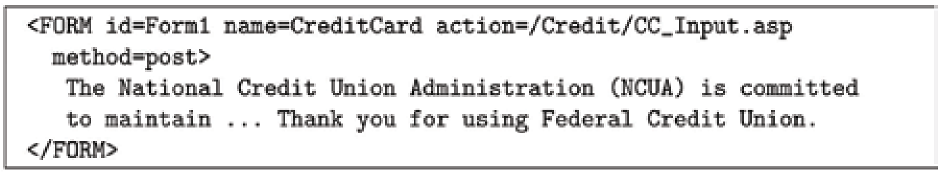


Fig. 6. Building the phishing dataset.

Thus, we filter out *stopwords* from the text of the body. We use a list of 816 commonly used English stopwords. Lastly, we find the most frequent terms using TF/IDF (Term Frequency Inverse Document Frequency) and choose the top 60 most frequent terms that appear in phishing emails. TF/IDF calculates the number of times a word appears in a document multiplied by a (monotone) function of the inverse of the number of documents in which the word appears. In consequence, terms that appear often in a document and do not appear in many documents have a higher weight (Berry, 2004).

4.1.3 Structural analysis

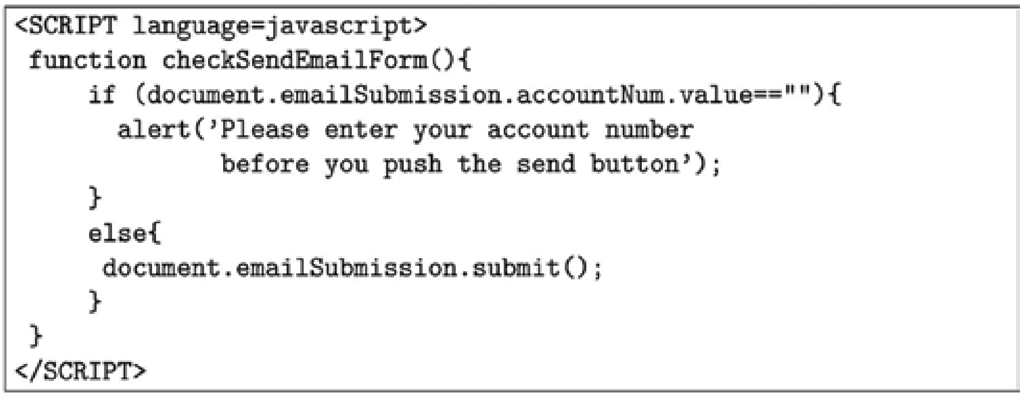
Textual analysis generates the first 60 features in the dataset and the last 10 features are generated using structural analysis. Unlike textual analysis, here we only strip the attachments of emails keeping HTML tags and elements for further analysis. First, we perform HTML analysis, in which we analyze *form* tags, *javascript* tags, and *image* tags. Legitimate emails rarely contain *form* tags that validate the user input. Phishing emails, on the other hand, use this techniques to validate victims' credentials before submitting them to the phishing site. In consequence, if an email contains a *form* tag, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 7 shows an example of a Federal Credit Union phish which contains a *form* tag.



```
<FORM id=Form1 name=CreditCard action=/Credit/CC_Input.asp
  method=post>
   The National Credit Union Administration (NCUA) is committed
   to maintain ... Thank you for using Federal Credit Union.
</FORM>
```

Fig. 7. Form validation in phishing email.

Similarly, legitimate emails rarely contain *javascript*, however, phishers use javascript to validate users input or display certain elements depending on the user input. If the email contains javascript, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 8 shows an example of javascript that is used by a phisher to validate the victims account number.



```
<SCRIPT language=javascript>
 function checkSendEmailForm(){
     if (document.emailSubmission.accountNum.value==""){
       alert('Please enter your account number
              before you push the send button');
     }
     else{
      document.emailSubmission.submit();
     }
 }
</SCRIPT>
```

Fig. 8. Javascript to validate account number.

Spammers have used images that link to external servers in their emails, also dubbed as *Web beacons*, to verify active victims who preview or open spam emails. Phishers also have been following the same technique to verify active victims and also to link to pictures from legitimate sites. We analyze emails that contain *image* tags that link to external servers. If the email contains such an image, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 9 shows an example of a image tag with an external link.

```
<a href="http://201.128.53.64/index.php">
 <img src="http://pics.ebaystatic.com/aw/pics/navbar/eBayLogoTM.gif">
</a>
```

Fig. 9. Image linking to an external server.

The second part in structural analysis involves the *link analysis* process. Here we analyze links in emails. It is well known that phishers use several techniques to spoof links in emails and in webpages as well to trick users into clicking on these links. When analyzing links we look for link mismatch, URL contains IP address, URL uses non-standard ports, the maximum total number of dots in a link, total number of links in an email, URL redirection, and URL encoding. In what follows we describe these steps in more details.
When identifying a link mismatch we compare links that are displayed to the user with their actual destination address in the `<a href>` tag. If there is a mismatch between the displayed link and the actual destination in any link in the email, then the corresponding feature in the dataset is set to 1, otherwise, it is set to 0. Figure 10 shows an example of a PayPal phish, in which the phisher displays a legitimate Paypal URL to the victim; however, the actual link redirects to a Paypal phish.

```
<a href="http://218.214.124.67/paypal/cgi-bin/index.php">
 https://www.paypal.com/cgi-bin/webscr?cmd=_login-run
</a>
```

Fig. 10. URL mismatch in link.

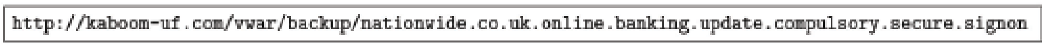
A commonly used technique, but easily detected even by naive users, is the use of IP addresses in URLs (i.e. unresolved domain names). This has been and is still seen in many phishing emails. It is unlikely to see unresolved domain names in legitimate emails; however, phishers use this technique frequently, as it is more convenient and easier to setup a phishing site. If the email contains a URL with an unresolved name, then the corresponding feature in the dataset is set to 1, otherwise, it is set to 0. Phishers often trick victims by displaying a legitimate URL and hiding the unresolved address of the phishing site in the `<a href>` tag as shown in the example in Figure 10.
Since phishing sites are sometimes hosted at compromised sites or botnets, they use non-standard port numbers in URLs to redirect the victim's traffic. For example instead of using port 80 for http or port 443 for https traffic, they use different port numbers. If the email contains a URL that redirects to a non-standard port number, then the corresponding feature in the dataset is set to 1, otherwise it is set to 0. Figure 11 shows an example of a phishing URL using a non-standard port number.

```
To receive email notifications in plain text instead of HTML, update your preferences, <a
href="http://www.online-paypal.h1x.com:8088/security.htm"> Click here.  < /a>
```

Fig. 11. Phishing URL using non-standard port number.

We count the number of links in an email. Usually, phishing emails contain more links compared to legitimate ones. This is a commonly used technique in spam detection, where messages that contain a number of links more than a certain threshold are filtered as spam.

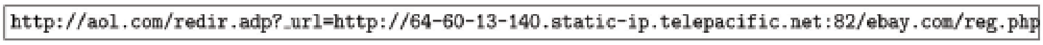
However, since phishing emails are usually duplicate copies of legitimate ones, this feature might not help in distinguishing phishing from financial-related legitimate emails; however, it helps in distinguishing phishing from other regular legitimate messages.
Since phishing URLs usually contain multiple sub-domains so the URL looks legitimate, the number of dots separating sub-domains, domains, and TLDs in the URLs are usually more than those in legitimate URLs. Therefore, in each email we find the link that has the maximum number of dots. The maximum total number of dots in a link in an email thus is used as a feature in the dataset. Figure 12 shows an example of a Nationwide spoof link. Note the dots separating different domains and sub-domains.



```
http://kaboom-uf.com/vwar/backup/nationwide.co.uk.online.banking.update.compulsory.secure.signon
```

Fig. 12. Number of dots in a Nationwide spoof URL.

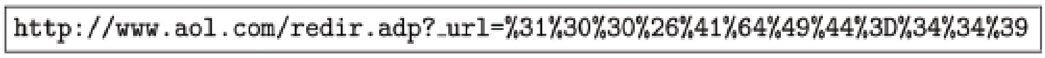
Phishers usually use open redirectors to trick victims when they see legitimate site names in the URL. Specifically, they target open redirectors in well known sites such as `aol.com`, `yahoo.com`, and `google.com`. This technique comes handy when combined with other techniques, especially URL encoding, as naive users will not be able to translate the encoding in the URL. Figure 13 shows an example of an AOL open redirector.



```
http://aol.com/redir.adp?_url=http://64-60-13-140.static-ip.telepacific.net:82/ebay.com/reg.php
```

Fig. 13. Open redirector at AOL.

The last technique that we analyze here is URL encoding. URL encoding is used to transfer characters that have a special meaning in HTML during http requests. The basic idea is to replace the character with the "%" symbol, followed by the two-digit hexadecimal representation of the ISO-Latin code for the character. Phishers have been using this approach to mask spoofed URL and hide the phony addresses of these sites. However, they encode not only special characters in the URL, but also the complete URL. As we mentioned earlier, when this approach is combined with other techniques, it makes the probability of success for the attack higher, as the spoofed URL looks more legitimate to the naive user. Figure 14 shows an example of URL encoding combined with URL redirection.



```
http://www.aol.com/redir.adp?_url=%31%30%30%26%41%64%49%44%3D%34%34%39
```

Fig. 14. URL encoding combined with URL redirection.

Figure 6 depicts a block diagram of the approach used in building the dataset. It shows both textual and structural analysis and the procedures involved therein.

4.2 Evaluation metrics

We use the area under the receiver operating characteristic (ROC) curve (AUC) to measure and compare the performance of classifiers. According to (Huang & Ling, 2005), AUC is a better measure than accuracy when comparing the performance of classifiers. The ROC curve plots false positives (FP) vs. true positives (TP) using various threshold values. It compares the classifiers' performance across the entire range of class distributions and error costs (Huang & Ling, 2005).
Let N_L denote the total number of legitimate emails, and N_P denote the total number of phishing emails. Now, let $n_{L\to L}$ be the number of *legitimate* messages classified as *legitimate*,

$n_{L\to P}$ be the number of *legitimate* messages misclassified as *phishing*, $n_{P\to L}$ be the number of *phishing* messages misclassified as *legitimate*, and $n_{P\to P}$ be the number of *phishing* messages classified as *phishing*. False positives are legitimate emails that are classified as phishing, hence the false positive rate (FP) is denoted as:

$$FP = \frac{n_{L\to P}}{N_L}. \tag{8}$$

True positives are phishing emails that are classified as phishing, hence the true positive rate (TP) is denoted as:

$$TP = \frac{n_{P\to P}}{N_P}. \tag{9}$$

False negatives are phishing emails that are classified as legitimate, hence the false negative rate (FN) is denoted as:

$$FN = \frac{n_{P\to L}}{N_P}. \tag{10}$$

True negatives are legitimate emails that are classified as legitimate, hence the true negative rate (TN) is denoted as:

$$TN = \frac{n_{L\to L}}{N_L}. \tag{11}$$

Further we evaluate the predictive accuracy of classifiers, by applying the *weighted error* (W_{Err}) measure proposed in (Sakkis et al., 2003) and (Zhang et al., 2004). We test the classifiers using λ = 1 that is when legitimate and phishing emails are weighed equally. Hence the weighted accuracy (W_{Acc}), which is 1 - $W_{Err}(\lambda)$, can be calculated as follows

$$W_{Acc}(\lambda) = \frac{\lambda \cdot n_{L\to L} + n_{P\to P}}{\lambda \cdot N_L + N_P}. \tag{12}$$

In addition, we use several other measures to evaluate the performance of classifiers. We use the phishing *recall*(r), phishing *precision*(p), and phishing f_1 measures. According to (Sakkis et al., 2003), spam recall measures the percentage of spam messages that the filter manages to block (filter's effectiveness). Spam precision measures the degree to which the blocked messages are indeed spam (filter's safety). F-measure is the weighted harmonic mean of precision and recall. Here we use f_1 when recall and precision are evenly weighted. For the above measures, the following equations hold

$$p = \frac{n_{P\to P}}{n_{P\to P} + n_{L\to P}} \tag{13}$$

$$p = \frac{n_{P\to P}}{n_{P\to P} + n_{L\to P}} \tag{14}$$

$$f_1 = \frac{2pr}{p+r} \tag{15}$$

We use the AUC as the primary measure, as it allows us to gauge the trade off between the FP and TP rates at different cut-off points. Although the error rate W_{Err} (or accuracy) has been widely used in comparing classifiers' performance, it has been criticized as it highly depends on the probability of the threshold chosen to approximate the positive classes. Here we note that we assign new classes to the positive class if the probability of the class is greater than or equal to 0.5 (threshold=0.5). In addition, in (Huang & Ling, 2005) the authors prove theoretically and empirically that AUC is more accurate than accuracy to evaluate classi_ers' performance. Moreover, although classifiers might have different error rates, these rates may not be statistically significantly different. Therefore, we use the Wilcoxon signed-ranks test (Wilcoxon, 1945) to compare the error rates of classifiers and find whether the di_erences among these accuracies is significant or not (Demšar, 2006).

4.3 Experimental studies

We optimize the classifiers' performance by testing them using different input parameters. In order to find the maximum AUC, we test the classifiers using the complete dataset applying different input parameters. Also, we apply *10-fold-cross-validation* and average the estimates of all 10 folds (sub-samples) to evaluate the average error rate for each of the classifiers, using the 70 features and 6561 emails. We do not perform any preliminary variable selection since most classifiers discussed here can perform automatic variable selection. To be fair, we use L1-SVM and penalized LR, where variable selection is performed automatically.

We test NNet using different numbers of units in the hidden layer (i.e. different sizes (s)) ranging from 5 to 35. Further, we apply different weight decays (w) on the interconnections, ranging from 0.1 to 2.5. We find that a NNet with s = 35 and w = 0.7 achieves the maximum AUC of 98.80%.

RF is optimized by choosing the number of trees used. Specifically, the number of trees we consider in this experiment is between 30 and 500. When using 50 trees on our dataset, RF achieves the maximum AUC of 95.48%.

We use the L1-SVM C-Classification machine with radial basis function (RBF) kernels. L1-SVM can automatically select input variables by suppressing parameters of irrelevant variables to zero. To achieve the maximum AUC over different parameter values, we consider cost of constraints violation values (i.e. the "c" constant of the regularization term in the Lagrange formulation) between 1 and 16, and values of the γ parameter in the kernels between 1×10^{-8} and 2. We find that γ= 0.1 and c = 12 achieve the maximum AUC of 97.18%.

In LR we use penalized LR and apply different values of the lambda regularization parameter under the L2 norm, ranging from 1×10^{-8} to 0.01. In our dataset $\lambda = 1 \times 10^{-4}$ achieves the maximum AUC of 54.45%.

We use two BART models; the first is the original model and as usual, we refer to this as "BART". The second model is the one we modify so as to be applicable to classification, referred to as "CBART". We test both models using different numbers of trees ranging from 30 to 300. Also, we apply different power parameters for the tree prior, to specify the depth of the tree, ranging from 0.1 to 2.5. We find that BART with 300 trees and *power* = 2.5 achieves the maximum AUC of 97.31%. However, CBART achieves the maximum AUC of 99.19% when using 100 trees and *power* = 1.

4.4 Experimental results

In this section we present the experimental results be measuring the AUC using the complete dataset. In addition, we compare the *precision*, *recall*, *f*1, and W_{Err} measures using the optimum parameters achieved from the previous section. Figure 15 illustrates the ROCs for all classifiers.

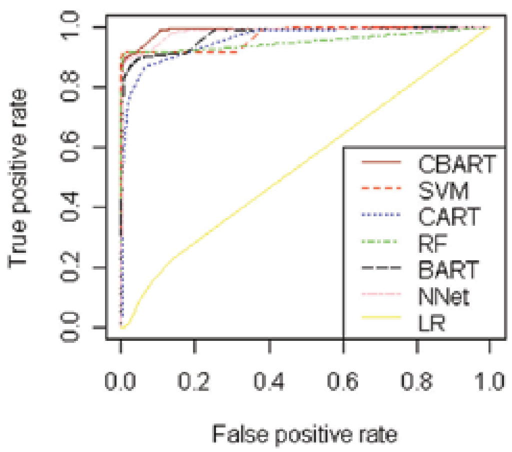


Fig. 15. ROC for all classifiers using the complete dataset.

Table 3 illustrates the AUC, FP, FN, *presicion*, *recall*, *f*1, and W_{Err} for all classi_ers. Note that the *FPrate* = 1 - *precision* and the *FNrate* = 1 - *recall*.

	AUC	W_{Err}	P	R	F1	FP	FN
CBART	99.19%	4.41%	97.02%	88.86%	92.8%	2.98%	11.14%
NNet	98.80%	4.31%	93.84%	85.68%	89.53%	6.16%	14.32%
SVM	97.18%	2.39%	94.57%	86.23%	90.17%	5.43%	13.77%
BART	97.31%	4.74%	93.82%	83.52%	88.32%	6.18%	16.48%
CART	96.06%	7.00%	88.45%	77.90%	82.67%	11.55%	22.10%
RF	95.48%	3.68%	95.75%	86.80%	91.02%	4.25%	13.20%
LR	54.45%	5.34%	92.71%	81.62%	86.77%	7.29%	18.38%

Table 3. Classifiers AUC, W_{Err}, precision, recall, f1, false positive, false negative.

In Table 4, we compare *p-value* of the error rate for each subsample among the 10 subsamples in cross validation by applying the Wilcoxon signed-rank test. Since CBART has a comparable error rate to that of RF, SVM, and NNet, we merely compare these three classifiers.

Classifier	p-value
RF	0.03711
SVM	0.1602
NNet	0.625

Table 4. Comparing the p-value using the Wilcoxon-signed ranked test

4.5 Discussion

Here we investigate the application of a modified version of BART for phishing detection. The results demonstrate that CBART outperforms other classifiers on the phishing dataset, achieving the maximum AUC of 99.19%. The results show that CBART has the highest AUC, followed by NNet with 98.80%, BART with 97.31%, SVM with 97.18%, CART with 96.06%, RF with 95.48%, and LR with 54.45% respectively. Apparently the AUC for CBART has improved by 1.88% compared to the original BART model.

The results show that CBART's error rate is comparable to other classifiers (merely RF, SVM, and NNet). When the error rates for the 10 subsamples are compared using the Wilcoxon-singed ranked test, the *p-value* of the tests for SVM and NNet are greater than 0.05, which is an indication that the difference in the results is not statistically significantly different (see Table 4). However, for RF, the *p-value* is less than 0.05, which is an indication that the error rates *may be* statistically significantly different. Table 3 illustrates that the AUC for CBART is greater than RF by approximately 4%, therefore, we conclude that CBART's performance is better than RF.

SVM achieves the minimum error rate (maximum accuracy) of 2.39%, followed by RF with 3.68%, NNet with 4.31%, CBART with 4.41%, BART with 4.74%, LR with 5.34%, and CART with 7% respectively. Note that the accuracy of CBART has improved, insignificantly though, by 0.33%.

CBART achieves the minimum FP rate of 2.98%, followed by RF with 4.24%, SVM with 5.43%, NNet with 6.16%, BART with 6.18%, LR with 7.29%, and CART with 11.55%. The minimum FN rate is achieved by CBART with 11.14%, followed by RF with 13.20%, SVM with 13.77%, NNet 14.32%, BART 16.48%, LR 18.38%, and CART 22.10% respectively.

It is well known that LR performs very well when the relationship underlying data is linear. We believe that the comparatively low predictive accuracy of LR is an indication of a non-linear relationship among the features and the response in the dataset.

With its superior classification performance, relatively high predictive accuracy, relatively low FP rate, and distinguished features, CBART proves to be a competitive and a practical method for phishing detection.

5. Conclusions

A modified version of Bayesian Additive Regression Trees (CBART) proved to be suitable as a phishing detection technique. The performance of CBART was compared against well-known classification methods on a phishing dataset with both textual and structural features.

The results showed that CBART is a promising technique for phishing detection and it has several features that make it competitive. Further, the results demonstrated that the performance of the modified BART model outperforms other well-known classifiers and comparatively achieves low error rate and false positives. CBART outperformed all the other classifiers and achieved the maximum AUC of 99.19% on the phishing dataset we built, decreasing by 1.88% compared to AUC of BART prior to the modification. SVM achieved the minimum error rate of 2.39% leaving behind, RF with 3.68% and NNet with 4.31% followed by CBART with 4.41%. In addition, CBART achieved the minimum FP rate of 2.98% followed by RF with 4.25%, SVM with 5.43%, NNet with 6.16%, and BART with 6.18%.

Automatic variable selection in BART motivates future work to explore BART as a variable selection technique. This includes comparing its performance to other well known variable selection methods.

6. References

Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. *eCrime '07: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit* (pp. 60-69). New York, NY, USA: ACM.

Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2008). Bayesian additive regression trees-based spam detection for enhanced email privacy. *ARES '08: Proceedings of the 3rd International Conference on Availability, Reliability and Security* (pp. 1044-1051).

Albert, J. H., & Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, *88*, 669-679.

Berry, M. W. (Ed.). (2004). *Survey of text mining: Clustering, classification, and retrieval*. Springer.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5-32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Chapman & Hall/CRC.

Chandrasekaran, M., Narayanan, K., & Upadhyaya, S. (2006). Phishing email detection based on structural properties. *NYS Cyber Security Conference*.

Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, *93*, 935-947.

Chipman, H. A., George, E. I., & McCulloch, R. E. (2006). BART: Bayesian Additive Regression Trees. http://faculty.chicagogsb.edu/robert.mcculloch/research/code/BART-7-05.pdf.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1-30.

Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to detect phishing emails. *WWW '07: Proceedings of the 16th international conference on World Wide Web* (pp. 649-656). New York, NY, USA: ACM Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning data mining, inference, and prediction*. Springer Series in Statistics. Springer.

Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *17*.

James, L. (2005). *Phishing exposed*. Syngress.

Kandola, J. S. (2001). *Interpretable modelling with sparse kernels*. Doctoral dissertation, University of Southampton.

Marques de Sa, J. P. (2001). *Pattern recognition: Concepts, methods and applications*. Springer.

Massey, B., Thomure, M., Budrevich, R., & Long, S. (2003). Learning spam: Simple techniques for freely-available software. *USENIX Annual Technical Conference, FREENIX Track* (pp. 63-76).

Nazario, J. (2007). Phishing corpus. http://monkey.org/~jose/phishing/phishing3.mbox.

Neal, R. M. (1995). *Bayesian learning for neural networks*. Springer-Verlag Publishers.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C., & Stamatopoulos, P. (2003). A memory-based approach to anti-spam filtering for mailing lists. *Information Retrieval*, *6*, 49-73.

Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill.

Weiss, S., Indurkhya, N., Zhang, T., & Damerau, F. (2004). *Text mining: Predictive methods for analyzing unstructured information*. Springer.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, *1*, 80-83.

Zhang, L., Zhu, J., & Yao, T. (2004). An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, *3*, 243-269.

Learning Optimal Web Service Selections in Dynamic Environments when Many Quality-of-Service Criteria Matter

Stéphane Dehousse[1], Stéphane Faulkner[2], Caroline Herssens[3], Ivan J. Jureta[4] and Marcos Saerens[5]
[1,2,4]PReCISE Research Center (University of Namur), Namur,
[3]PReCISE Research Center(University of Louvain), Louvain-la-Neuve,
[5]ISYS Research Unit (University of Louvain), Louvain-la-Neuve,
Belgium

1. Introduction

The emergence of the World Wide Web has lead to growing needs for interacting components capable of achieving – together – complex requests on the Web. Service oriented Systems (SoS) are a response to this issue, given available standards for describing individual services and interaction between them, and the attention to interoperability combined with an uptake in industry. A *service* is a self-describing and self-contained modular application designed to execute a well-delimited task, and that can be described, published, located, and invoked over a network (McIlraith & Martin, 2003; Papazoglou & Georgakopoulos, 2003). A *web service* is a service made available on the Internet via tailored technologies such as WSDL, SOAP or UDDI (Walsh, 2002). To fulfill elaborate requests that involve many execution steps, web services participate in *web services compositions*. To optimize such compositions, each step of the execution is achieved by the most competitive available web service. The most competitive web service is the one who performs the given task while fulfilling its functional requirements and providing the best observed values of quality of service (QoS).
QoS are the nonfunctional properties of a web service and refer to concerns such as availability, reliability, cost or security (Menascé, 2002). The selection of all web services that can participate in a composition (i.e., web services that will perform at least one step of the execution) is under the responsibility of the *service composer*. To achieve QoS-aware service selection, we rely on a Multi-Criteria Randomized Reinforcement Learning approach (MCRRL). MCRRL authorizes automated continuous optimization of service monitoring and leads the system to respond to the variation of the availability of web services without human involvement.
This paper focuses on the composition of services under the constraint of openness, resource distribution, and adaptability to changing web service availability w.r.t. multiple criteria. To enable such system characteristics, a fit between the system architecture and services composition behavior is needed, that is: (1) To support openness, few assumptions can be

made about the behavior of the web services that may participate in compositions. It thus seems reasonable to expect services composition responsibility not to be placed on any web service: the architecture ought to integrate a special set of web services, the service composers, that coordinate services composition. (2) To allow the distribution of web services, no explicit constraints should be placed on the origin of entering services. (3) To enable adaptability, composer behavior should be specified along with the architecture. (4) Since there is no guarantee that web services will execute tasks at performance levels advertised by the providers, composition should be grounded in empirically observed service performance and such observations executed by the composers. (5) The variety of stakeholder expectations requires services composition to be driven by multiple criteria. (6) To ensure continuous adaptability at runtime, composition within the architecture should involve continual observation of service performance, the use of available information to account for service behavior and exploration of new options to avoid excessive reliance on historical information.

Contributions. We provide a complete composition process involving several steps: (1) The service user requests a service that involves several tasks that can be fulfilled by different web services. These tasks and all possible execution paths are described on on a statechart. (2) The service composer observes available web services and rejects those that can not achieve one of the existing task of the statechart. It then builds the resulting execution plan as a Directed Acyclic Hypergraph, on which it represents all services available for each task. (3) The service requester expresses its quality expectations with the help of our QoS model. (4) The composer rejects services that do no meet quality requirements and scores each candidate web service with our proposed QoS aggregation model to get a multi criteria measure of their performance. (5) This value is the one that the service composer maximizes in our RRL algorithm. The result of the computation gives us web services to select to get the most competitive composite web service.

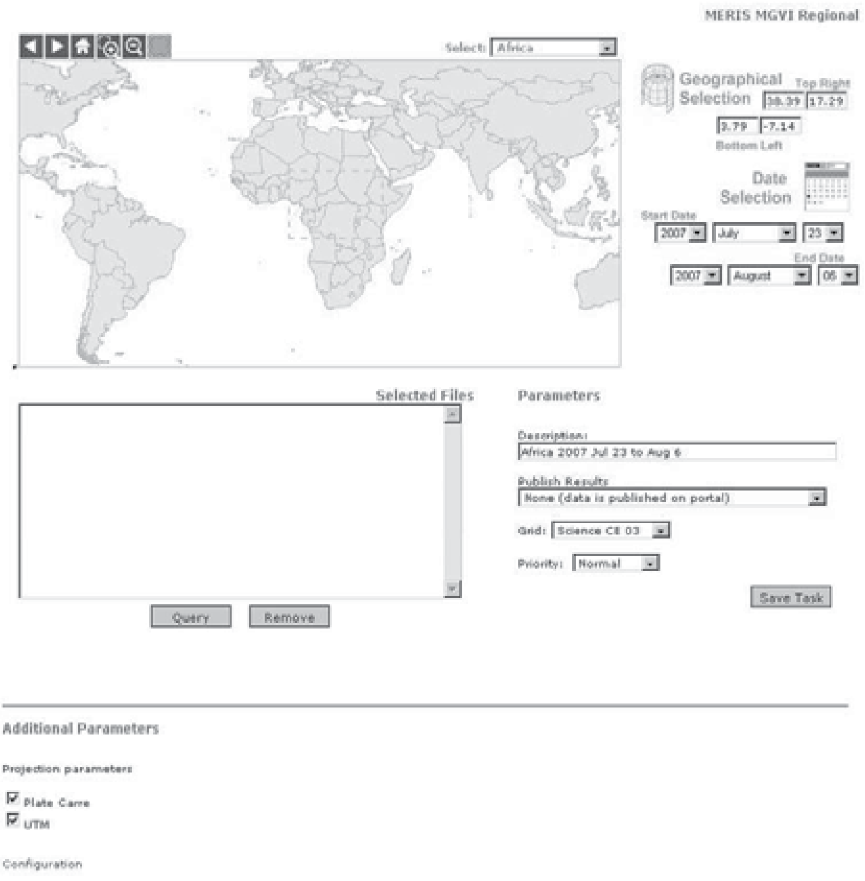
Organization. We present our conceptual foundations for the remaining of the paper in Section 2. That section covers the case study used throughout this paper and our composition model with its statechart representation and its Directed Acyclic Hypergraph derivation. It also proposes our QoS model applied by the service user to specify its priorities and preferences about QoS. Section 3 presents how multiple quality criteria are aggregated into a single measure of performance. The method is illustrated with the previously introduced case study. Section 4 introduces our Reinforcement Learning solution to the composition problem by liken it to the task allocation problem. Section 5 presents experiments that we made on our Multi-Criteria Randomized Reinforcement Learning proposal. Section 6 outlines the related work. Finally Section 7 concludes this paper and exposes our future work.

2. Baseline

This section presents the different conceptual elements used through the paper. Our case study is introduced in Subsection 2.1. Our services composition model is proposed in Subsection 2.2 and involves two steps. The first is to define the possible composition process with a statechart as described in Subsection 2.2.1. We illustrate the statechart representation with the composition of web services introduced in the case study. The second is to represent candidate services for each elementary task of the whole composition. This representation is derived from statecharts with Directed Acyclic Hypergraph. The procedure for doing so is explained and illustrated with our case study in Subsection 2.2.2. We present in Subsection 2.3 the QoS model dedicated to the service user to make its particular requirements about its QoS priorities and preferences.

2.1 Case study

To illustrate our selection of services entering in a composition, we propose a case study subsequently used throughout the paper. The European Space Agency's (ESA) program on Earth observation allows researchers to access and use infrastructure operated and data collected by the agency.[1] Our case study focuses on the information provided by the MERIS instrument on the Envisat ESA satellite. MERIS is a programmable, medium-spectral resolution imaging spectrometer operating in the solar reflective spectral range. MERIS is used in observing ocean color and biology, vegetation and atmosphere and in particular clouds and precipitation. In relation to MERIS, web services are made available by the ESA for access to the data the instrument sends and access and use of the associated computing resources.



Select: Africa

Top Right: 38.39 17.29
Bottom Left: 3.79 -7.14

Start Date: 2007 July 23
End Date: 2007 August 06

Parameters

Description: Africa 2007 Jul 23 to Aug 6
Publish Results: None (data is published on portal)
Grid: Science CE 03
Priority: Normal

Additional Parameters

Projection parameters
✓ Plate Carre
✓ UTM

Configuration

Fig. 1. Graphical user interface of the ENVISAT/MERIS MGVI web service

[1]http://gpod.eo.esa.int

Among available functionalities delivered by these web services, we focus our attention to services enabling to extract the vegetation index, or more precisely, the Fraction of Absorbed Photosynthetically Active Radiation (FAPAR) from MERIS data. The graphical interface used to determine the requested information for the vegetation index on a given region is illustrated in Figure 1. The graphical output produced for the world-wide map is given in Figure 2.

Two main services allow the extraction of such data, one processing the information for the world-wide map and the other computing information for a given area of the world. The graphical user interface of the service providing regional data differs from the world wide one with a bounding box enabling to select an area on the map. World-wide data is much often requested than data for a given region of the world, so World-wide data can be more rapidly retrieved than the regional. Moreover, some services authorize the extraction of regional data from world-wide data.

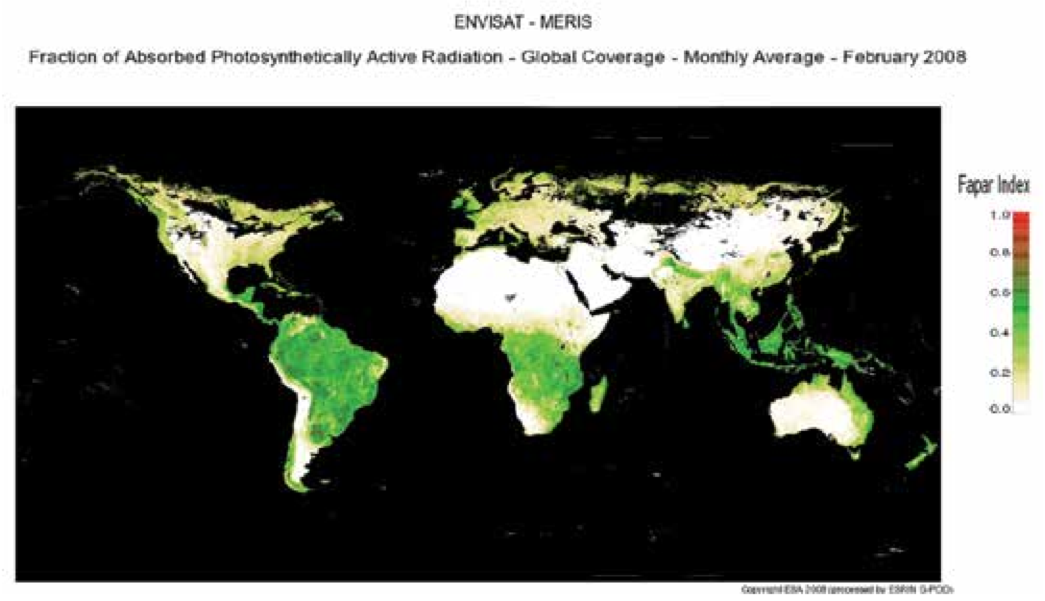

ENVISAT - MERIS
Fraction of Absorbed Photosynthetically Active Radiation - Global Coverage - Monthly Average - February 2008
Fapar Index
1.0
0.8
0.6
0.4
0.2
0.0

Fig. 2. Output provided by the world-wide vegetation service

2.2 Web services composition model

Service requests pointed out that various criteria can be used in specifying a service request; namely, QoS concepts cover deadline, reputation, monetary cost, and explicit requester preferences. Reputation and trust receive considerable attention in the literature (e.g., (Maximilien & Singh, 2005; Zacharia & Maes, 2000)). In AOSS, the ideas underlying Maximilien and Singh's approach (Maximilien & Singh, 2005) can be followed, with two caveats: they use "trust" to select services from a pool of competing services and exploit user-generated opinions to calculate reputation, whereas herein WS are selected automatically and reputation can be generated by comparing WS behavior observed by the composer and the advertised behavior of the WS. The following is one way to define reputation in AOSS.[2]

[2]Reputation is used here instead of trust since no user opinions are accounted for.

Definition 1. Reputation R_k^{a,w_i} of a WS w_i over the QoS aggregated score k is:

$$R_k^{a,w_i} = \frac{1}{n-1}\sum_{i=1}^{n}\left[\left(v_k^{Adv} - \hat{v}_k^i\right)^2 \delta^{-time(\hat{v}_k^i)}\right]$$

where $time()$ returns the time of observation (a natural, 1 for the most recent observed value, $time(\hat{v}_k^i) > 1$ for all other) and δ is the dampening factor for the given quality (can be used with $time()$ to give less weight to older observations). We assume that the advertised quality for w_i is $\langle(p_1,d_1,v_1^{Adv},u_1),\ldots,(p_r,d_r,v_r^{Adv},u_r)\rangle$, and that n observations $\hat{v}_k^i, 1 \le i \le n$ have been made over a quality parameter k.
It is apparent that many other criteria can be accounted for when selecting among alternative WS compositions. Decision making in presence of multiple criteria does not require full specification of all possible criteria for each WS---instead, it is up to the requester to choose what criteria to specify. The algorithm thus optimizes a single normalized variable (i.e., taking values in the interval $[0,1]$). An aggregation function for the criteria relevant to the service requester is applied, so that the result of the function is what the algorithm will optimize. The process providing the aggregation function is presented in Section 3.

2.2.1 Statechart representation

A services composition is a succession of elementary tasks, whose exution fulfills a complex request. We assume the request describes a process to execute. Individual web services are combined together according to their functional specifications. Compositions support alternative possibilities and concurrency of elementary tasks. Similarly to Zeng and colleagues, our service process is defined as a statechart (Zeng et al., 2003). Statecharts offer well defined syntax and semantics so that rigorous analysis can be performed with formal tools to check specification concordance between services. Another advantage is that they incorporate flow constructs established in process modeling languages (i.e, sequence, concurrency, conditional branching, structured loops, and inter-thread synchronization). Consequently, standardized process modeling languages, such as, e.g., BPMN (OMG, 2006a), can be used to specify the process model when selecting services that will enter in the composition. Statecharts offer the possibility to model alternatives and a composite task can be achieved by different paths in the statechart. Such paths are named *execution paths* ant their definition in relation to statecharts is given in Definition 2.2.1. The statechart is a useful representation of a process that a WS composition needs to execute, most selection algorithms cannot process a statechart in its usual form. Instead, a statechart is mapped onto a Directed Acyclic Hypergraph (DAH), using Definition 2.2.1 and the technique for constructing DAH, described below.
(Adapted from (Zeng et al., 2003)) An execution path of a statechart is a sequence of states $[t_1,t_2,\ldots,t_n]$, such that t_1 is the initial state, t_n the final state, and for every state $t_i (1 < i < n)$, the following holds:

- t_i is a direct successor of one of the states in $[t_1,\ldots,t_{i-1}]$.
- t_i is not a direct successor of any of the states in $[t_{i+1},\ldots,t_n]$.

- There is no state t_j in $[t_1,\ldots,t_{i-1}]$ such that t_j and t_i belong to two alternative branches of the statechart.

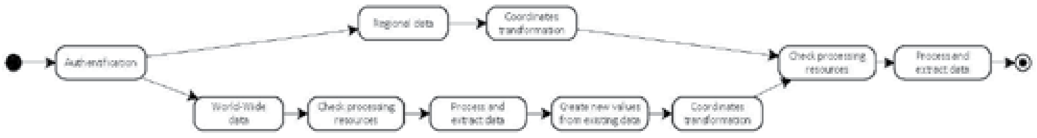

Authentification
Regional data
Coordinates transformation
Check processing resources
Process and extract data
World-Wide data
Check processing resources
Process and extract data
Create new values from existing data
Coordinates transformation

Fig. 3. Statechart representation of the composite service

We concentrate our efforts here on the description of elementary tasks of a composite service processing the FAPAR for a given region of the world. Two main paths of tasks allow to achieve this composite service. Besides elementary tasks stepping in both paths, the first path uses services allowing to process data for a given region of the world while the second path process the world-wide data and restrains the information to the given area. The composite service and its elementary tasks are illustrated with the corresponding statechart in Figure 2.2.1.

2.2.2 Directed acyclic hypergraph instantiation

It is apparent that an acyclic statechart has a finite number of execution paths. If the statechart is not acyclic, it must be "unfolded" (Zeng et al., 2003): logs of past executions need to be examined in order to determine the average number of times that each cycle is taken. The states between the start and end of a cycle are then duplicated as many times as the cycle is taken on average. Assuming for simplicity here that the statechart is acyclic, an execution path can be represented as a Directed Acyclic Hypergraph.

Given a set of distinct execution paths $\{[t_{1,k},\ldots,t_{n,k}]\}$ (k is the index for execution paths), the Directed Acyclic Hypergraph (DAH) is obtained as follows:

- DAH has an edge for every pair $(task, WS)$ which indicates the allocation of WS to the given task. DAH thus has as many edges as there are possible allocations of WS to tasks.
- DAH has a node for every state of the task allocation problem. Such a state exists between any two sequentially ordered tasks of the task allocation problem (i.e., a node connecting two sets of edges in the DAH, whereby the two tasks associated to the two sets of edges are to be executed in a sequence).

Note that: (i) the DAH shows all alternative allocations and all alternative execution paths for a given statechart; (ii) conditional branchings in a statechart are represented with multiple execution paths.

Available web services for fulfilling individual tasks of our composite service proposed in Figure 2.2.1 need to be represented in a DAH to apply our selection approach. Each state of the statechart will become a node in the DAH with an additional starting node depicting the initial state. The resulting DAH is available in Figure 3 with each edge standing for a service able to fulfill the task specified in the outgoing node of the edge. Several services provided by the ESA are able to fulfill each individual tasks of the composite service providing the FAPAR index. The DAH representation gather web services which can be used at different steps of the execution.

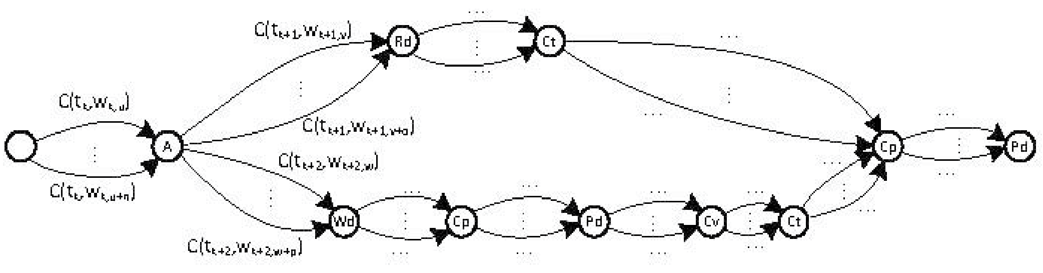


Fig. 4. DAH representation of the composite service

2.3 Specification of user priorities and preferences

We suggest a QoS model that enables the user to express accurately its needs about quality properties of its required service. To account for various aspects of user expectations, this model must include advanced concepts such as priorities over quality characteristics or preferences on offered values. To enable specifying these concepts, the model contains modeling constructs dedicated to various facets of user expectations.

Among multiple available QoS models (D'Ambrogio, 2006; Keller & Ludwig, 2003; Zhou et al., 2004), we base our model on the UML QoS Profile. The original UML QoS Framework metamodel, introduced by the Object Management Group (OMG, 2006b), includes modeling constructs for the description of QoS considerations. It has some advantages over other models: it is based on the Unified Modeling Language (UML); it is a standard provided by the Object Management Group (OMG); it is a metamodel that can be instantiated in respect to users needs; and it covers numerous modeling constructs and allows to add some extensions. This model with our extensions are shown in Figure 2.3.

In that metamodel, a *QoS Characteristic* is a description for some quality consideration, such as e.g., latency, availability, reliability or capability. Extensions and specializations of such elements are available with the sub-parent self-relation. A characteristic has the ability to be derived into various other characteristics as suggested by the templates-derivations self-relation. A *QoS Dimension* specifies a measure that quantifies a QoS Characteristic. The *unit* attribute specifies the unit for the value dimension. *QoS Values* are instantiations of QoS Dimensions that define specific values for dimensions depending on the value definitions given in QoS DimensionSlots. A *QoS DimensionSlot* represents the value of QoSValue. It can be either a primitive QoS Dimension or a referenced value of another QoSValue. While constraints usually combine functional and non-functional considerations about the system, *QoS Context* is used to describe the context in which quality expression are involved. A context includes several QoS Characteristics and model elements. The aim of *QoS Constraints* is to restrict values of QoS Characteristics. Constraints describe limitations on characteristics of modeling elements identified by application requirements and architectural decisions.

In comparison with the original OMG metamodel, we make some additional assumptions:

- In the OMG standard, *QoS Characteristics* are quantified by means of one or several *QoS Dimensions*. We assume that the value of a QoS Dimension can similarly be calculated with quantitative measures of other QoS Dimensions. This assumption is expressed in the metamodel in Figure 2.3 through the *Compose-Composed by* relationship of the *QoS Dimension* metaclass.

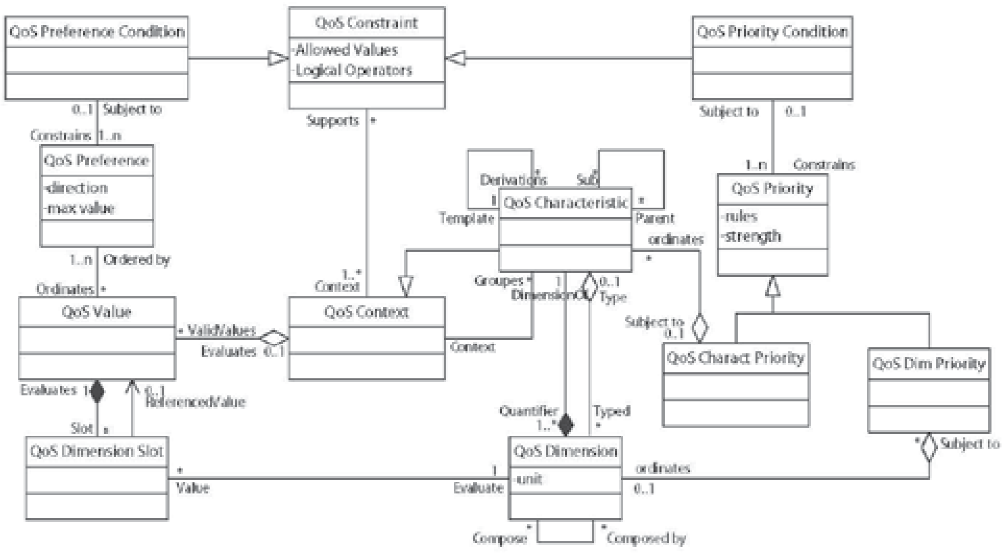


Fig. 5. UML metaclasses to user modeling

- We allow the user to express its priorities over QoS Characteristics and over QoS Dimensions by means of, respectively, *QoS Charact Priority* and *QoS Dim Priority* metaclasses whose are specializations of the *QoS Priority* metaclass. Its attribute *rules* concerns QoS Characteristics or QoS Dimensions involved in the priority and the direction of the priority while the attribute *strength* indicates the relative importance of the priority. *QoS Priority Condition* indicates conditions that need to hold in order for the priority to become applicable.
- To enable the user to express its preferences over values of QoS Characteristics and QoS Dimensions, we add a specific metaclass: *QoS Preference*. Preferences over values are defined with some attributes: *direction* states if the value has to be minimized or maximized; *max value* indicates the maximal value expected by the user and defines its preference.

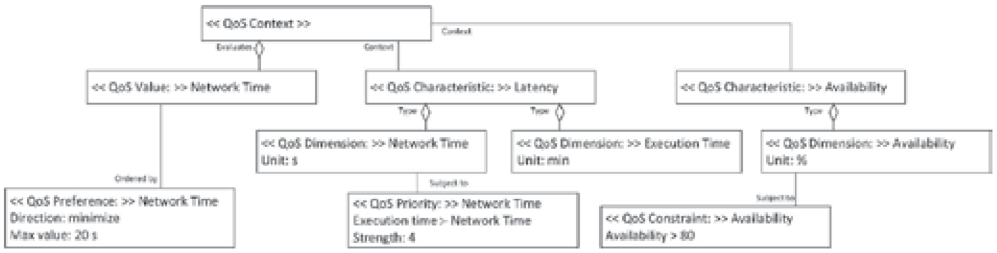

<< QoS Context >>
Context
Evaluation
Context
<< QoS Value: >> Network Time
<< QoS Characteristic: >> Latency
<< QoS Characteristic: >> Availability
Type
Type
Type
<< QoS Dimension: >> Network Time
Unit: s
<< QoS Dimension: >> Execution Time
Unit: min
<< QoS Dimension: >> Availability
Unit: %
Ordered by
Subject to
Subject to
<< QoS Preference: >> Network Time
Direction: minimize
Max value: 20 s
<< QoS Priority: >> Network Time
Execution time > Network Time
Strength: 4
<< QoS Constraint: >> Availability
Availability > 80

Fig. 6. User specifications

To illustrate our scoring model, we suppose a service requester who wishes to use our composite service processing FAPAR for a given area of the world while optimizing the following QoS Characteristics: *availability, cost, latency, reliability, reputation* and *security*. Some of these quality considerations are not directly quantifiable, and are measured with help of multiple QoS Dimensions (e.g.: latency is quantified by network time and execution time), others are measured with a single QoS Dimension (e.g.: the availability is a measure provided in %). All these information are specified by the service requester with the help of our proposed QoS model. Parts of the complete specification of the user are illustrated in Figure 2.3.

3. QoS scoring of services

In order to select web services that will fulfill the different elementary tasks of the composition, the service composer must decide between them. Because web services represented in the DAH meet functional requirements, their discrimination will be made on their quality properties. To account for multiple quality properties in the reinforcement learning composition process, QoS need to be adequately aggregated. We explain in this section how the composer give an aggregated QoS score to each available service of the composition with help of Multi-Criteria Decision Making (MCDM) techniques. The QoS score is calculated by considering quality requirements expressed by the service user. To express such requirements, that must be interpretable by the service composer, the user needs an appropriate quality model. We present our QoS model and illustrate its utilization with the earth observation composite service of the ESA introduced in Subsection 2.2.
The service composer uses information specified with the QoS model in combination with Multi-Criteria Decision Making (MCDM) techniques to establish an aggregated measure of quality properties on all available services. This measure must be calculated for each service candidate of the composition. However multiple execution paths are available in the DAH representation of the composite service and, these paths can be subject to major variations in quality performance. In our ESA case study, we observed that services used to generate world-wide data are slower than services providing regional data but are also more reliable. Anyway, scores of services need to be comparable to service candidates on all paths of the composition. To achieve this global measurement, the scoring will be established by pairwise comparisons on all services suitable for any tasks of the composition.
The scoring process involves the following steps: (1) apply hard constraints on services, to restrict the set of services upon whose MCDM calculation will be made. (2) establish the hierarchy of quality properties with information related to characteristics and dimensions decomposition, each property being considered as a criterion of the MCDM model. Moreover, two distinct hierarchies are build, the first dedicated to benefits, i.e.: criteria to maximize, the second dedicated to costs, i.e.: criteria to minimize. (3) fix the priorities of quality properties by applying the Analytic Hierarchy Process (AHP) on both hierarchies. (4) give a score to each service alternative for both benefits and costs hierarchies. This step is done with the Simple Additive Weighting (SAW) process, which gives us the opportunity to score alternatives with few information given on criteria. (5) for each alternative, the ratio benefits/costs is computed by service composer and a score is linked to each available service.

3.1 Fixing hard constraints

Hard constraints on quality properties (i.e.: QoS Characteristics or QoS Dimensions) are defined by the user to restrict the set of accepted services. These are specified with the *QoS Constraint* metaclass and fix thresholds to values of a QoS Dimension. While the service composer assigns best available services to the service requester, services that do not fulfill thresholds values for the different QoS Dimensions taken into account are considered

irrelevant. Constraints allow us to decrease the number of alternative services to consider when applying MCDM - all services that do not satisfy the constraints are not considered for comparison.
The complete specification made by the service requester with the QoS model is transmitted to the service composer that will process all steps of the selection. The composer starts by rejecting services that do not fulfill hard constraints. For example, in specification given in Figure 2.3, the composer restrains available services to those that have an *Availability* higher than 80%.

3.2 Characteristics and dimensions hierarchies

Decomposition of QoS Characteristics into QoS Dimensions and QoS Dimensions into others QoS Dimensions may be used by the service composer to build a complete hierarchy of QoS properties. This information is expressed with help of the relations *Type - Typed* between the QoS Characteristic and the QoS Dimension metaclasses and *Compose - Composed by* defined over the QoS Dimension metaclass. The hierarchy established by the service composer allows to bind weights to QoS properties at different levels. This way, their relative importance is aggregated in accordance with the QoS properties that these quantify. To account for measurement of QoS Characteristics by QoS Dimensions and quantification of QoS Dimensions, we classify them into two separate hierarchies. The first is dedicated to benefits, all quality properties that have to be maximized: availability, reliability, reputation, etc. The second is designed for costs, involving quality properties to minimize: execution time, failures, cost, etc. Modality (maximize or minimize) of QoS properties is defined with the attribute direction of the QoS Value class. These two hierarchies are linked to the same global optimization goal. This top-down organization clearly indicates the contributions of lower levels of quality properties to upper ones. The final hierarchy obtained takes the form of a tree.
The second step of the service composer is to establish benefits and costs hierarchies with the information provided by the service requester. The hierarchy corresponding to expectations formulated by the requester for the ESA composite service is illustrated in Figure 3.2.

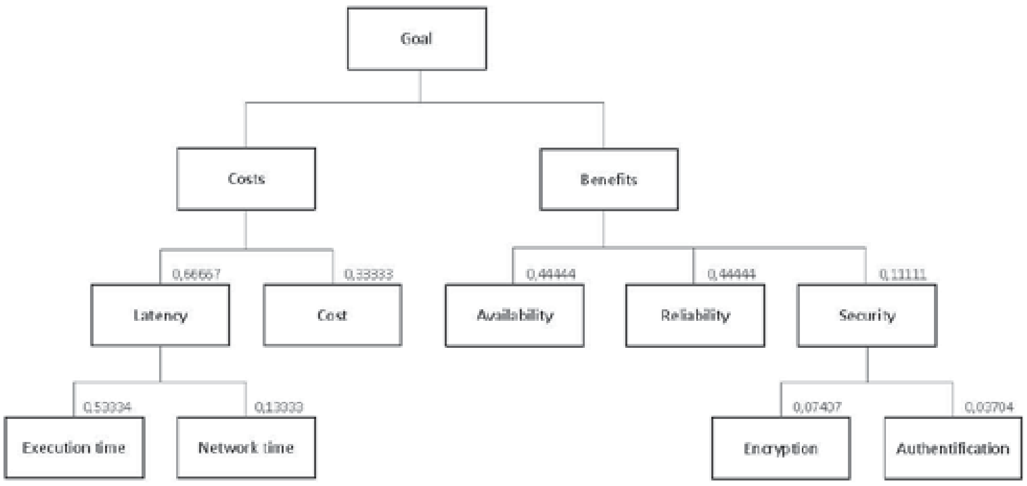


Fig. 7. Benefits and costs hierarchies

3.3 Priorities over criteria

Priorities information is used to bind weights to QoS Characteristics and QoS Dimensions, reflecting their respective relative importance. These weights are defined using QoS Priorities specifications given by the service user and are linked to the corresponding QoS properties. Once the hierarchy is established, the relative importance of each QoS property has to be fixed with a weight reflecting its contribution to the main optimization goal. These weights must be fixed independently for benefits criteria and for costs criteria to consider separately positive and negative QoS properties. To fix weights on such hierarchies, we use the Analytic Hierarchy Process (AHP) (Saaty, 1980). The Analytic Hierarchy Process fixes weights to criteria with help of comparison matrices provided for each level of criteria. For a same level, each criterion is compared with other criteria of its level on a scale fixed between 1/9 and 9. Each matrix is build with QoS Priority specifications: rules express direction of pairwise comparisons of criteria and strength fixes the value chosen by the user on the scale for the comparison. Next, weights of QoS properties are obtained with the computation of the right eigenvector of the matrix. The eigenvector is computed by raising the pairwise matrix to powers that are successively squared each time. The rows sums are then calculated and normalized. The computation is stopped when the difference between these sums in two consecutive calculations is smaller than a prescribed value. The service composer adopts a top-down approach, the weights of each level being multiplied by the weight of the quality property of its upper level to determine its relative importance on the whole hierarchy. This process is performed on both sides of the tree, for positive and negative quality properties.

The third step of the composer is to fix weights for each level of criteria with the AHP method. With the information provided by QoS Priority instance in Figure 2.3, the service composer is able to build a comparison matrix for dimensions quantifying the *Latency*. In the case or our composite service computing the FAPAR index for a given area of the world, the service requester favors the *Execution time* rather than the *Network time*. In fact, *Execution time* is the main bottleneck of the service execution due to huge quantity of data processed. This matrix is $\begin{pmatrix} 1 & 4 \\ 1/4 & 1 \end{pmatrix}$. The composer computes its eigenvector to obtain weights for this level, in the example: 0.2 for *Network Time* and 0.8 for *Execution Time*. These weights are multiplied by weights of upper levels to determine weights of the whole hierarchy that are illustrated in Figure 3.2.

3.4 QoS scoring with user preferences

Preferences information specified by the user on QoS Values is used by the service composer to compute the score of the service. We use this information to determine what values are preferred for a given QoS Characteristic or QoS Dimension. The priorities of quality properties have been fixed with weights reflecting their relative importance. Preferences on values allow us to discriminate services on a given criterion. To quantify these preferences, we rely on a specific class of MCDM methods: scoring methods (Figueira et al., 2005) and more specifically the Simple Additive Weighting (SAW) method (Hwang & Yoon, 1981). This method is based on the weighted average. An evaluation score is calculated for each alternative by multiplying the scaled value given to the alternative of that attribute with the weights given by the AHP method. Next, these products are summed for all criteria involved in the decision making process. Each service alternative is evaluated on both hierarchies, i.e.: benefits and costs, with the following formula:

$$s_u = \sum w_i \times x^*_{ui} \tag{1}$$

Where w_i is the weight of the QoS property *i* get with the AHP method and x^*_{ui} is the scaled score of the service alternative *u* on the QoS property *i*.
The scores for the QoS properties are measured with different scales, i.e.: percentage, second, level, etc. Such measurement scales must be standardized to a common dimensionless unit before applying the SAW method. The scaling of a service alternative for a given QoS property is evaluated with the following formula:

$$x^*_{ui} = \frac{x_{ui}}{x^{max}_i} \tag{2}$$

where x^*_{ui} is the scaled score of the service alternative *u* on the QoS property i. x_{ui} is the score of the service alternative *u* on the QoS property *i* expressed with its original unit. x^{max}_i is the maximal possible score on the QoS property *i*. This maximal score is expressed by the user with help of the *max value* attribute of the QoS Preference class illustrated in Figure 2.3. When the unit of the QoS Property is a percentage, the maximal value is systematically equal to 100. If the unit is a time period as second, the user defines himself the maximal value. So, the scaled scores will reflect the preferences of the user with means of the relative importance of the maximal value by contrast to observed values.
Once weights reflecting the relative importance of each QoS property have been fixed, the fourth step of the service composer is to define the score of each alternative for both benefits and costs hierarchies with user preferences. It uses the SAW method and begins by scaling the score of all alternatives on all QoS properties involved in the selection process. For example, in Figure 2.3, the max value proposed by the service user for the Network time is 20 sec. With a service alternative offering a Network Time of 13 sec, the scaled score of this service for the Network Time QoS property is 65%. This score is then multiplied by 0,13333, the weight of the Network Time. This process is summed for all QoS properties considered and repeated for all existing service alternatives on both hierarchies.

3.5 Benefits/costs analysis

Scores of services alternatives get with the SAW method on both hierarchies define the relative performance of services on positive properties (benefits) and negative properties (costs). Benefits should be maximized while costs have to be minimized, to aggregate both considerations into a single measure of performance, the AHP MCDM method proposes to execute the benefits/costs ratio (Figueira et al., 2005). The benefits/costs ratio is evaluated with the following formula:

$$r_u = \frac{s^{benefits}_u}{s^{costs}_u} \tag{3}$$

where r_u is the final score of the service alternative *u*. $s^{benefits}_u$ is the score of the service *u* on the benefits hierarchy and s^{costs}_u is the score of the service a on the costs hierarchy.

The last step of the composer is then to compute the benefits/costs ratio of each alternative as suggested by some AHP variations. E.g.: if a service alternative has a score of 0,8126 for its benefits hierarchy and a score of 0,7270 for its costs hierarchy, the final score of its service is $\frac{0,8126}{0,7270} = 1,1177$. The respective score of each service is then linked to reflect its relative performance.

4. Web services composition with randomized RL algorithm

An important issue is the selection of WS that are to participate in performing the process described in the composition model. This problem is referred to as the task allocation problem in the remainder.

Reinforcement Learning (RL) (see, e.g., (Sutton & Barto, 1998) for an introduction) is a particularly attractive approach to allocating tasks to WS. RL is a collection of methods for approximating optimal solutions to stochastic sequential decision problems (Sutton & Barto, 1998). An RL system does not require a teacher to specify correct actions. Instead, the learning agent tries different actions and observes the consequences to determine which are best. More specifically, in the RL framework, a learning agent interacts with an environment over some discrete time scale $t = 0,1,2,3,\ldots$. At each time step t, the environment is in some state, k_t. The agent chooses an action, u_t, which causes the environment to transition to state k_{t+1} and to emit a feedback, r_{t+1}, called ``reward''. A reward may be positive or negative, but must be bounded and it informs the agent on the performance of the selected actions. The next state and reward depend only on the preceding state and action, but they may depend on it in a stochastic fashion. The objective of reinforcement learning is to use observed rewards to learn an optimal (or nearly optimal) mapping from states to actions, which is called an optimal policy, Π. An optimal policy is a policy that maximizes the expected total reward (see, § 4.2, Eq. 5). More precisely, the objective is to choose action u_t, for all $t \leq 0$, so as to maximize the expected return. Using the terminology of this paper, RL can be said to refer to trial-and-error methods in which the composer learns to make good allocations of WS to tasks through a sequence of " interactions" . In task allocation, an interaction consists of the following:

1. The composer identifies the task to which a WS is to be allocated.
2. The composer chooses the WS to allocate to the task.
3. The composer receives a reward after the WS executes the task. Based on the reward, the composer learns whether the allocation of the given WS to the task is appropriate or not.
4. The composer moves to the next task to execute (i.e., the next interaction takes place).

One advantage of RL over, e.g., queuing-theoretic algorithms (e.g., (Urgaonkar et al., 2005)), is that the procedure for allocating WS to tasks is continually rebuilt at runtime: i.e., the composition procedure changes as the observed outcomes of prior composition choices become available. The WS composer tries various allocations of WS to tasks, and learns from the consequences of each allocation. Another advantage is that RL does not require an explicit and detailed model of either the computing system whose operation it manages, nor of the external process that generates the composition model. Finally, being grounded in Markov Decision Processes, the RL is a sequential decision theory that properly treats the possibility that a decision may have delayed consequences, so that the RL can outperform alternative approaches that treat such cases only approximately, ignore them entirely, or cast decisions as a series of unrelated optimizations.

One challenge in RL is the tradeoff between *exploration* and *exploitation*. Exploration aims to try new ways of solving the problem, while exploitation aims to capitalize on already well-established solutions. Exploration is especially relevant when the environment is changing: good solutions can deteriorate and better solutions can appear over time. In WS composition, exploitation consists of learning optimal allocations of WS to tasks, and systematically reusing learned allocations. Without exploration, the WS composer will not consider allocations different than those which proved optimal in the past. This is not desirable, since in absence of exploration, the WS composer is unaware of changes in the availability of WS and appearance of new WS, so that the performance at which the composition is fulfilled inevitably deteriorates over time in an open and distributed service-oriented system.

Two forms of exploration can be applied: *preliminary* and *continual online* exploration. The aim with *preliminary* exploration is to discover the state to reach, and to determine a first optimal way to reach it. As the composition model specifies the state to reach in WS composition, *continual online* exploration is of particular interest: therein, the set of WS that can be allocated to tasks is continually revised, so that future allocations can be performed by taking into account the availability of new WS, or the change in availability of WS used in prior compositions. Preliminary exploration is *directed* if domain-specific knowledge is used to guide exploration (e.g., (Thrun, 1992b; Thrun, 1992a; Thrun et al., 2005; Verbeeck, 2004)). In *undirected* preliminary exploration, the allocation of new WS to tasks is randomized by associating a probability distribution to the set of competing WS available for allocation to a given task.

To avoid domain-specificity in this paper, the RL algorithm in MCRRL relies on *undirected continual exploration*. Both exploitation and undirected continual exploration are used in WS composition: exploitation uses available data to ground the allocation decision in performance observed during the execution of prior compositions, whereas exploration introduces new allocation options that cannot be identified from past performance data. This responds to the first requirement on WS composition procedures (item 1, § 1), namely that optimal WS compositions will be built and revised at runtime, while accounting for change in the availability of WS and the appearance of new WS. As shown in the remainder (see, § 4.1), the WS composition problem can be formulated as a global optimization problem which follows either a *deterministic shortest-path* (in case the effects of WS executions are deterministic) or a *stochastic shortest-path* formulation. Requirement 4 (§ 1) is thus also addressed through the use of RL to guide WS composition. Since the RL approach can be based on observed performance of WS in compositions, and the algorithm in MCRRL accepts multiple criteria and/or constraints (see, § 3 and § 4.1), requirements 2 and 3 (§ 1) are fulfilled as well.

4.1 Task-allocation problem

If RL is applied to task allocation, the exploration/ exploitation issue can be addressed by periodically readjusting the policy for choosing task allocations and re-exploring up-to-now suboptimal execution paths (Mitchell, 1997; Sutton & Barto, 1998). Such a strategy is, however, suboptimal because it does not account for exploration. The Randomized

Reinforcement Learning (RRL) algorithm introduced in (Saerens et al., 2004) is adapted herein to task allocation in WS composition, allowing the assignment of tasks to WS while: (i) optimizing criteria, (ii) satisfying the hard constraints, (iii) learning about the performance of new agents so as to continually adjust task allocation, and (iv) exploring new options in task allocation. The exploration rate is quantified with the Shannon entropy associated to the probability distribution of allocating a task to a task specialist. This permits the continual measurement and control of exploration.

The task-allocation problem that the RRL resolves amounts to the composer determining the WS to execute the tasks in a given process model. By conceptualizing the process of the composition model as a DAH (see, § 2.2.2), the task-allocation problem amounts to a deterministic shortest-path problem in a *directed weighted hypergraph*. In the hypergraph, each node is a *step* in WS composition problem and an edge corresponds to the *allocation of a task* t_k *to a WS* $w_{k,u}^{WS}$, where u ranges over WS that can execute t_k according to the criteria set with the QoS model. Each individual allocation of a task to a WS incurs a cost $c(t_k, w_{k,u}^{WS})$, whereby this " cost" is a function of the aggregated criteria (as discussed earlier § 3) formulated so that the minimization of cost corresponds to the optimization of the aggregated criteria (i.e., minimization or maximization of aggregation value). For illustration, consider the DAH representation of our composite ESA service in Figure 3.

The task allocation problem is a global optimization problem: learn the optimal complete probabilistic allocation that minimizes the expected cumulated cost from the initial node to the destination node while maintaining a fixed degree of exploration, and under a given set of hard constraints (specified with the QoS model). At the initial node in the graph (in Fig.3, blank node), no tasks are allocated, whereas when reaching the destination node (last 'Pd' node in the same figure), all tasks are allocated.

The remainder of this Section is organized as follows: § 4.2 introduces the notations, the standard deterministic shortest-path problem, and the management of continual exploration. § 4.3 introduces the unified framework integrating exploitation and exploration presented in (Achbany et al., 2005). Finally, § 4.3 describes our procedure for solving the deterministic shortest-path problem with continual exploration.

4.2 RL formulation of the problem

At a state k_i of the task allocation problem, choosing an allocation of $t_{k_i,l}$ (where l ranges over tasks available in state k_i) to $w_{k_i,u}^{WS}$ (i.e., moving from k_i to another state) from a set of potential allocations $U(k_i)$ incurs a cost $c(t_{k_i,l}, w_{k_i,u}^{WS})$. Cost is an inverse function of the aggregated criteria the user wishes to optimize (see, § 3), say r. The cost can be positive (penalty), negative (reward), and it is assumed that the service graph is acyclic (Christofides, 1975). Task allocation proceeds by comparing WS over estimated $\hat{r}$ values and the hard constraints to satisfy (see, s 3.1). The allocation $(t_{k_i,l}, w_{k_i,u}^{WS})$ is chosen according to a *Task Allocation policy (TA)* Π that maps every state k_i to the set $U(k_i)$ of admissible allocations with a certain probability distribution $\pi_{k_i}(u)$, i.e., $U(k_i)$: $\Pi \equiv \{\pi_{k_i}(u), i = 0,1,2,\ldots,n\}$. It is assumed that: (i) once the action (i.e., allocation of a given task to a WS) has been chosen, the sate next to k_i, denoted $k_{i'}$, is known deterministically, $k_{i'} = f_{k_i}(u)$ where f is a one-to-one mapping from states and actions to a resulting state; (ii) different actions lead to different states; and (iii) as in (Bertsekas, 2000), there is a special cost-free *destination* state;

once the composer has reached that state, the task allocation process is complete. Although the current discussion focuses on the deterministic case, extension to the stochastic case is discussed elsewhere (Achbany et al., 2005) due to format constraints.
As remind, one of the key features of reinforcement learning is that it explicitly addresses the exploration/exploitation issue as well as the online estimation of the probability distributions in an integrated way. Then, the exploration/ exploitation tradeoff is stated as a global optimization problem: find the exploration strategy that minimizes the expected cumulated cost, while maintaining fixed degrees of exploration at same nodes. In other words, exploitation is maximized for constant exploration. To control exploration, entropy is defined at each state.
The degree of exploration E_{k_i} at state k_i is quantified as:

$$E_{k_i} = -\sum_{u \in U(k_i)} \pi_{k_i}(u) \log \pi_{k_i}(u) \tag{4}$$

which is the entropy of the probability distribution of the task allocations in state k_i (Cover & Thomas, 1991; Kapur & Kesavan, 1992). E_{k_i} characterizes the uncertainty about the allocation of a task to a WS at k_i. It is equal to zero when there is no uncertainty at all ($\pi_{k_i}(u)$ reduces to a Kronecker delta); it is equal to $\log(n_{k_i})$, where n_{k_i} is the number of admissible allocations at node k_i, in the case of maximum uncertainty, $\pi_{k_i}(u) = 1/n_{k_i}$ (a uniform distribution).
The exploration rate $E^r_{k_i} \in 0,1]$ is the ratio between the actual value of E_{k_i} and its maximum value: $E^r_{k_i} = E_{k_i} / \log(n_{k_i})$.
Fixing the entropy at a state sets the exploration level for the state; increasing the entropy increases exploration, up to the maximal value in which case there is no more exploitation---the next action is chosen completely at random (using a uniform distribution) and without taking the costs into account. Exploration levels of composers can thus be controlled through exploration rates. Service provision then amounts to minimizing *total expected cost* $V_\pi(k_0)$ accumulated over all paths from the initial k_0 to the final state:

$$V_\pi(k_0) = E_\pi\left[\sum_{i=0}^{\infty} c(k_i, u_i)\right] \tag{5}$$

The expectation E_π is taken on the policy Π that is, on all the random choices of action u_i in state k_i.

4.3 Computation of the Optimal Policy

The composer begins with task allocation from the initial state and chooses from state k_i the allocation of a WS u to a task $t_{k_i,l}$ with a probability distribution $\pi_{k_i}(u)$, which aims to exploration. The composer then performs the allocation of the task $t_{k_i,l}$ to a WS u and the associated aggregated quality score, the cost $c(t_{k_i,l}, w^{WS}_u)$ is incurred and is denoted, for simplicity $c(k_i, u)$ (note that this score may also vary over time in a dynamic environment); the composer then moves to the new state, $k_{i'}$. This allows the composer to update the estimates of the aggregated quality score of the policy, and of the average aggregated quality value until destination; these estimates will be denoted by $c(k_i, i)$, $\pi_{k_i}(i)$ and $V(k_i)$. The RRL for an acyclic graph, where the states are ordered in such a way that there is no edge going backward (i.e., there exists no edge linking a state $k_{i'}$ to a state k_i where $k_{i'}$ is a successor state of k_i ($k_{i'} > k_i$), is as follows (a detailed treatment can be found in (Achbany et al., 2005)):

1. *Initialization phase:* Set $V(k_d) = 0$, which is the expected cost at the destination state.
2. *Computation of the TA policy and the expected cost under exploration constraints:* For $k_i = (k_d - 1)$ to the initial state k_0, compute:

$$\begin{cases} \pi_{k_i}(u) = \dfrac{\exp\left[-\theta_{k_i}\left(c(k_i,u)+V(k'_{i,u})\right)\right]}{\sum\limits_{u' \in U(k_i)} \exp\left[-\theta_{k_i}\left(c(k_i,u')+V(k'_{i,u'})\right)\right]}, \\ V(k_i) = \sum\limits_{u \in U(k_i)} \pi_{k_i}(u)\left[c(k_i,u)+V(k'_{i,u})\right] \text{ for } k_i \neq k_d \end{cases} \tag{6}$$

where $k'_{i,u} = f_k(u)$, $k'_{i,u'} = f_k(u')$ and θ_{k_i} is set in order to respect the prescribed degree of entropy at each state (see Eq.4 which can be solved by a simple bisection search).

Various approaches can be applied to update the estimated criterion $\hat{r}_u$; e.g., exponential smoothing leads to:

$$r_u \leftarrow \alpha \overline{r}_u + (1-\alpha) r_u \tag{7}$$

where $\overline{r}_u$ is the observed value of the criterion for w_u^{WS} and $\alpha \in]0,1[$ is the smoothing parameter. Alternatively, various stochastic approximation updating rules could also be used. The composer updates its estimates of the criterion each time a WS performs a task and the associated cost is updated accordingly.

5. Simulation results

Experimental setup. Task allocation for the service provision problem diplayed in Fig.3 was performed. A total of three distinct WS were made available for each distinct task. Each $w_{k,u}$ is characterized by its actual r_u which is an indicator of the WS's performance over the optimization criterion (see, § 4.2). In this simulation, it will simply be the probability of successfully performing the task (1 -- probability of failure). In total, 42 WS are available to the Composer for task allocation. For all WS u, r_u takes its value $\in 0,1]$; for 70% of the WS, the actual r_u is *hidden* (assuming it is unknown to the Composer) and its initial expected value, r_u, is set, by default, to 0.3 (high probability of failure since the behavior of the WS has never been observed up to now), while actual r_u value is available to the Composer for the remaining 30% (assuming these WS are well known to the Composer). Actual r_u is randomly assigned from the interval $[0.5, 1.0]$ following a uniform probability distribution. It has been further assumed that $c(t_i, w_u) = -ln(r_u)$, meaning that it is the product of the r_u

along a path that is optimized (this is a standard measure of the reliability of a system). After all tasks are allocated, the selected WS execute their allocated tasks according to their actual r_u value (with failure $1-r_u$). The estimated WS criterion $\hat{r}_u$ is then updated by exponential smoothing, according to Eq.7. In Eq.7, $\overline{r}_u$ equals 1 if w_u is successful at executing the task it has been allocated, 0 otherwise. Estimated costs are of course updated in terms of the r_u and each time a complete allocation occurs, the probability distributions of choosing a WS are updated according to Eq.6. 10,000 complete allocations were simulated for exploration rate 20%.

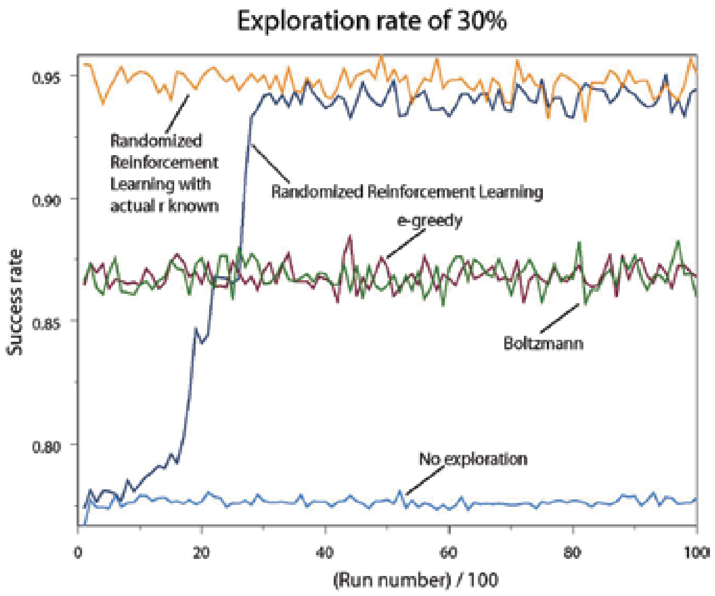


Fig. 8. Success rate in terms of run number, for an exploration rate of 20%, and for the five methods (no exploration, actual r known, ε *-greedy*, naive Boltzmann, RRL).

Results. The RRL is compared to two other standard exploration methods, ε *-greedy* and *naive Boltzmann* (see (Achbany et al., 2005) for details), while tuning their parameters to ensure the same exploration level as for RRL. The *success rate* is defined as the proportion of services that are successfully completed (i.e., all tasks composing the service are allocated and executed successfully) and is displayed in Fig. 4 in terms of the run number (one run corresponding to one complete assignment of tasks, criterion estimation and probability distribution update). Fig. 4 shows the RRL behaves as expected. Its performance converges almost to the success rate of the RRL in which all actual r are known from the outset (i.e., need not be estimated)---and indicate that exploration clearly helps by outperforming the allocation system without exploration (which has a constant 75% success rate). Fig.5 compares the three exploration methods by plotting the average absolute difference between actual r_u and estimated r_u criterion values for a 30% exploration rate. Exploration is therefore clearly helpful when the environment changes with the appearance of new agents---i.e., exploration is useful for directing Composer behavior in dynamic, changing, and open architectures, i.e., in the SCA.

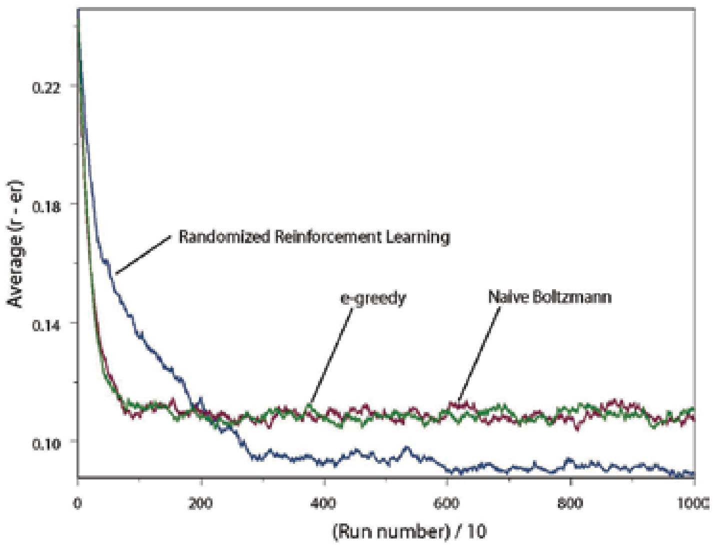


Fig. 9. Average absolute difference between actual (r) and estimated (r) criterion values in terms of run number, for three exploration methods (ε -greedy, naive Boltzmann, RRL).

6. Related work

Various possibilities for representation of web services composition have already been addressed. Jaeger et al. use composition patterns (sequence, loop, xor, and, or , etc.) to represent structural elements of the composition. Hamadi (Hamadi & Benatallah, 2003) and Benetallah and Fu et al. (Fu et al., 2006) approaches refer both to Petri nets for modeling web services control flow. Rather than using such patterns and their associated aggregation rules, we choose, as Zeng et al. (Zeng et al., 2003b; Zeng et al., 2004), Benatallah and Dumas (Benatallah et al., 2002) and Zhang et al. (Zhang et al., 2007), to control services composition with the help of statecharts.

While statecharts and their associated formal semantic are used to represent the different tasks entering in the composition, Directed Acyclic Graph (DAG) (Gu & Nahrstedt, 2002) are used to represent alternative web services allowing to fulfill these tasks. Zeng (Zeng et al., 2004) proposes to model alternatives with multiple execution paths derived from statecharts possibilities. We choose to represent our composition possibilities with a Directed Acyclic Hypergraph (DAH) where nodes represent functional steps of execution and edges are web services alternatives to fulfill a given task.

Our selection of services that will enter in the services composition is based on their quality properties, i.e., their QoS. To lead the selection from the requester view, we provide him a QoS model enabling to specify its expectations about quality behavior. This behavior is expressed by relationships between characteristics and dimensions, priorities between quality properties and preferences over values. The preference over values has already been addressed in other approaches under the form of a direction attribute indicating if a property has to be maximized or minimized (Jaeger et al., 2004; Liu et al., 2004; Naumann et al., 1999; Zeng et al., 2004). Our preference structure offers more information, we allow the user to specify conditions and indifference thresholds. The priority relationship is defined in some proposals with means of a weight attribute associated to quality properties (Jaeger et al., 2004; Zeng et al., 2004). Our model authorizes weights binded to quality properties at different levels and we define a method to fix adequately these weights.

Most QoS composition approaches aim at summing QoS values of services entering in the composition rather than computing their individual performance (Cardoso et al., 2004; Cheng et al., 2006; Jaeger et al., 2005; Yu & Lin, 2004; Yu & Lin, 2005; Zeng et al., 2003b; Zhang et al., 2007). In our MCRRL proposal, we focus on the individual evaluation of each web service candidate to the whole composition. Rather than using Reinforcement Learning computation, Zeng and colleagues (Zeng et al., 2003b) proceed to finding optimal WS compositions through linear programming techniques. In contrast to RL, their approach considers each WS composition as a new problem to solve, so that there is no learning. Canfora and colleagues (Canfora et al., 2004) use genetic algorithms, avoiding thus the need for a linear objective function and/or linear constraints in the search for the optimal WS composition (required for the linear programming approach (Zeng et al., 2003b)). MCRRL improves responsiveness of the system to varying availability and appearance of new WS because of exploration. MCRRL allows the execution of potentially complex processes, permits concurrency, while assuming that the set of available WS is changing. One distinctive characteristic the composer's behavior suggested in the present paper is that the MCRRL accounts for a vector of criteria when allocating tasks, including QoS, service provision deadline, provision cost, explicit user preferences, and agent reputation. Feedback mechanisms are also used by Maximilien and Singh (Maximilien & Singh, 2005) that propose service selection driven by trust values assigned to individual services. Trust is extracted from user-generated reports of past service performance (as usual in reputation systems) over qualities defined by a system-specific QoS ontology. Level of trust depends on the degree to which reputation and quality levels advertised by the provider match. Similar approaches have been proposed, yet fail to address service selection in open, distributed MAS architecture, furthermore without dynamic allocation so that autonomic requirements are not fulfilled. By basing selection on trust only and generating levels of trust from advertised and user-observed behavior, Maximilien and Singh's approach involves learning driven by exploitation of historical information, without exploration.

7. Conclusions and future work

This paper advocates that WS compositions optimal w.r.t. a set of criteria need to be learned at runtime and revised as new WS appear and availability of old WS changes, whereby the learning should be based on observed WS performance, and not the performance values advertised by the service providers. To enable such learning, a selection procedure is needed which both *exploits* the data on observed WS performance in the past, and *explores* new composition options to avoid excessive reliance on past data.

As a response, this paper proposes the Multi-Criteria Randomized Reinforcement Learning (MCRRL) approach to WS composition. MCRRL combines a generic service request and the

Randomized Reinforcement Learning (RRL), a reinforcement learning algorithm. The SR model describes the process to execute by the WS composition and the criteria and constraints to meet when executing it. The RRL selects the WS for performing tasks specified in the service request. The algorithm decides on the WS to select among competing WS based on multiple criteria, while both exploiting available WS performance data and exploring new composition options.

MCRRL responds to four common requirements when defining a task allocation procedure for WS composition. First, the RRL uses both exploitation and undirected continual exploration in WS composition: exploitation uses available data to ground the allocation decision in performance observed during the execution of prior compositions, whereas exploration introduces new allocation options that cannot be identified from past performance data. Optimal WS compositions are thus identified revised at runtime. Second, the generic SR model combined with the optimization approach in the RRL allow many criteria for comparing alternative task allocations. Third, the comparison over various criteria relies on observed performance over the given criteria, instead of vales advertised by service providers. Finally, the algorithm can be extended to allow underterministic outcomes of WS executions (as explained elsewhere (Achbany et al., 2005)).

Since undirected exploration may be costly in actual applications, future work will investigate the performance of MCRRL within realistic applications, so that the approach can be optimized for practical settings.

8. Acknowledgments

We are grateful to Emmanuel Mathot of the European Space Agency, who provided precise information about the GPOD project and assisted our efforts in describing quality information of services related to the GPOD project.

9. References

Achbany, Y.; Fouss, F.; Yen, L.; Pirotte, A. & Saerens, M. (2005) Tuning Continual Exploration in Reinforcement Learning. *Technical report, http://www.isys.ucl.ac.be/staff/francois/Articles/ Achbany2005a.pdf.*

Benatallah, B.; Sheng, Q. Z.; Ngu, A. H. & Dumas M. (2002) Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. *Proceedings of the International Conference on Data Engineering*, pages 0297, Los Alamitos, CA, USA.

Bertsekas, D. P. (2000) *Dynamic programming and optimal control*. Athena sientific.

Canfora, G.; Di Penta, M.; Esposito, R. & Villani, M.-L. (2004) A Lightweight Approach for QoS-Aware Service Composition. *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04)*, pages 36-47.

Cardoso, J.; Sheth, A. P.; Miller, J. A.; Arnold, J. & Kochut, K. (2004) Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281-308.

Chen , Y.P.; Zeng-Zhi, L.; Qin-Xue, J. & Chuang, W. (2006) Study on QoS Driven Web Services Composition. *Frontiers of WWW Research and Development - APWeb 2006,* :702--707.

Christofides, N. (1975) *Graph theory: An algorithmic approach*. Academic Press.

Cover, T. M. & Thomas, J. A. (1991) *Elements of information theory*. John Wiley and Sons.

D'Ambrogio, A. (2006) A Model-driven WSDL Extension for Describing the QoS ofWeb Services. *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 789--796, Washington, DC, USA, 2006, IEEE Computer Society.

Figueira, J.; Greco, S. & Ehrgott, M. (2005) *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer Verlag, Boston, Dordrecht, London.

Fu, Y.; Zhijiang, D. & Xudong, H. (2006) Modeling, validating and automating composition of web services. *ICWE '06: Proceedings of the 6th international conference on Web engineering*, pages 217--224, New York, NY, USA, 2006. ACM.

Gu, X. & Nahrstedt, K. (2002) A scalable QoS-aware service aggregation model for peer-to-peer computing grids. *Proceedings of the IEEE HPDC-11*, 2002.

Hamadi, R. & Benatallah, B. (2003) A Petri net-based model for web service composition. *ADC '03: Proceedings of the 14th Australasian database conference*, pages 191--200, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

Hwang, C.L. & Yoon, K. (1981) *Multiple attribute decision making : Methods and applications*. Springer-Verlag, 1981.

Jaeger, M. C.; Rojec-Goldmann, R. & Muhl, G. (2004) QoS Aggregation for Web Service Composition using Workflow Patterns. *edoc*, 00:149-159.

Jaeger, M. C.; Rojec-Goldmann, R. & Muhl, G. (2005) QoS Aggregation in Web Service Compositions. *EEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pages 181--185, Washington, DC, USA, 2005. IEEE Computer Society.

Kapur, J. N. & Kesavan, H. K. (1992) *Entropy optimization principles with applications*. Academic Press, 1992.

Keller, A. & Ludwig, H. (2003) The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *J. Netw. Syst. Manage.*, 11(1):57--81.

Liu, Y.; Ngu, A. H. & Zeng, L. Z. (2004) QoS Computation and Policing in Dynamic Web Service Selection. pages 66--73, New York, NY, USA, ACM Press.

Maximilien, E. M. & Singh, M. P. (2005) Multiagent System for Dynamic Web Services Selection. *Proceedings of the Int. Conf. Auton. Agents and Multi-Agent Syst.*, 2005.

McIlraith, S. A. & Martin, D. L. (2003) Bringing Semantics to Web Services. *IEEE Intelligent Systems*, 18(1):90--93.

Menascé, D. A. (2002) QoS Issues in Web Services. *IEEE Internet Computing*, 6(6):72--75, 2002.

Mitchell, T. M. (1997) *Machine learning*. McGraw-Hill Compagnies, 1997.

Naumann, F.; Leser, U. & Freytag, J. C. (1999) Quality-driven Integration of Heterogenous Information Systems. *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 447-458, Edinburgh, UK, 1999.

OMG / Business Process Management Initiative. Business Process Modeling Notation Specification. Final Adopted Specification dtc/06-02-01, 2006.Technical report, Object Management Group / Business Process Management Initiative., 2006.

OMG. UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms. Technical report, Object Management Group, 2006.

OMG. UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms. Technical report, Object Management Group, 2006.

Papazoglou, M. P. & Georgakopoulos, D. (2003) Service-oriented computing. *Commun. ACM*, 46(10).

Saaty, T.L. (1980) *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. McGraw-Hill, New york.

Saerens, M.; Souchon, N.; Renders, J. M. & Decaestecker, C. (2004) Decision-making under uncertainty about the class priors. *Submitted for publication*.

Sutton, R.S. & Barto, A.G. (1998) *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Thrun, S. (1992) The Role of Exploration in Learning Control. In D.A. White and D.A. Sofge, editors, *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky 41022.

Thrun, S.; Burgard, W. & Fox, D. (2005) *Probabilistic Robotics*. MIT Press.

Thrun, S. (1992) Efficient Exploration In Reinforcement Learning. Technical report, Pittsburgh, PA, USA.

Urgaonkar, B.; Pacifici, G.; Shenoy, P.; Spreitzer, M. & Tantawi, A. (2005) An analytical model for multi-tier internet services and its applications. *SIGMETRICS Perform. Eval. Rev.*, 33(1):291--302.

Verbeeck, K. (2004) Coordinated Exploration in Multi-Agent Reinforcement Learning. PhD thesis, Vrije Universiteit Brussel, Belgium.

Walsh, A. E. (2002) *Uddi, Soap, and Wsdl: The Web Services Specification Reference Book*. Prentice Hall Professional Technical Reference, 2002.

Tao, Y. & Kwei-Jay, L. (2005) Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. *Service-Oriented Computing - ICSOC 2005*, :130--143.

Tao, Y. & Kwei-Jay, L. (2004) Service Selection Algorithms for Web Services with End-to-End QoS Constraints. *CEC '04: Proceedings of the IEEE International Conference on E-Commerce Technology (CEC'04)*, pages 129--136, Washington, DC, USA, 2004. IEEE Computer Society.

Zacharia, G. & Maes, P. (2000) Trust Management Through Reputation Mechanisms. *Applied Artificial Intelligence*, 14:881-907.

Zeng, L.; Benatallah, B.; Dumas, M.; Kalagnanam, J. & Sheng, Q. Z. (2003) Quality Driven Web Services Composition. *Proc. Int. Conf. on World Wide Web (WWW2003)*, 2003.

Zeng, L.; Benatallah, B.; Ngu, A. H.; Dumas, M.; Kalagnanam, J. & Sheng, H. (2004) QoS-Aware Middleware for Web Services Composition. *IEEE Trans. Softw. Eng.*, 30(5):311--327.

Zeng, L; Jeng, J.-J.; Kumaran, S. & Kalagnanam, J. (2003) Reliable Execution Planning and Exception Handling for Business Process. *Technologies for E-Services*, pages 119-130.

Zhang, C.; Chang, R. N.; Perng, C.-S.; So, E.; Tang, C. & Tao, T. (2007) QoS-Aware Optimization of Composite-Service Fulfillment Policy. *Services Computing, 2007. SCC 2007. IEEE International Conference on*, :11--19.

Zhou, C.; Chia, L.-T. & Lee, B.-S. (2004) DAML-QoS Ontology for Web Services. *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 472, Washington, DC, USA, 2004. IEEE Computer Society.

Model Selection for Ranking SVM Using Regularization Path

Karina Zapien[1], Gilles Gasso[1], Thomas Gärtner[2] and Stéphane Canu[1]
[1] LITIS EA 4108 - INSA de Rouen
[2] Fraunhofer IAIS,
[1]France
[2]Germany

1. Introduction

This chapter deals with supervised learning problems under the ranking framework. Ranking algorithms are typically introduced as a tool for personalizing the order in which document recommendations or search results - in the web, for example - are presented. That is, the more important a result is to the user, the earlier it should be listed. To this end, two possible settings can be considered :

i. the algorithm tries to interactively rearrange the results of one search such that relevant results come the closer to the top the more (implicit) feedback the user provides,
ii. the algorithm tries to generalize over several queries and presents the results of one search in an order depending on the feedback obtained from previous searches.

The first setting deals with an active learning while the second setting deals with a passive supervised learning. This kind of problems have gain major attention given the nowadays amount of available informations. This is without doubt a challenging task in the medium and large scale context.

Several methods have been proposed to solve these problems. For the passive setting, the Rankboost algorithm (Freund et al. (2003)) is an adaptation from the Adaboost algorithm to the ranking problem. This is a boosting algorithm which works by iteratively building a linear combination of several "weak" algorithms to form a more accurate algorithm. The Pranking algorithm (Crammer & Singer (2001)) is an online version of the weighted algorithm. The SVRank and RankSVMalgorithms are the adaptation of the Support Vector machines for classification and regression, respectively, while the MPRank (Cortes et al. (2007)) is a magnitude-preserving algorithm, which searches not only to keep the relative position of each sample but also to preserve the distance given by the correct ordering. This last algorithm has as well the form of a regularization problem as the two previous with a different cost function.

Later, the Ranking SVM (RankSVM) algorithm was proposed by Herbrich et al. (2000) and Joachims (2002) as an optimization problem with constraints given by the induced graph of the ordered queries' results. This algorithm forms part of the family of kernel algorithms of the SVM type (Boser et al. (1992); Schölkopf & Smola (2002)).

Kernel methods like the SVM or the ranking SVM solve optimization problems of the form

$$\hat{f}_\lambda = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \mathcal{V}(f) + \lambda\Omega(f) \tag{1}$$

where $\mathcal{V} : \mathcal{H} \to \mathbb{R}^+$ is a loss function, $\lambda \in \mathbb{R}^+$ is a regularization parameter, Ω: $\mathcal{H} \to \mathbb{R}^+$ is the regularizer (which allows to enforce some nice properties as smoothness or simplicity of f) and $\mathcal{H}$ represents the hypothesis space. Usually $\mathcal{H}$ is chosen as a reproducing kernel Hilbert space. Although a key bottleneck for applying such algorithms in the real-world is choosing λ, research often ignores this. As empirical results, however, strongly depend on the chosen λ, runtime intensive repeated cross-validations have to be performed. Hence, in this chapter we concentrate on speeding up and automating this choice by building on the *regularization path* for SVMs (Hastie et al. (2004)).

2. Piecewise linear solutions

This framework is a kind of a more generic regularized optimization problems, already studied for regularization problems (Rosset & Zhu (2007)) and for parametric quadratic programming (Markowitz (1959)) for portfolio optimization. We are interested by the efficient computation of the regularization path. Hence, let us define first this notion.

Definition 2.1 (Regularization path)

The regularization path of Problem (1) is the set of all solutions obtained when varying λ over $\mathbb{R}^+$ *i.e. Path* = $\{f_\lambda$, *with* $\lambda \in [0,+\infty]\}$.

As one can see, with this definition, the pursued policy can have a high computational price. In order to gain in efficiency, the family of piecewise linear solution path is of particular interest. To highlight this fact, we consider the following definition.

Definition 2.2 (piecewise linear solution path)

The solution path is said to be piecewise linear when there exists a strictly decreasing (or increasing) sequence λ^t, $t = 1, \dots, N$ such that :

$$\hat{f}_\lambda = \hat{f}_{\lambda^t} + (\lambda - \lambda^t)\, h^t \qquad \forall \lambda \in [\lambda^{t+1}, \lambda^t] \;\; (\textit{resp. in } [\lambda^t, \lambda^{t+1}]) \tag{2}$$

where h^t, $t = 1, \dots, N$ *denotes a sequence of functions in* $\mathcal{H}$.

With such property, it is easy to efficiently generate the whole path of solution. Indeed, in such case, one only needs the sequence λ^t and the corresponding h^t. Any other functions in-between can be simply obtained by linear interpolation. Hence, owing to such property, the computational cost of obtaining the whole path of solution may be of the order of a single solution computation.

The question induced by this remark is to find which kind of objective functions makes the solution path piecewise linear. In Rosset and Zhu (2007), the necessary conditions were given for Problem (1) to admit a linear solution path. The main result is summarized by the theorem below.

Theorem 1

Assume the loss $\mathcal{V}(f)$ *and the regularizer* $\Omega(f)$ *are convex functions. If one objective function (either* $\mathcal{V}(f)$ *or* $\Omega(f)$*) is piecewise linear and the other one piecewise quadratic then the solution path of the Problem (1) is piecewise linear.*

Proof Assume $\mathcal{V}$(f) and Ω(f)) are twice differentiable in a neighborhood of f_{λ^t} solution of (1) corresponding to λ^t. Let also $\lambda = \lambda^t + \delta\lambda$ and its related solution f_λ. Consider finally J(f) = $\mathcal{V}(f_\lambda) + \lambda\, \Omega(f_\lambda)$. The optimality conditions associated to f_{λ^t} and f_λ are respectively

$$\nabla_{\mathsf{f}}\, \mathsf{J}(\mathsf{f}_{\lambda^t}) \quad = \quad \nabla_{\mathsf{f}}\, \mathcal{V}(\mathsf{f}_{\lambda^t}) + \lambda^t\, \nabla_{\mathsf{f}}\, \Omega(\mathsf{f}_{\lambda^t}) = 0 \tag{3}$$

$$\nabla_{\mathsf{f}}\, \mathsf{J}(\mathsf{f}_{\lambda}) \quad = \quad \nabla_{\mathsf{f}}\, \mathcal{V}(\mathsf{f}_{\lambda}) + \lambda\, \nabla_{\mathsf{f}}\, \Omega(\mathsf{f}_{\lambda}) = 0 \tag{4}$$

where ∇_f J(f) represents the functional derivative of J in $\mathcal{H}$. For small values of $\delta\lambda$ we can consider the following second order Taylor expansion of (4) around f_{λ^t}

$$\nabla_{\mathsf{f}}\, \mathcal{V}(\mathsf{f}_{\lambda^t}) + \delta\lambda\, \mathsf{f}\prime_{\lambda^t}\, \nabla^2_{\mathsf{f}}\, \mathcal{V}(\mathsf{f}_{\lambda^t}) + \lambda^t \left(\nabla_{\mathsf{f}}\Omega(\mathsf{f}_{\lambda^t}) + \delta\lambda\, \mathsf{f}\prime_{\lambda^t}\, \nabla^2_{\mathsf{f}}\Omega(\mathsf{f}_{\lambda^t})\right) + \delta\lambda\, \nabla_{\mathsf{f}}\Omega(\mathsf{f}_{\lambda^t}) = 0$$

with $\mathsf{f}_\lambda = \mathsf{f}_{\lambda^t} + \delta\lambda \mathsf{f}\prime_{\lambda^t}$. Using it we have the following limit

$$\lim_{\delta\lambda \to 0} \frac{\nabla_{\mathsf{f}}\, \mathsf{J}(\mathsf{f}_\lambda) - \nabla_{\mathsf{f}}\, \mathsf{J}(\mathsf{f}_{\lambda_t})}{\delta\lambda} = \mathsf{f}\prime_{\lambda^t}\, \nabla^2_{\mathsf{f}}\, \mathcal{V}(\mathsf{f}_{\lambda^t}) + \lambda^t\, \mathsf{f}\prime_{\lambda^t}\, \nabla^2_{\mathsf{f}}\Omega(\mathsf{f}_{\lambda^t}) + \nabla_{\mathsf{f}}\Omega(\mathsf{f}_{\lambda^t}) = 0$$

that gives

$$\mathsf{f}\prime_{\lambda^t} = -\left(\nabla^2_{\mathsf{f}}\, \mathcal{V}(\mathsf{f}_{\lambda^t}) + \lambda^t\, \nabla^2_{\mathsf{f}}\Omega\,(\mathsf{f}_{\lambda^t})\right)^{-1} \nabla_{\mathsf{f}}\Omega(\mathsf{f}_{\lambda^t})$$

The piecewise behavior is possible if $\mathsf{f}\prime_{\lambda^t}$ is constant. To fulfill this condition, it requires $\nabla^2_{\mathsf{f}}\Omega(\mathsf{f}_{\lambda^t}) = 0$ (independence with respect to λ) and $\nabla^2_{\mathsf{f}}\, \mathcal{V}(\mathsf{f}_{\lambda^t})$ to be constant. The latter condition is satisfied as the loss or the regularizer are assumed linear or quadratic. These requirements achieve the proof.

In fact, similar to SVM classification, it turns out that $\hat{\mathsf{f}}_\lambda$ as a function of λ is piecewise linear and hence forms a *regularization path*. Indeed, in the RankSVM algorithm, the loss function $\mathcal{V}$(f) is the hinge loss (which is a L_1 type-function) and the regularizer Ω(f) is chosen as a quadratic or L_1 function (see Figure 1). These choices therefore fulfill the requirements of the theorem.

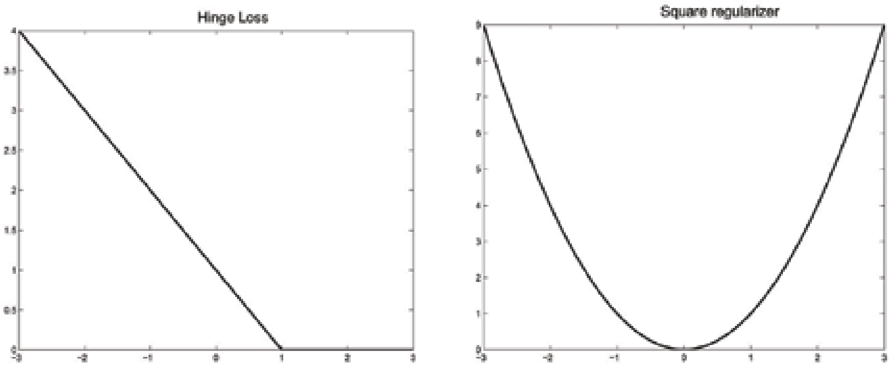


Fig. 1. Illustration of the typical choices of loss function and regularizer in SVM framework. Left) Hinge loss, Right) Square regularizer.

As in SVM classification, the breakpoints of this path correspond to certain events (described in more detail in Section 5). Points of the regularization path which are not breakpoints can not be distinguished in terms of margin-errors of the training data. To choose a particular regularization parameter, and hence a particular solution to the ranking problem, we evaluate $\hat{f}_\lambda$ on a validation set for each breakpoint of the regularization path.
Before delving into the details of solution path computation, the next two sections present the ranking SVM algorithm.

3. Ranking SVM

For clarity and simplification sakes, let consider the example of web pages search in ranking problems like (*i*) and (*ii*) from the introduction. To this purpose, we consider a set of query-document samples $x = (q, d) \in X$, together with a label y that induces a relative order or preference between the documents d accordingly to a query q. Each query induces a directed acyclic graph (X, E), with $E \subseteq X^2$ (See Figure 2).

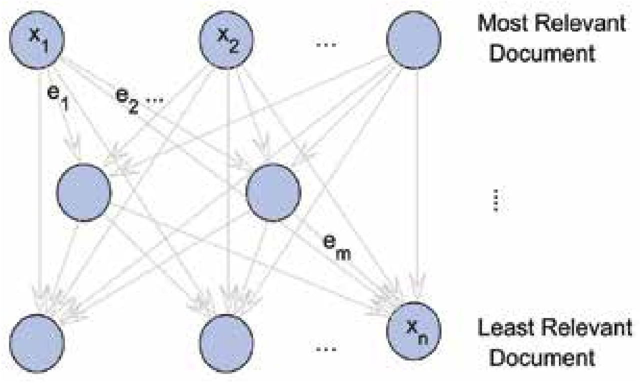


Fig. 2. Induced graph from ranking constraints for a particular query

For (*i*) the set of web pages forms the vertex set X of the digraph and we are also given some further information about the web pages (like a bag-of-words representation). For (*ii*) each vertex of the graph is a pair containing a query ($q \in \mathcal{Q}$) and a document ($d \in \mathcal{D}$). Hence, the vertex set is $X \neg \mathcal{Q} \times \mathcal{D}$ and edges of the form $((q, d), (q, d')) \in E$ with $d, d' \in \mathcal{D}$; $q \in \mathcal{Q}$ represent that d was more relevant than d' for an user asking query q. In addition one typically assumes some joint representations of queries and web pages.
The beauty of these problems is that classification and ordinal regression problems can be written as a ranking problem, therefore, the ranking SVM framework can be as well used for this kind of problems. The exact decision frontier can be calculated via a ROC curve, for example.
In both cases, ranking algorithms aim to find an ordering (permutation) of the vertex $\pi : X \to [\![n]\!]$ where $n = |X|$ and $[\![n]\!] = \{1, \ldots, n\}$ such that the more relevant a document is, the higher it is placed after the permutation, while as few as possible preferences are violated by the permutation.
Ranking SVM approaches such learning problems by solving the following primal optimization problem :

$$\begin{aligned} \hat{\mathrm{f}}_\lambda = \quad & \underset{\mathrm{f}\in\mathcal{H}}{\operatorname{argmin}} \quad \xi^\top \mathbb{1} + \lambda \|\mathrm{f}\|^2_{\mathcal{H}} \\ & \text{subject to :} \quad \mathrm{f}(v) - \mathrm{f}(u) \geq 1 - \xi_{vu} \quad \forall (u,v) \in E \\ & \qquad\qquad\quad\ \xi_{vu} \geq 0 \qquad\qquad\qquad \forall (u,v) \in E\,. \end{aligned} \tag{5}$$

Here, $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), $\lambda \in \mathbb{R}^+$ is a regularization parameter, and the square norm $\|\cdot\|^2_{\mathcal{H}}$ in the Hilbert space serves as the regularizer. As in SVM for classification, the slack variables ξ_{vu}, $(u, v) \in E$ traduce the cost related to the violation of the constraints (u, v). The final permutation π is then obtained by sorting X according to f and resolving ties randomly.

Now, to easy the notation, let $k : X \times X \to \mathbb{R}$ be the reproducing kernel of $\mathcal{H}$ and denote the vertex by $\mathbf{x}_i$ such that $X = \{\mathbf{x}_i \mid i \in [\![n]\!]\}$. The set of violated constraints is $\{(\mathbf{x}_i, \mathbf{x}_j) \in E \mid \pi(\mathbf{x}_i) < \pi(\mathbf{x}_j)\}$. The decision function will have the form $\hat{\mathrm{f}}_\lambda(\cdot) = \sum_{i=1}^n \beta_i k(\mathbf{x}_i, \cdot)$ with $\beta_i \in \mathbb{R}$. With slight abuse of notation we write $\mathbf{k}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), ..., k(\mathbf{x}, \mathbf{x}_n))^\mathrm{T}$. Using this notation, a ranking problem (5) with m preferences $E = \{(\mathbf{x}_{k_i}, \mathbf{x}_{l_i}) \mid i \in [\![m]\!]\}$ can be written as :

$$\begin{aligned} \hat{\boldsymbol{\beta}}(\lambda) = \quad & \underset{\boldsymbol{\beta}\in\mathbb{R}^n, \boldsymbol{\xi}\in\mathbb{R}^m}{\operatorname{argmin}} \quad \boldsymbol{\xi}^\top \mathbb{1} + \tfrac{\lambda}{2} \boldsymbol{\beta}^\top K \boldsymbol{\beta} \\ & \text{s. t.} \quad \boldsymbol{\beta}^\top (\mathbf{k}(\mathbf{x}_{k_i}) - \mathbf{k}(\mathbf{x}_{l_i})) \geq 1 - \xi_i \quad \forall i \in [\![m]\!] \\ & \qquad\ \xi_i \geq 0 \qquad\qquad\qquad\qquad\qquad\quad \forall i \in [\![m]\!] \end{aligned} \tag{6}$$

with $K = [K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)] \in \mathbb{R}^{n\times n}$ the Gram matrix and $\boldsymbol{\beta} = [\beta_1 \dots \beta_n]^\mathrm{T}$.

The complexity of the problem comes from the fact that the number of such preference constraints m is of order the square of the training set size that is $m = O(n^2)$. The Lagrangian L of problem (6) is given by :

$$\mathcal{L} = \boldsymbol{\xi}^\top \mathbb{1} + \frac{\lambda}{2} \boldsymbol{\beta}^\top K \boldsymbol{\beta} - \sum_{i=1}^m \alpha_i \left(\boldsymbol{\beta}^\top (\mathbf{k}(\mathbf{x}_{k_i}) - \mathbf{k}(\mathbf{x}_{l_i})) - 1 + \xi_i \right) - \sum_i^m \gamma_i \xi_i$$

with $\alpha_i \geq 0$, $\gamma_i \geq 0$. A matrix $P \in \mathbb{R}^{m\times n}$ can be defined with entries

$$P_{ij} = \begin{cases} +1 & \text{if } j = k_i \\ -1 & \text{if } j = l_i \\ 0 & \text{otherwise} \end{cases} \quad \Longrightarrow \quad PK = \begin{pmatrix} \mathbf{k}(\mathbf{x}_{k_1})^\top - \mathbf{k}(\mathbf{x}_{l_1})^\top \\ \mathbf{k}(\mathbf{x}_{k_2})^\top - \mathbf{k}(\mathbf{x}_{l_2})^\top \\ \vdots \\ \mathbf{k}(\mathbf{x}_{k_m})^\top - \mathbf{k}(\mathbf{x}_{l_m})^\top \end{pmatrix} \tag{7}$$

so that the Lagrangian can be expressed as :

$$\mathcal{L} = \boldsymbol{\xi}^\top \mathbb{1} + \frac{\lambda}{2} \boldsymbol{\beta}^\top K \boldsymbol{\beta} - \boldsymbol{\beta}^\top K P^\top \boldsymbol{\alpha} + \mathbb{1}^\top \boldsymbol{\alpha} - \boldsymbol{\xi}^\top \boldsymbol{\alpha} - \boldsymbol{\xi}^\top \boldsymbol{\gamma}$$

with $\boldsymbol{\alpha} \geq \mathbf{0}$, $\boldsymbol{\gamma} \geq \mathbf{0}$ (the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ contain respectively the Lagrange parameters α_i and γ_i). Using the Karush-Kuhn-Tucker (KKT) conditions, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \xi} = 0 \Rightarrow 0 = \mathbb{1} - \boldsymbol{\alpha} - \gamma \quad \text{together with} \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = 0 \Rightarrow 0 = \lambda K \boldsymbol{\beta} - K P^\top \boldsymbol{\alpha}.$$

These equations result in conditions, $0 \leq \alpha_i \leq 1$ and $\boldsymbol{\beta} = \frac{1}{\lambda} P^\top \boldsymbol{\alpha}$, so that

$$\hat{\mathrm{f}}_\lambda(\mathbf{x}) = \frac{1}{\lambda} \boldsymbol{\alpha}^\top P \mathbf{k}(\mathbf{x}).$$

Finally, the dual of Problem (6) is:

$$\begin{array}{rcll} \hat{\boldsymbol{\alpha}}(\lambda) & = & \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmax}} & \boldsymbol{\alpha}^\top \mathbb{1} - \frac{1}{2\lambda} \boldsymbol{\alpha}^\top P K P^\top \boldsymbol{\alpha} \\ & & \text{s.t.} & 0 \leq \boldsymbol{\alpha} \leq \mathbb{1}. \end{array} \tag{8}$$

4. RankSVM singularity

As mentioned in the introduction, the Ranking SVM optimization problem induces a directed graph for each query. This structure constraints an edge for each relationship of relevance between samples that has to be satisfied. These constraints include as well all transitive relationships that could in fact be induced by other ones. This redundancy in the constraints setting cause the Hessian matrix in Problem (8) to be singular.
This issue can be overcome by designing for each query a sample as the maximum of all his rank for this query, so that edges from the chosen sample will be added to the other samples. For the immediate upper level, all samples in it will be joint to the maximum of the previous rank and so on. The obtained graph would look as in Figure (3).

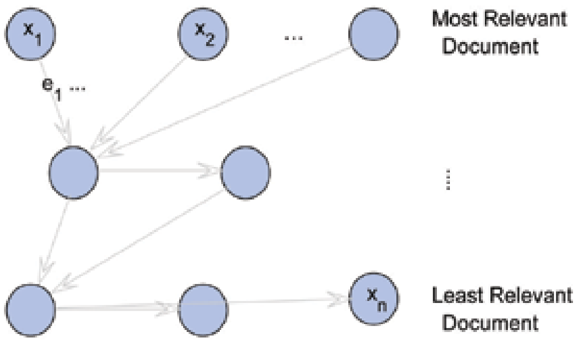


Fig. 3. New graph that will generate a non singular Hessian on the dual problem

The advantage of this new formulation is that the number of constraints is significantly smaller than in the original RankSVMalgorithm. The first one can be of order $O(n^2)$, while the second one is of order $O(n)$ This will lead to a smaller problem and faster training time with a consistent problem equivalence.

5. Regularization path for ranking SVM

Following the arguments developed in Rosset and Zhu (2007), it can be shown that the solution $\hat{\boldsymbol{\alpha}}(\lambda)$ of the above dual problem is a piecewise linear function of λ. Hence the

problem admits a piecewise linear regularization path. A regularization path has breakpoints $\lambda^1 > \lambda^2 > \ldots$ such that for an interval $[\lambda^{t+1}, \lambda^t]$ (i.e., with no breakpoint) the optimal solutions $\hat{\boldsymbol{\alpha}}(\lambda)$ and $\hat{\boldsymbol{\beta}}(\lambda)$ can easily be obtained for all $\lambda \in [\lambda^{t+1}, \lambda^t]$.
Following the work of Hastie et al. (2004) we now derive the regularization path of ranking SVM. For given λ, and to simplify the notations, let f(**x**) and $\boldsymbol{\alpha}$ be the decision function and the optimal solution for Problems (6) and (8), respectively (i.e. $\hat{\mathsf{f}}_\lambda(\mathbf{x}) \equiv \mathsf{f}(\mathbf{x})$ and $\hat{\boldsymbol{\alpha}}(\lambda) \equiv \boldsymbol{\alpha}$). Then, the following partition derived from the KKT optimality conditions can be made :

$$\begin{aligned}
\mathcal{I}_\alpha &= \{i \in [\![m]\!] \mid \mathsf{f}(\mathbf{x}_{k_i}) - \mathsf{f}(\mathbf{x}_{l_i}) = 1\} = \{i \in [\![m]\!] \mid 0 < \alpha_i < 1\},\\
\mathcal{I}_0 &= \{i \in [\![m]\!] \mid \mathsf{f}(\mathbf{x}_{k_i}) - \mathsf{f}(\mathbf{x}_{l_i}) > 1\} = \{i \in [\![m]\!] \mid \alpha_i = 0\}, \quad \text{and}\\
\mathcal{I}_1 &= \{(i \in [\![m]\!] \mid \mathsf{f}(\mathbf{x}_{k_i}) - \mathsf{f}(\mathbf{x}_{l_i}) < 1\} = \{i \in [\![m]\!] \mid \alpha_i = 1\}.
\end{aligned}$$

The set $\mathcal{I}_0$ represents the satisfied constraints whereas $\mathcal{I}_1$ is devoted to the violated constraints and $\mathcal{I}_\alpha$ includes the "margin constraints".
Similarly, we will denote by $\boldsymbol{\alpha}^t$ and f ^{t}(**x**) the optimal solution of the dual Problem (6) for the regularization parameter λ^t. Note that we assume the above sets $(\mathcal{I}_\alpha^t, \mathcal{I}_1^t, \mathcal{I}_0^t)$ induced by the solution of the optimization problem for λ^t remain unchanged for $\lambda \in [\lambda^{t+1}, \lambda^t]$, i.e., $(\mathcal{I}_\alpha^t, \mathcal{I}_1^t, \mathcal{I}_0^t) = (\mathcal{I}_\alpha, \mathcal{I}_1, \mathcal{I}_0)$. Hence, $\alpha_i \in \mathcal{I}_\alpha$ depends linearly on λ, This can be seen by writing f(**x**) as follows :

$$\begin{aligned}
\mathsf{f}(\mathbf{x}) &= \left[\mathsf{f}(\mathbf{x}) - \frac{\lambda^t}{\lambda}\mathsf{f}^t(\mathbf{x})\right] + \frac{\lambda^t}{\lambda}\mathsf{f}^t(\mathbf{x})\\
&= \frac{1}{\lambda}\left[(\boldsymbol{\alpha} - \boldsymbol{\alpha}^t)^\top P\mathbf{k}(\mathbf{x}) + \lambda^t \mathsf{f}^t(\mathbf{x})\right] \qquad (9)\\
\mathsf{f}(\mathbf{x}) &= \frac{1}{\lambda}\left[(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha})^\top P_{\mathcal{I}_\alpha}\mathbf{k}(\mathbf{x}) + \lambda^t \mathsf{f}^t(\mathbf{x})\right]
\end{aligned}$$

where the last line is true as $\alpha_i - \alpha_i^t$ for all $i \notin \mathcal{I}_\alpha$. $P_{\mathcal{I}_\alpha}$ is the submatrix of P containing the rows corresponding to $\mathcal{I}_\alpha$ and all columns. For all $i \in \mathcal{I}_\alpha$ we have that $1 = \mathsf{f}(\mathbf{x}_{k_i}) - \mathsf{f}(\mathbf{x}_{l_i}) = \mathsf{f}^t(\mathbf{x}_{k_i}) - \mathsf{f}^t(\mathbf{x}_{l_i})$, leading to

$$1 = \frac{1}{\lambda}\left[(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha})^\top P_{\mathcal{I}_\alpha}(\mathbf{k}(\mathbf{x}_{k_i}) - \mathbf{k}(\mathbf{x}_{l_i})) + \lambda^t\right].$$

Therefore

$$\lambda - \lambda^t = (\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha})^\top P_{\mathcal{I}_\alpha}(\mathbf{k}(\mathbf{x}_{k_i}) - \mathbf{k}(\mathbf{x}_{l_i})) \ . \qquad (10)$$

This equation is valid for all pairs in $\mathcal{I}_\alpha$ for fixed sets $\mathcal{I}_\alpha, \mathcal{I}_0$, and $\mathcal{I}_1$. It can be simplified by transposing Eq. (10) and using Eq. (7) in it, getting :

$$(\lambda - \lambda^t)\mathbb{1}_{\mathcal{I}_\alpha} = P_{\mathcal{I}_\alpha} K P^\top_{\mathcal{I}_\alpha}\left(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha}\right) \qquad (11)$$

If we define $\eta = (P_{\mathcal{I}_\alpha} K P_{\mathcal{I}_\alpha}^\top)^{-1} \mathbb{1}_{\mathcal{I}_\alpha}$, with $\mathbb{1}_{\mathcal{I}_\alpha}$ a vector of ones of size $|\mathcal{I}_\alpha|$, then it can finally be seen that α_i, $i \in \mathcal{I}_\alpha$ changes piecewise linearly in λ as follows :

$$\alpha_i = \alpha_i^t - (\lambda^t - \lambda)\eta_i \qquad i \in \mathcal{I}_\alpha. \tag{12}$$

For all $\lambda \in [\lambda^{t+1}, \lambda^t]$, the optimal solution $\boldsymbol{\alpha}$ (and consequently the decision function f($\mathbf{x}$)) can be easily obtained until the sets change, i.e., an event occurs. From any optimal solution $\boldsymbol{\alpha}$ for λ, the corresponding sets $\mathcal{I}_\alpha$, $\mathcal{I}_0$, and $\mathcal{I}_1$ can be deduced and thereon the successive solutions for different λ.

5.1 Initialization

If λ is very large, $\boldsymbol{\beta} = \mathbf{0}$ minimizes Problem (6). This implies that $\xi_i = 1$, $\forall i$ and because of the strict complementary and KKT conditions, $\gamma_i = 0 \Rightarrow \alpha_i = 1$. To have at least one element in $\mathcal{I}_\alpha$, we need a pair $(\mathbf{x}_{k_i}, x_{l_i})$ that verifies $\boldsymbol{\beta}^\top(\mathbf{k}(x_{k_i}) - \mathbf{k}(x_{l_i}))$ =1. We know that $\boldsymbol{\beta} = \frac{1}{\lambda} P^\top \boldsymbol{\alpha}$ and therefore $\boldsymbol{\alpha}$ = 1I solves , for all pairs, the equation

$$\lambda_{k_i, l_i} = \boldsymbol{\alpha}^\top P(\mathbf{k}(x_{k_i}) - \mathbf{k}(x_{k_i})).$$

Hence, initially all pairs will be in $\mathcal{I}_1$ and, as initial λ value, we take

$$\lambda^0 = \max\{\lambda_{k_i, l_i}\}.$$

The set $\mathcal{I}_\alpha$ will contain the pairs which maximize the value of λ_0.

5.2 Event detection

At step t the optimal solution $\boldsymbol{\alpha}^t$ defines a partition $\mathcal{I}_\alpha$, $\mathcal{I}_1$, $\mathcal{I}_0$. If these sets remain fixed for all λ in a given range then the optimal solution $\boldsymbol{\alpha}(\lambda)$ is a linear function of λ. If an event occurs, i.e., the sets change, then the linear equation has to be readjusted. Two types of events have to be determined:

- a pair in $\mathcal{I}_\alpha$ goes to $\mathcal{I}_1$ or $\mathcal{I}_0$
- a pair in $\mathcal{I}_1$ or $\mathcal{I}_0$ goes to $\mathcal{I}_\alpha$.

5.2.1 Pair in $\mathcal{I}_\alpha$ goes to $\mathcal{I}_1$ or $\mathcal{I}_0$

This event can be determined by analyzing at which value of λ the corresponding α_i turns zero or one. Eq. (12) is used and the following systems are solved for λ_i :

$$1 = \alpha_i^t - (\lambda^t - \lambda_i)\eta_i \qquad i \in \mathcal{I}_\alpha \tag{13}$$

$$0 = \alpha_i^t - (\lambda^t - \lambda_i)\eta_i \qquad i \in \mathcal{I}_\alpha. \tag{14}$$

Using these last equations, the exact values for λ_i that produces an event on pairs in $\mathcal{I}_\alpha$ moving to $\mathcal{I}_0 \cup \mathcal{I}_1$ can be determined.

5.2.2 Pair in $\mathcal{I}_1$ or $\mathcal{I}_0$ goes to $\mathcal{I}_\alpha$

To detect this event, note that Equation (11) can also be written as follows :

$$\begin{aligned} \left(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha}\right) &= (\lambda - \lambda^t)\left[\left(P_{\mathcal{I}_\alpha} K P^\top_{\mathcal{I}_\alpha}\right)^{-1} \mathbf{1}_{\mathcal{I}_\alpha}\right] \\ \left(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha}\right) &= (\lambda - \lambda^t)\boldsymbol{\eta}^\top . \end{aligned} \tag{15}$$

Plugging Eq. (15) in Eq. (9), we can write f($\mathbf{x}$) in a convenient manner:

$$\mathsf{f}(\mathbf{x}) = \frac{1}{\lambda}\left[\lambda^t \mathsf{f}^t(\mathbf{x}) + (\lambda - \lambda^t)\boldsymbol{\eta}^\top P_{\mathcal{I}_\alpha}\mathbf{k}(\mathbf{x})\right].$$

If we let $\mathsf{h}^t(\mathbf{x}) = \boldsymbol{\eta}^\top_{\mathcal{I}^t_\alpha} P_{\mathcal{I}_\alpha}\mathbf{k}(\mathbf{x})$, then

$$\mathsf{f}(\mathbf{x}) = \frac{1}{\lambda}\left[\lambda^t \mathsf{f}^t(\mathbf{x}) - \lambda^t \mathsf{h}^t(\mathbf{x}) + \lambda \mathsf{h}^t(\mathbf{x})\right] \tag{16}$$

An event on pair $(k_i, l_i) \in \mathcal{I}_0 \cup \mathcal{I}_1 \Vdash \mathcal{I}_\alpha$ means that $\mathsf{f}(\mathbf{x}_{k_i}) - \mathsf{f}(\mathbf{x}_{l_i}) = 1$ and can be detected by using Equation (16). The corresponding λ_i that generates this event is calculated as follows:

$$\lambda_i = \frac{\lambda^t\left[\left(\mathsf{f}^t(\mathbf{x}_{k_i}) - \mathsf{f}^t(\mathbf{x}_{l_i})\right) - \left(\mathsf{h}^t(\mathbf{x}_{k_i}) - \mathsf{h}^t(\mathbf{x}_{l_i})\right)\right]}{1 - \left(\mathsf{h}^t(\mathbf{x}_{k_i}) - \mathsf{h}^t(\mathbf{x}_{l_i})\right)} \tag{17}$$

Finally, λ^{t+1} will be the largest resulting $\lambda_i < \lambda^t$ from Equations (13), (14) and (17). In a cross validation framework, model selection can be done by learning the parameters in the training sets, an estimation of the generalization error (or ranking accuracy) can be taken by applying each model to the validation set.
The path computation is summarized by the pseudo-code of Algorithm 1.

5.3 Remarks and comments

Here we discuss briefly some issues of the algorithm related to the piecewise variation, the numerical complexity and how to address the emptiness of the set $\mathcal{I}_\alpha$.

On the functional piecewise variation

Let the function g = λf corresponding to the regularization parameter λ. In a similar manner, consider the function $g^t = \lambda^t f^t$ which corresponds to the solution for the value λ^t. From Eq. (16), one derives easily the relation $g = g^t + (\lambda - \lambda^t)h^t$. Therefore, we recover the piecewise linear variation stated in theorem 1. This linear variation formally concerns the function g instead of f. However the parameters $\boldsymbol{\alpha}$ involved in f evolves linearly with λ.

On the numerical complexity

The numerical complexity of the algorithm can be analyzed as follows. We assume the whole matrix $P\,K\,P^\mathrm{T}$ is available beforehand as it can be built and stored at the beginning of

Algorithm 1 Pseudo-code of the RankingSVM path computation

```
Require: The set of $m$ preferences $E = \{(\mathbf{x}_{k_i}, \mathbf{x}_{l_i}) \mid i \in [\![m]\!]\}$
Ensure: All solutions $(\mathrm{f}_\lambda, \lambda)$
  Set $t = 0$
  Compute the initial value $\lambda^0$ and estimation $\boldsymbol{\alpha}^0$.
  Deduce the corresponding sets $(\mathcal{I}_\alpha^0, \mathcal{I}_1^0, \mathcal{I}_0^0)$
  repeat
    Compute the update direction of the parameters $\boldsymbol{\eta}^t = (P_{\mathcal{I}_\alpha^t} K P_{\mathcal{I}_\alpha^t}^\top)^{-1} \mathbb{1}_{\mathcal{I}_\alpha^t}$
    Use $\boldsymbol{\eta}^t$ to detect all the necessary events and the corresponding regularization parameter values
    Select the appropriate boundary value $\lambda^{t+1}$, save the event type (pair in $\mathcal{I}_1 \cup \mathcal{I}_0$ goes to $\mathcal{I}_\alpha$ or pair in $\mathcal{I}_\alpha$ goes to $\mathcal{I}_1$ or $\mathcal{I}_0$) and the related pair(s) of constraint
    Update the parameters according to Eq. (12)
    Update the sets $(\mathcal{I}_\alpha^t, \mathcal{I}_1^t, \mathcal{I}_0^t)$ according to the retained event and the concerned pair(s)
    $t = t + 1$
  until a termination criterion is reached
```

the algorithm and this computation requires $\mathcal{O}(mn^2)$ operations from the knowledge of the matrices P and K. At each iteration, solving the linear system (11) involves a cost of order $\mathcal{O}(|I_\alpha|^3)$. The calculation of all next values λ^{t+1} (using Eq. 13-14 and 17) has a numerical complexity of $\mathcal{O}(m|I_\alpha|)$ whereas the detection of the next event is of order $\mathcal{O}(m)$. Let $y_i = \mathrm{f}(\mathbf{x}_{k_i}) - \mathrm{f}(\mathbf{x}_{l_i})$ the evaluation of the preference $(\mathbf{x}_{k_i}, x_{l_i})$, $i \in [\![m]\!]$. According to (16), the update of all y_i is $\mathcal{O}(m)$. We can note that the computational complexity is essentially related to the cardinality of $I_\alpha|$. The cubic complexity of the linear system can be decreased to square complexity using a Sherman-Morrison rule to update the inverse of the matrix $P_{\mathcal{I}_\alpha} K P_{\mathcal{I}_\alpha}^\top$ or a Choleski update procedure. The exact complexity of the algorithm is hard to predict since the total number of events needed for exploring entirely the regularization path is data-dependent and the mean size of $|\mathcal{I}_\alpha|$ is difficult to guess beforehand. However, the total complexity is few multiples of the cost for solving directly the dual problem (8).

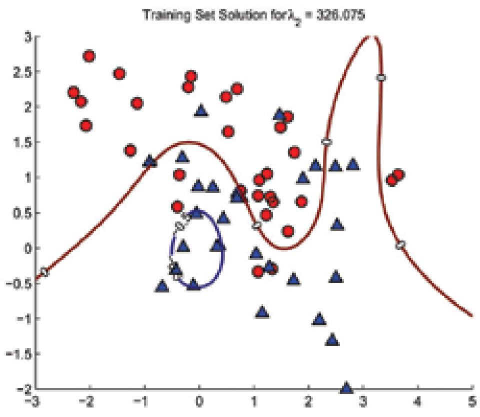
On the emptiness of $\mathcal{I}_\alpha$

It may happen during the algorithm that the set $\mathcal{I}_\alpha$ becomes empty. In such situation, a new initialization of the algorithm is needed. We apply the procedure developed in Subsection 5.1 except the fact we consider solely the pairs in $\mathcal{I}_1$ keeping unchanged the set $\mathcal{I}_0$.

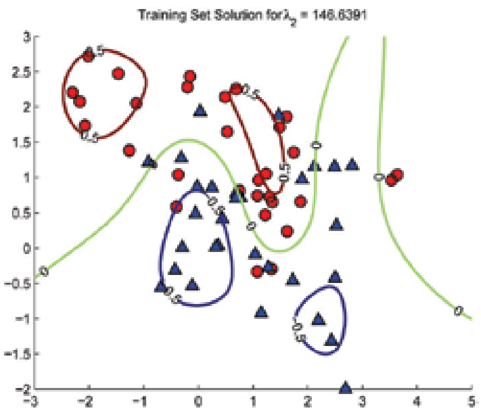
6. Experimental results

Several datasets where used to measure the accuracy and time to process the regularization path for the RankSVM algorithm. Firstly, a toy example generated from Gaussian distributions (Hastie et al. (2001)) was applied. Some invetisgations on real life datasets taken from the UCI repository[1] are further presented.

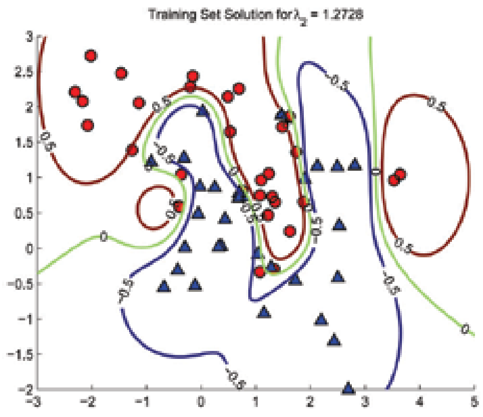
The mixtures dataset of Hastie et al. (2004) was originally designed for binary classification with instances $\mathbf{x}_i$ and corresponding labels $y_i \in \{\pm 1\}$. However, it can be viewed as a ranking problem with $E = \{(\mathbf{x}_i, \mathbf{x}_j) \mid y_i > y_j\}$. It contains 100 positive and 100 negative points which would induce 10000 constraints. The regularization path was run on this dataset and a decision function was taken on zero. This decision boundary can still be improved by observing the generated ROC curve at each level. Figure (4) illustrates the decision function



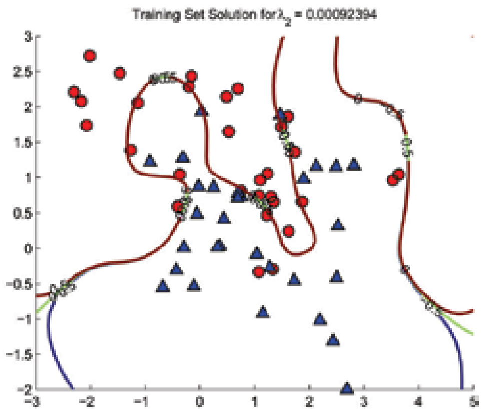
(a) Initialization



(b) Solution after some iterations



(c) Solution after more iterations



(d) Solution for the smallest λ

Fig. 4. Illustration of the regularization path for the mixture dataset, all red points must be ranked higher than the blue points. As λ decreases, the margin gets smaller and the distance between pairs tends to be larger than one.

[1] http ://archive.ics.uci.edu/ml/datasets.html

for different breakpoints of the regularization path. The initial solution (a) is poor but after some iterations the results are improved as shown in subfigure (b). The most interesting solution is illustrated on subfigure (c) where almost constraints are satisfied.
The others datasets are regression problems and can also be viewed as ranking problems by letting $E = \{(\mathbf{x}_i, \mathbf{x}_j) \mid y_i > y_j\}$.
The number of induced constraints on the complete dataset and those obtained after following the graph design in Figure (3) are compared in Table 1.
For the experiments, a training, a validation and a test sets where built, being the last two of about half the size of the training set each. The number of involved features, training and test instances, and training and test constraints are summarized in Table 2.
Finally, the experiment was run 10 times, the error is measured as the percentage of misclassified samples. The size of A tells the number of support vectors and finally the time, is the average time (in seconds) to train a regularization path. The results are gathered in Table 3. We can see that the computation cost needed to obtain all possible soultions and their evaluation on test samples (in order to pick up the best one) is fairly cheaper making the approach particularly interesting.

7. Conclusions

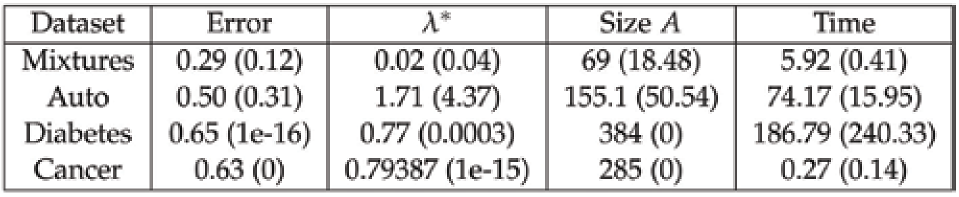
Regularization parameter search for the ranking SVM can be efficiently done by calculating the regularization path. This approach calculates efficiently the optimal solution for all possible regularization parameters by solving (in practice) small linear problems. This approach has the advantage of overcoming local minimum of the regularization function. These advantages make the parameter selection considerably less time consuming and the obtained optimal solution for each model more robust.

Dataset	RanksSVM	Reduced RankSVM
Mixtures	10000	199
Auto	75245	656
Diabetes	134000	767
Cancer	75684	568

Table 1. Number of training instances under the original RankSVM and the ones obtained after the graph reformulation

Dataset	Features	Tr. Instances	Tr. Constraints	Test Instances	Test Constraints
Mixtures	2	99	100	50	50
Auto	7	305	196	98	98
Diabetes	8	383	384	192	192
Cancer	30	284	285	142	142

Table 2. Summary of the features of the training, validation and test sets



Dataset	Error	λ^*	Size A	Time
Mixtures	0.29 (0.12)	0.02 (0.04)	69 (18.48)	5.92 (0.41)
Auto	0.50 (0.31)	1.71 (4.37)	155.1 (50.54)	74.17 (15.95)
Diabetes	0.65 (1e-16)	0.77 (0.0003)	384 (0)	186.79 (240.33)
Cancer	0.63 (0)	0.79387 (1e-15)	285 (0)	0.27 (0.14)

Table 3. Obtained results by running the regularization path on the datasets described in Table 1. The results are averaged over 10 trials.

The numerical complexity of the algorithm depends on the number of iterations needed to explore the overall solution path and the mean size of $\mathcal{I}_\alpha$. At each iteration, a linear system is solved to get $\boldsymbol{\eta}$ which has complexity $\mathcal{O}(|\mathcal{I}_\alpha|^2)$. Empirically we observed that the number of iterations is typically only 2-3 times larger than the number of training pairs

Another key point is the determination of kernel hyper-parameter. This problem was not tackled here. However, one can seek to combine our regularisation path with the kernel parameter path developed in Gang Wang and Lochovsky (2007).

8. References

Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152.

Cortes, C., Mohri, M., and Rastogi, A. (2007). An alternative ranking problem for search engines. In Demetrescu, C., editor, *WEA*, volume 4525 of *Lecture Notes in Computer Science*, pages 1–22. Springer.

Crammer, K. and Singer, Y. (2001). Pranking with ranking.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4 :933–969.

Gang Wang, D.-Y. Y. and Lochovsky, F. H. (2007). A kernel path algorithm for support vector machines. In *Proceedings of ICML'2007.*

Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. (2004). The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5 :1391–1415.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning : Data Mining, Inference and Prediction*. Springer Verlag, New York.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In Smola, A., Bartlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 115–132, Cambridge, MA. MIT Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133– 142.

Markowitz, H. M. (1959). *Portfolio selection : Efficient diversification of investments*. John Wiley and Sons, Inc.

Rosset, S. and Zhu, J. (2007). Piecewise linear regularized solution paths. *Annals of Statistics*, 35(3) :1012–1030.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. MIT Press.

Generation of Facial Expression Map using Supervised and Unsupervised Learning

Masaki Ishii[1], Kazuhito Sato[1], Hirokazu Madokoro[1] and Makoto Nishida[2]
[1]Akita Prefectural University,
[2]Akita University
Japan

1. Introduction

Recently, studies of human face recognition have been conducted vigorously (Fasel & Luettin, 2003; Yang et al., 2002; Pantic & Rothkrantz, 2000a; Zhao et al., 2000; Hasegawa et al., 1997; Akamatsu, 1997). Such studies are aimed at the implementation of an intelligent man-machine interface. Especially, studies of facial expression recognition for human-machine emotional communication are attracting attention (Fasel & Luettin, 2003; Pantic & Rothkrantz, 2000a; Tian et al., 2001; Pantic & Rothkrantz, 2000b; Lyons et al., 1999; Lyons et al., 1998; Zhang et al., 1998).

The shape (static diversity) and motion (dynamic diversity) of facial components such as the eyebrows, eyes, nose, and mouth manifest expressions. Considering facial expressions from the perspective of static diversity because facial configurations differ among people, it is presumed that a facial expression pattern appearing on a face when facial expression is manifested includes person-specific features. In addition, from the viewpoint of dynamic diversity, because the dynamic change of facial expression originates in a person-specific facial expression pattern, it is presumed that the displacement vector of facial components has person-specific features. The properties of the human face described above reveal the following tasks.

The first task is to generalize a facial expression recognition model. Numerous conventional approaches have attempted generalization of a facial expression recognition model. They use the distance of motion of feature points set on a face and the motion vectors of facial muscle movements in its arbitrary regions as feature values. Typically, such methods assign that information to so-called Action Units (AUs) of a Facial Action Coding System (FACS) (Ekman & Friesen, 1978). In fact, AUs are described qualitatively. Therefore, no objective criteria pertain to the setting positions of feature points and regions. They all depend on a particular researcher's experience. However, features representing facial expressions are presumed to differ among subjects. Accordingly, a huge effort is necessary to link quantitative features with qualitative AUs for each subject and to derive universal features therefrom. It is also suspected that a generalized facial expression recognition model that is applicable to all subjects would disregard person-specific features of facial expressions that are borne originally by each subject. For all the reasons described above, it is an important task to establish a method to extract person-specific features using a common approach to every subject, and to build a facial expression recognition model that incorporates these features.

The second task is to verify the validity of categorizing emotions into six basic emotions: anger, sadness, disgust, happiness, surprise, and fear. In general, facial expressions rarely appear as a pure and solitary basic emotion, but they often appear as a mixture of various emotions. Moreover, the variety of motions of facial parts and forms is not unique; motions are diverse patterns of facial expression. Facial expressions are presumed to be classifiable into categories whose number is determined as optimal for each subject. Consequently, the categorization of facial expressions is attributed to a problem of classification into an unknown number of categories. Accordingly, it is necessary to establish a method for determining the optimal number of categories for each subject.

An ideal facial expression recognition system is expected to be capable of categorizing facial expressions into as many types as possible. For that purpose, it is desirable that a facial expression pattern be categorized with its operator's subjectivity excluded, and that the operator be able to attribute emotions uniquely to the categories. That is, because an emotion in one universal category might yield different patterns of facial expression in each subject, a system is expected to be capable of varying criteria for facial expression categorization according to the subjective interpretation of an operator.

For this chapter, we assume categorization of facial expression as a classification problem into an unknown number of categories. We propose a generation method of a person-specific Facial Expression Map (FEMap) using the Self-Organizing Maps (SOM) (Kohonen, 1995) of unsupervised learning and Counter Propagation Networks (CPN) (Nielsen, 1987) of supervised learning together. The proposed method consists of an extraction phase of person-specific facial expression categories using a SOM and a generation phase of an FEMap using a CPN. During the first phase, we particularly examine the unsupervised learning function and data compression function using the SOM of a narrow mapping space. The topological change of a face pattern in the expressional process of facial expression is learned hierarchically using the SOM of a narrow mapping space. The number of person-specific facial expression categories is generated along with the representative images of each category. Next, psychological significance based on a neutral expression and those of six basic emotions (anger, sadness, disgust, happiness, surprise, and fear) is assigned to each category. In the latter phase, we specifically address the supervised learning function and data extension function using the CPN of a large mapping space. The categories and the representative images described above are learned using the CPN of a large mapping space; a category map that expresses the topological characteristics of facial expression is generated. This study defines this category map as an FEMap. Experimental results for six subjects illustrate that the proposed method can generate a person-specific FEMap based on topological characteristics of facial expression appearing on face images.

2. Algorithms of SOM and CPN

2.1 Self-Organizing Maps (SOM)

The SOM is a learning algorithm that models the self-organizing and adaptive learning capabilities of a human brain (Kohonen, 1995). A SOM comprises two layers: an input layer, to which training data are supplied; and a Kohonen layer, in which self-mapping is performed by competitive learning. The learning algorithm of a SOM is described below.

1. Let $w_{i,j}(t)$ be a weight from an input layer unit i to a Kohonen layer unit j at time t. Actually, $w_{i,j}$ is initialized using random numbers.

2. Let $x_i(t)$ be input data to the input layer unit i at time t; calculate the Euclidean distance d_j between $x_i(t)$ and $w_{i,j}(t)$ using (1).

$$d_j = \sqrt{\sum_{i=1}^{I} \left(x_i(t) - w_{i,j}(t)\right)^2} \tag{1}$$

3. Search for a Kohonen layer unit to minimize d_j, which is designated as a winner unit.
4. Update the weight $w_{i,j}(t)$ of a Kohonen layer unit contained in the neighborhood region of the winner unit $N_c(t)$ using (2), where $a(t)$ is a learning coefficient.

$$w_{i,j}(t+1) = w_{i,j}(t) + \alpha(t)\left(x_i(t) - w_{i,j}(t)\right) \tag{2}$$

5. Repeat processes 2)–4) up to the maximum iteration of learning.

2.2 Counter Propagation Networks (CPN)

The CPN is a learning algorithm that combines the Grossberg learning rule with the SOM (Nielsen, 1987). A CPN comprises three layers: an input layer to which training data are supplied; a Kohonen layer in which self-mapping is performed by competitive learning; and a Grossberg layer, which labels the Kohonen layer by the counter propagation of teaching signals. A CPN is useful for automatically determining the label of a Kohonen layer when a category in which training data will belong is predetermined. This labeled Kohonen layer is designated as a category map. The learning algorithm of a CPN is described below.

1. Let $w^i_{n,m}(t)$ and $w^j_{n,m}(t)$ respectively indicate weights to a Kohonen layer unit (n, m) at time t from an input layer unit i and from a Grossberg layer unit j. In fact, $w^i_{n,m}$ and $w^j_{n,m}$ are initialized using random numbers.
2. Let $x_i(t)$ be input data to the input layer unit i at time t, and calculate the Euclidean distance $d_{n,m}$ between $x_i(t)$ and $w^i_{n,m}(t)$ using (3).

$$d_{n,m} = \sqrt{\sum_{i=1}^{I} \left(x_i(t) - w^i_{n,m}(t)\right)^2} \tag{3}$$

3. Search for a Kohonen layer unit to minimize $d_{n,m}$, which is designated as a winner unit.
4. Update weights $w^i_{n,m}(t)$ and $w^j_{n,m}(t)$ of a Kohonen layer unit contained in the neighborhood region of the winner unit $N_c(t)$ using (4) and (5), where $a(t)$, $\beta(t)$ are learning coefficients, and $t_j(t)$ is a teaching signal to the Grossberg layer unit j.

$$w^i_{n,m}(t+1) = w^i_{n,m}(t) + \alpha(t)\left(x_i(t) - w^i_{n,m}(t)\right) \tag{4}$$

$$w^j_{n,m}(t+1) = w^j_{n,m}(t) + \beta(t)\left(t_j(t) - w^j_{n,m}(t)\right) \tag{5}$$

5. Repeat processes 2)–4) up to the maximum iteration of learning.
6. After learning is completed, compare weights $w^j_{n,m}$ observed from each unit of the Kohonen layer; and let the teaching signal of the Grossberg layer with the maximum value be the label of the unit.

3. Proposed method

Figure 1 depicts the procedure used for the proposed method. The proposed method consists of two steps: extraction of person-specific facial expression categories using a SOM and generation of FEMap using a CPN. The proposed method is explained in detail below.

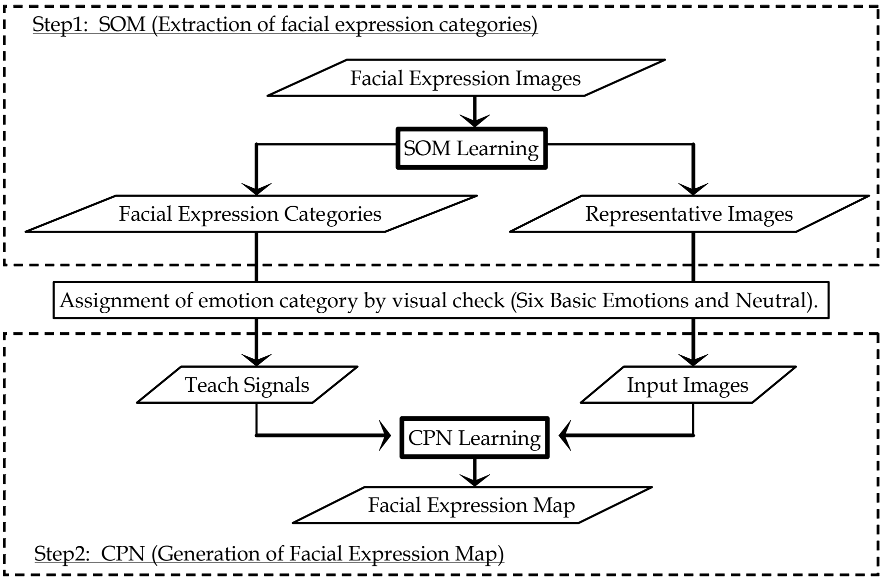

Step1: SOM (Extraction of facial expression categories)
Facial Expression Images
SOM Learning
Facial Expression Categories
Representative Images
Assignment of emotion category by visual check (Six Basic Emotions and Neutral).
Teach Signals
Input Images
CPN Learning
Facial Expression Map
Step2: CPN (Generation of Facial Expression Map)

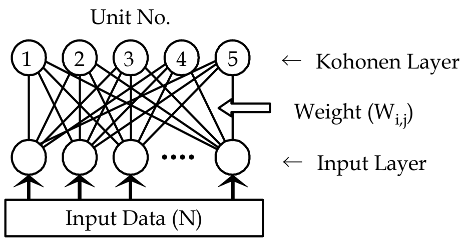
Fig. 1. Flow chart of proposal method.

3.1 Extraction of person-specific facial expression categories with SOM

The proposed method was used in an attempt to extract a person-specific facial expression category hierarchically using a SOM with a narrow mapping space. A SOM is an unsupervised learning algorithm; it classifies given facial expression images in a self-organized manner based on their topological characteristics. For that reason, it is suitable for classification problems with an unknown number of categories. Moreover, a SOM compresses the topological information of facial expression images using a narrow mapping space and performs classification based on features that roughly divide the training data. We speculate that repeating these hierarchically renders the classified amount of change of facial expression patterns comparable; thereby, a person-specific facial expression category can be extracted. Figure 2 depicts the extraction procedure of a facial expression category. Details of the process are explained below.

1. Expression images described in Section 4 were used as training data. The following processing was performed for each facial expression. The number of training data is assumed as *N* frames.
2. The facial expression topological characteristics of the training data were learned using the 1-D SOM of the Kohonen layer consisting of five units (Fig. 2(a)). The brightness value of images was used as input data because the brightness distribution represents the topological structure of the facial expression. The unit number of the input layer corresponds to the input image size.

3. The weight of the Kohonen layer $W_{i,j}(0 \leq W_{i,j} \leq 1)$ was converted to a value of 0–255 after the end of learning; visualized images were generated (Fig. 2(b)), where $n_1 - n_5$ are the numbers of training data classified into each unit.



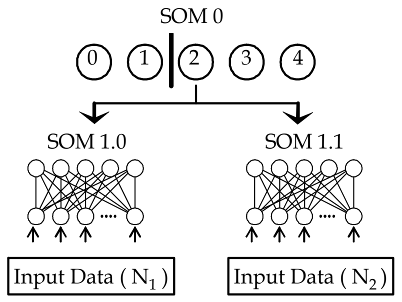
(a) Structure of SOM.

Unit No.	1	2	3	4	5
Visualized Image ($W_{i,j}$)					
Classification Result	n_1	n_2	n_3	n_4	n_5
Correlation Coefficient	0.9853	**<u>0.9786</u>**	0.9794	0.9866	
New Training Data	N_1		N_2		

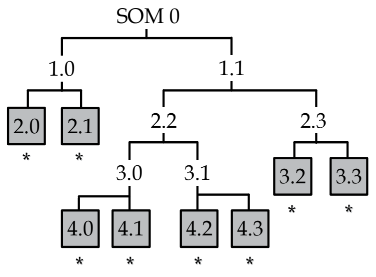
* $N = n_1 + n_2 + n_3 + n_4 + n_5$

* $N_1 = n_1 + n_2$, $N_2 = n_3 + n_4 + n_5$

(b) Learning with SOM and setup of new training data.



(c) Hierarchical learning with SOM.



Extracted categories	8 (*)
Representative Images	40 (5 units × 8)

(d) Generation of binary-tree structure.

Fig. 2. Extraction procedure of facial expression category.

4. Five visualized images can be considered as representative vectors of the training data classified into each unit ($n_1 - n_5$). Therefore, the images of five units were verified visually. All images were regarded as belonging to one category; processing was terminated if they were considered to represent the same facial expression. Subsequent processing was continued if multiple facial expressions were found to be mixed in the visualized images.
5. The correlation coefficient of weight $W_{i,j}$ between each adjacent unit in the Kohonen layer was calculated. The Kohonen layer was then divided into two borders between the unit pair where the coefficient was minimal because the input group categorized into both sides of the border was presumed to have a large difference in topological characteristics; the weight of an adjacent unit pair would be updated by the neighborhood learning of the SOM to a similar value (Fig. 2(b)).
6. The groups of training data categorized into both sides of the divided Kohonen layers (N_1 and N_2, where $N = N_1 + N_2$) can be considered as two independent sub-problems (Fig. 2(b)). Actually, N_1 and N_2 were used as new training data, and processes 2)–5) were repeated recursively (Fig. 2(c)).

7. By repeating the processes described above, a hierarchical structure of the SOM (binary-tree structure) was generated (Fig. 2(d)). The lowermost layer of the hierarchical structure was defined as a facial expression category and five visualized images were defined as representative images of each category. Then the photographer of the facial expression images performed visual confirmation to each facial expression category and inferred their associated emotion categories.

The proposed method set the iterations of learning as 200,000 times. The radius of the neighborhood region $N_c(t)$ was fixed as the first neighborhood of the winner unit. The learning coefficient $a(t)$ was defined to decrease linearly from the initial value of 0.5–0.02 for learning iterations of 100,000 times; then subsequently to 0 at an iteration of learning of 200,000 times. The updating ratio of weights was set to 1 for the winner unit, and to 0.5 for its neighborhood units.

3.2 Generation of facial expression map with CPN

It is considered that recognition to a natural facial expression requires generation of a facial expression pattern (mixed facial expression) that interpolates each emotion category. The proposed method used the representative image obtained in Section 3.1 as training data and carried out data expansion of facial expression patterns among emotion categories using CPN with a large mapping space. The reason for adopting CPN, a supervised learning algorithm, is that the teaching signal of training data is known by processing in Section 3.1. The mapping space of CPN has a greater number of units than the number of training data; in addition, it has a toroidal structure because it is presumed that a large mapping space allows CPN to perform data expansion based on the similarity and continuity of training data. Figure 3 depicts the FEMap generation procedure. The processing details are described below.

1. The categories and representative images obtained in Section 3.1 were used as teaching signals and input data, which were then adopted as CPN training data.
2. The facial expression topological characteristics of an input group were learned using CPN with a two-dimensional Kohonen layer of 30 × 30 units and a Grossberg layer having as many units as the categories obtained in Section 3.1. The brightness values of the representative images were used as input data. Teaching signals to the Grossberg layer were set to 1 for units representing categories and 0 for the rest. The unit number of the input layer corresponded to the input image size.
3. The process described above was repeated until the maximum iterations of learning.
4. The weights (W_g) of the Grossberg layer were compared for each unit of the Kohonen layer after learning completion; an emotion category of the greatest value was used as the unit label.
5. A category map generated by the process described above was defined as a person-specific FEMap.

The proposed method set the iterations of learning as 20,000 times. The radius of the neighborhood region $N_c(t)$ was defined to decrease linearly from the initial value of the 14th to the first neighborhood of the winner unit at an iteration of learning of 10,000 times, and to be fixed at the first neighborhood of the winner unit for the subsequent 10,000 iterations. The learning coefficients $a(t)$ and $\beta(t)$ were defined to decrease linearly from the initial value of 0.5–0.02 at an iteration of learning of 10,000 times; then subsequently to 0 at an iteration of learning of 20,000 times. The updating ratio of weights was set to 1 for the winner unit, and to 0.5 for its neighboring units.

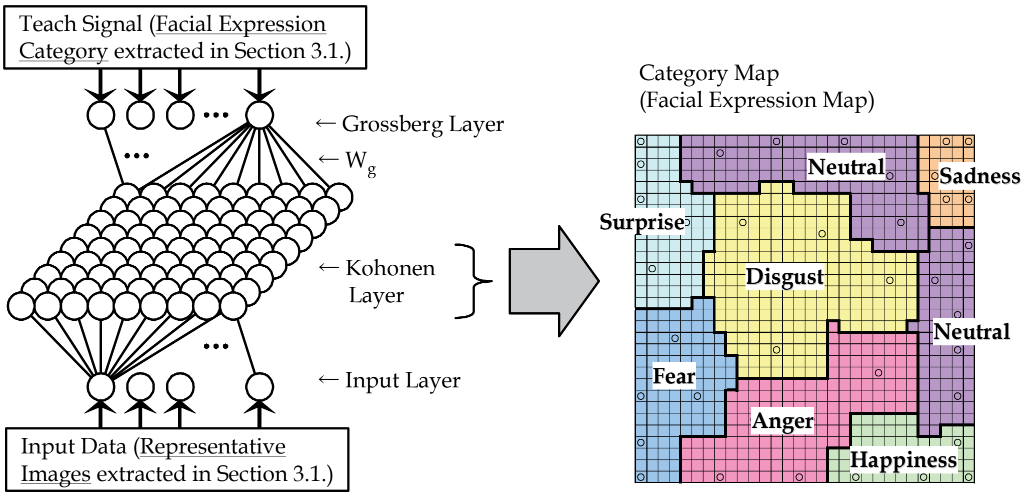

Teach Signal (Facial Expression Category extracted in Section 3.1.)
← Grossberg Layer
← W_g
← Kohonen Layer
← Input Layer
Input Data (Representative Images extracted in Section 3.1.)
Category Map (Facial Expression Map)
Neutral
Sadness
Surprise
Disgust
Neutral
Fear
Anger
Happiness

Fig. 3. Generation procedure of FEMap.

4. Facial expression images

Examples of facial expression images used in this study are presented in Fig. 4. This paper presents a discussion of six basic facial expressions and a neutral facial expression that six subjects manifested intentionally. Each subject's front face image was photographed under normal indoor conditions (lighting by fluorescent lamps) with the head enclosed inside the frame. Basic facial expressions were obtained as motion videos including a process in which a neutral facial expression and facial expressions were manifested five times respectively by turns for each facial expression. Neutral facial expressions were obtained as a motion video for about 10 s. The motion videos were converted into static images (10 frame/s, 8 bit gray, 320×240 pixels). Regions containing facial components, i.e., eyebrows, eyes, nose, and mouth, were extracted from each frame and used as training data. Table 1 presents the number of frames of all subjects' training data.

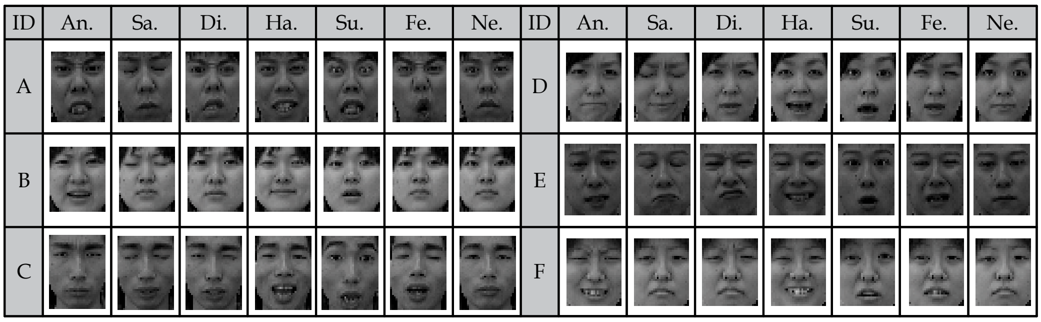


Fig. 4. Examples of facial expression images (ID, Subject; An., Anger; Sa., Sadness; Di., Disgust; Ha., Happiness; Su., Surprise; Fe., Fear; Ne., Neutral).

Open facial expression databases are generally used in conventional studies (Pantic et al., 2005; Gross, 2005). These databases contain a few images per expression and subject. For this study, we obtained facial expression images of ourselves because the proposed method extracts person-specific facial expression categories and the representative images of each category from large quantities of data.

ID	An.	Sa.	Di.	Ha.	Su.	Fe.	Ne.	Total
A	136	198	143	169	127	140	100	1013
B	152	136	153	162	154	190	100	1047
C	192	173	154	158	153	156	100	1086
D	152	158	178	177	158	170	100	1093
E	95	113	108	112	109	108	100	745
F	165	197	198	163	165	167	100	1155

Table 1. Numbers of frames of all subjects' training data.

5. Results and discussion

5.1 Extraction of person-specific facial expression categories

Figure 5 shows binary-tree structures generated with the proposed method applied to six subjects. Table 2 shows quantities of categories of facial expressions and representative images extracted from Fig. 5. Figure 5 shows that the binary-tree structure differs for each subject. Table 2 presents that the number of categories for each facial expression also differs for each subject.

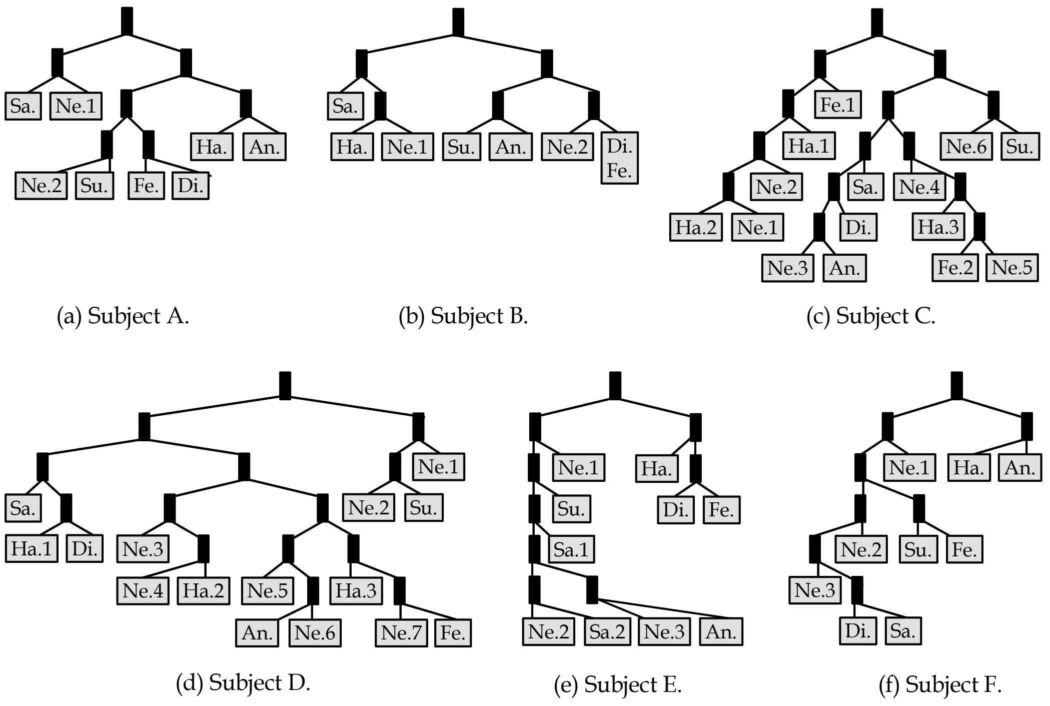


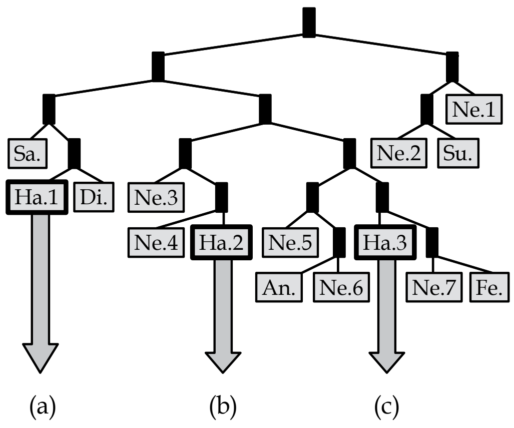
Fig. 5. Binary-tree structures generated with the proposed method.

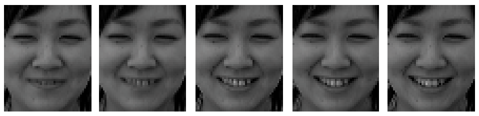
ID	An.	Sa.	Di.	Ha.	Su.	Fe.	Ne.	Extracted Categories	Representative Images
A	1	1	1	1	1	1	2	8	40 (5 units * 8)
B	1	1	1	1	1	-	2	7	35 (5 units * 7)
C	1	1	1	3	1	2	6	15	75 (5 units * 15)
D	1	1	1	3	1	1	7	15	75 (5 units * 15)
E	1	2	1	1	1	1	3	10	50 (5 units * 10)
F	1	1	1	1	1	1	3	9	45 (5 units * 9)

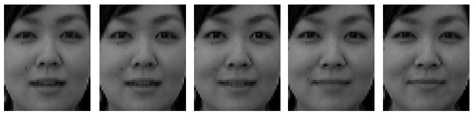
Table 2. Numbers of facial expression categories and representative images.

For Subject A, 8 categories are generated and 40 representative images are extracted. In fact, Subject A presented stable facial expression patterns within training data, and his six basic emotions were generated as one category each. A neutral expression was generated as two categories.

On the other hand, 15 categories were generated and 75 representative images were extracted for Subject D. Regarding happiness, three categories were generated from her one facial expression. Figure 6 shows representative images of happiness of Subject D, which reveals that three types of categories representing happiness were generated: (a) eyes are closed and the mouth is opened (showing teeth), (b) smiling, and (c) mouth is opened widely. These images suggest that the facial expression for the happiness of Subject D had multiple facial expression patterns, which were learned as different facial expression topological characteristics, and which were categorized into different categories in the binary-tree structure of SOM.




(a) Eyes are closed and mouth is opened (showing teeth).


(b) Smiling.

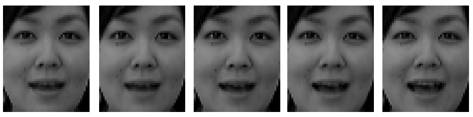

(c) Mouth is opened widely.

Fig. 6. Representative images of happiness of Subject D (Detail of Fig. 5(d)).

For Subject B, 7 categories were generated and 35 representative images were extracted. Regarding disgust and fear, both were classified into a single category (Fig. 7). Comparison of disgust and fear as facial expressions of Subject B shown in Fig. 4 suggests similarities in the patterns of facial expression and the consequent difficulty in visual distinction between both, which indicates that the binary-tree structure of SOM generated the facial expression of similar topological characteristics as one category.

The following were revealed. The proposed method enables classification of multiple facial expression patterns into separate different categories even if they are of the same facial expression. On the other hand, visually similar facial expressions are classifiable into one category.

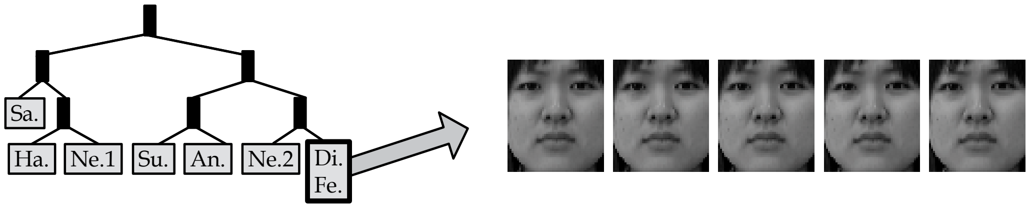


Fig. 7. Representative images of "disgust" and "fear" (Subject B, Detail of Fig. 5(b)).

Psychological significance is assigned to every category obtained with the binary-tree structure in this study. The operator might also assign importance to categories that are selected according to personal subjectivity. Moreover, intentional further hierarchization permits us to subdivide categories (subdivision of facial expression categorization). For example, Fig. 8 shows the subdivision result of the surprise category related to Subject E. The fourth layer, Fig. 8(a), was defined as a surprise category. Classification based on local and small changes of a facial expression pattern was performed by further intentional hierarchization: eyebrows are raised greatly (Fig. 8(b)), eyebrows are raised slightly (Fig. 8(c)), the mouth is opened narrowly (Figs. 8(d) and 8(f)), and the mouth is opened widely (Figs. 8(e) and 8(g)).

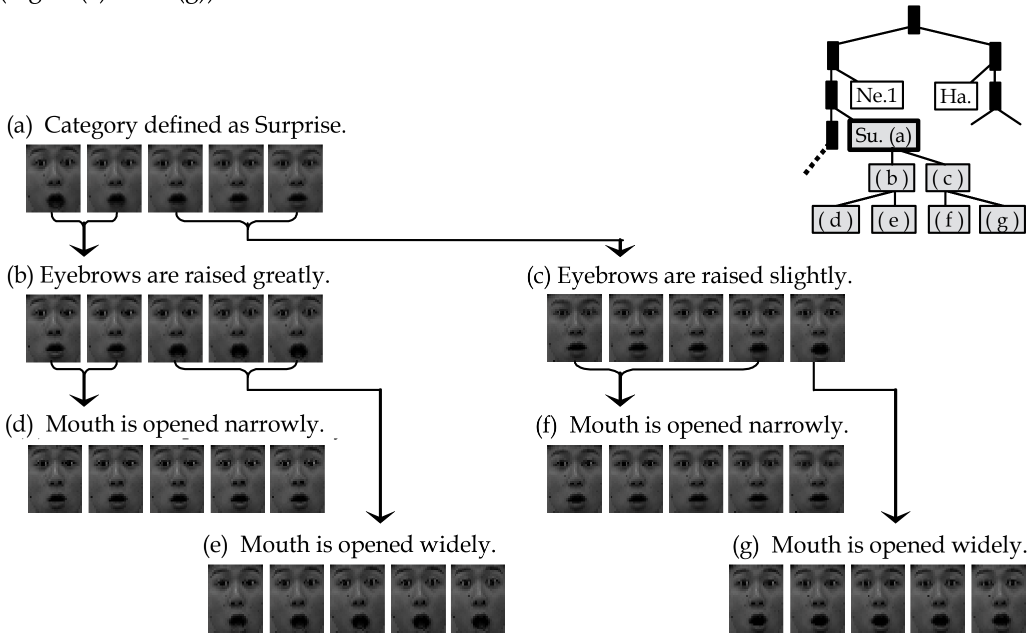

Ne.1
Ha.
Su. (a)
(b)
(c)
(d)
(e)
(f)
(g)
(a) Category defined as Surprise.
(b) Eyebrows are raised greatly.
(c) Eyebrows are raised slightly.
(d) Mouth is opened narrowly.
(f) Mouth is opened narrowly.
(e) Mouth is opened widely.
(g) Mouth is opened widely.

Fig. 8. Subdivision of a surprise category of Subject E (Detail of Fig. 5(e)).

5.2 Generation of facial expression map

The categories and representative images extracted in Section 5.1 were used respectively as teacher signals and input data of the CPN; the FEMaps shown in Fig. 9 were generated using the proposed method. Units with a round mark in the figures denote winner units when training data were input into the CPN after learning. These figures suggest that the area size of facial expression categories (number of labels) on FEMaps differs for each subject. Even within one subject, differences are apparent in the number of labels for each facial expression category.

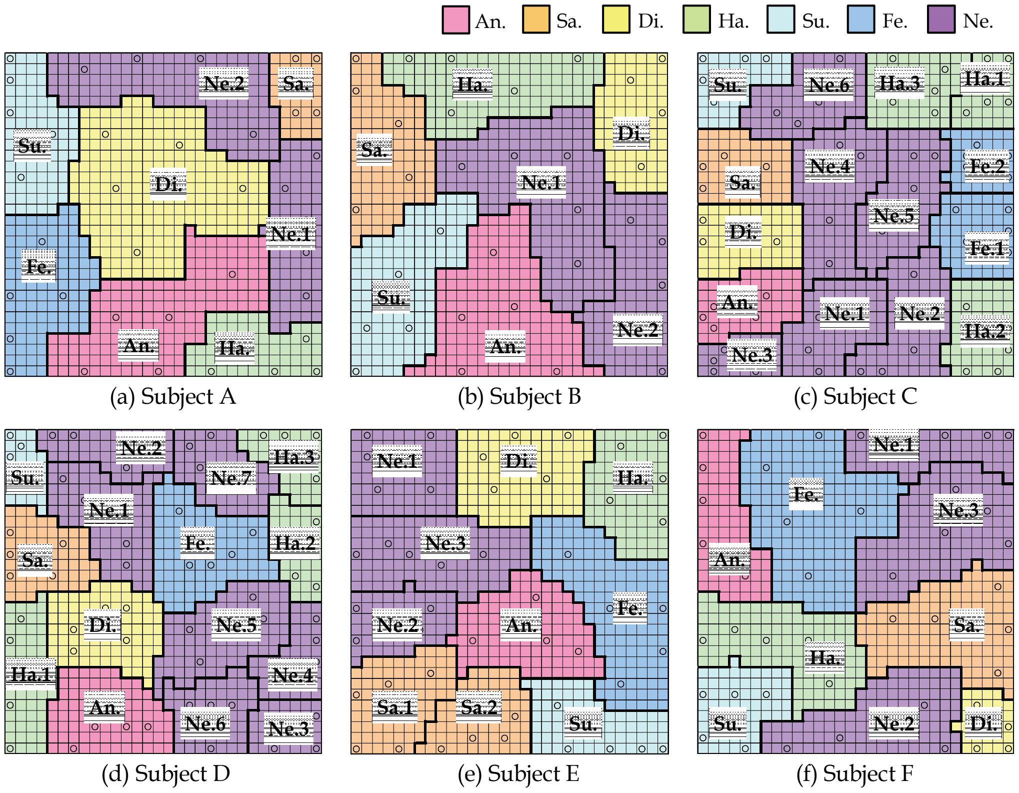

An.
Sa.
Di.
Ha.
Su.
Fe.
Ne.
Su. Ne.2 Sa. Di. Ne.1 Fe. An. Ha.
(a) Subject A
Ha. Di. Sa. Ne.1 Su. An. Ne.2
(b) Subject B
Su. Ne.6 Ha.3 Ha.1 Sa. Ne.4 Fe.2 Di. Ne.5 Fe.1 An. Ne.1 Ne.2 Ha.2 Ne.3
(c) Subject C
Ne.2 Ha.3 Su. Ne.7 Ne.1 Fe. Ha.2 Sa. Di. Ne.5 Ne.4 Ha.1 An. Ne.6 Ne.3
(d) Subject D
Ne.1 Di. Ha. Ne.3 Fe. Ne.2 An. Sa.1 Sa.2 Su.
(e) Subject E
Ne.1 Fe. Ne.3 An. Sa. Ha. Su. Ne.2 Di.
(f) Subject F

Fig. 9. FEMaps generated with the proposed method.

The percentages of the number of labels for each subject are listed in Table 3. Sadness and disgust occupy 4.1% and 25.2%, respectively, for Subject A. Even though training data of the same number for both categories (five images per category) are being used, great differences are apparent in the number of labels. Figure 9(a) portrays that winner units of training data for sadness are crowded, whereas those for disgust are dispersed widely, which are presumed to suggest the following Regarding sadness, the topological characteristics of training data are very similar compared to other facial expressions that the facial expression pattern changes little. However, for disgust, differences in the topological characteristics of training data are so large that the facial expression pattern changes greatly. For Subject D, the facial expression of happiness, for which three categories were generated, changes greatly (15.6%), although that of surprise shows little change (3.0%). For Subject F, the facial expression of fear changes greatly (20.2%), whereas that of disgust shows little change

(3.3%). These results suggest that the number of labels on an FEMap express the extent of difference of topological characteristics within a category, i.e., expressiveness of person-specific facial expressions.
Figure 10 portrays a magnified view of a part of surprise in the FEMap of Subject E, with the weights of each unit visualized. Units with the white frame in the figure denote winner units when training data were input into the CPN after learning. This figure suggests that facial expressions whose patterns differ slightly are generated in the neighborhood of five winner units. These results suggest that data expansion is performed based on the similarity and continuity of training data, and that more facial expression patterns such as mixed facial expressions between categories can be generated in the CPN mapping space.

ID	An.(%)	Sa. (%)	Di. (%)	Ha. (%)	Su. (%)	Fe. (%)	Ne. (%)
A	17.7	4.1	25.2	7.2	9.4	11.9	24.4
B	18.0	11.7	9.2	13.3	14.8	-	33.0
C	6.3	6.6	7.1	17.1	5.0	10.7	47.2
D	9.1	6.8	9.2	15.6	3.0	11.0	45.3
E	11.6	16.2	11.6	9.4	8.0	15.7	27.6
F	9.1	13.9	3.3	12.8	7.1	20.2	33.6

Table 3. Percentages of number of labels in the FEMap.

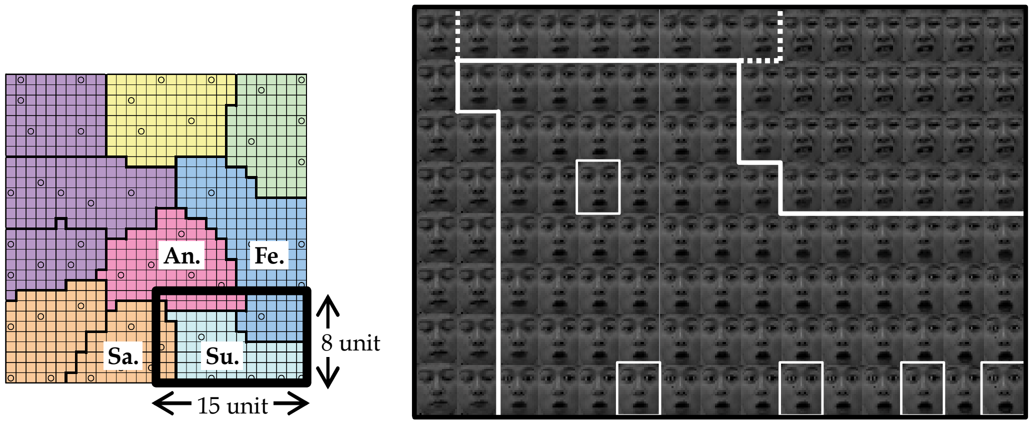


Fig. 10. Magnified view of a part of surprise in the FEMap of Subject E (Detail of Fig. 9(e)).

6. Conclusion

On the assumption that facial expression is a problem of classification into an unknown number of categories, this chapter describes an investigation of a generation method of a person-specific FEMap. The essential results obtained in this study are the following.
Hierarchical use of SOM with a narrow mapping space enables extraction of person-specific facial expression categories and representative images for each category. Psychological significance is assigned to every category obtained with the binary-tree structure in this study. The operator might also give special importance to categories selected according to personal subjectivity. Moreover, intentional further hierarchization of a binary-tree structure permits the additional subdivision of facial expression categorization.

The categories and category representative images obtained from the binary-tree structure were used as training data of a CPN with a large mapping space. Results revealed that data expansion is performed based on the similarity and continuity of training data, and that more facial expression patterns such as mixed facial expressions between categories can be generated in the CPN mapping space. It is expected that the use of an FEMap generated using the proposed method can be useful as a classifier in facial expression recognition that contributes to improvement in generalization capability.
This chapter specifically described a generation method of an FEMap and used facial expression images obtained during the same period. However, it is difficult to obtain all of a subject's facial expression patterns at one time; in addition, faces age with time. In future studies, we intend to take aging of a facial expression pattern into consideration, and study an automatic FEMap updating method using additional learning functions.

7. Acknowledgments

This work was supported in part by Ministry of Education, Culture, Sports, Science and Technology (MEXT) Grants-in-Aid for Young Scientists (B): No. 18700192 and No. 20700174.
This chapter is based on "Generation of Facial Expression Map based on Topological Characteristics of Face Images," by M. Ishii, K. Sato, H. Madokoro and M. Nishida, which appeared in the Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, (SMC 2007), Montreal, Quebec, Canada, October 2007. © 2007 IEEE.

8. References

Akamatsu, S. (1997). Computer Recognition of Human Face –A Survey-, *IEICE trans. Information and systems, Pt.2 (Japanese Edition)*, Vol.J80-D-II, No.8, pp.2031-2046.

Ekman, P. & Friesen, W.V. (1978). Facial Action Coding System, *Consulting Psychologist Press*.

Fasel, B., & Luettin, J. (2003). Automatic Facial Expression Analysis: A Survey, *Pattern Recognition*, Vol.36, pp.259-275.

Gross, R. (2005). Face Databases, *Handbook of Face Recognition*, S.Li and A.Jain, ed., Springer-Verlag.

Hasegawa, O.; Morishima, S. & Kaneko, M. (1997). Processing of Facial Information by Computer, *IEICE trans. Information and systems, Pt.2 (Japanese Edition)*, Vol.J80-D-II, No.8, pp.2047-2065.

Kohonen, T. (1995). Self-Organizing Maps, *Springer Series in Information Sciences*.

Lyons, M.J.; Budynek, J. & Akamatsu, S. (1999). Automatic Classification of Single Facial Images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.21, No.12, pp.1357-1362.

Lyons, M.J.; Akamatsu, S., Kamachi, M. & Gyoba, J. (1998). Coding Facial Expressions with GaborWavelets, *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pp.200-205.

Nielsen, R.H. (1987). Counterpropagation Networks, *Applied Optics*, vol.26, No.23, pp.4979-4984.

Pantic, M.; Valstar, M.F., Rademaker, R. & Maat, L. (2005). Webbased Database for Facial Expression Analysis, *Proc. IEEE Int. Conf. Multimedia and Expo*, pp.317-321.

Pantic, M. & Rothkrantz, L.J.M. (2000a). Automatic Analysis of Facial Expressions: The State of the Art, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.22, No.12, pp.1424-1445.

Pantic, M. & Rothkrantz, L.J.M. (2000b). Expert System for Automatic Analysis of Facial Expression, *Image and Vision Computing*, Vol.18, No.11, pp.881-905.

Tian, Y.L.; Kanade, T. & Cohn, J.F. (2001). Recognizing Action Units for Facial Expression Analysis, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.23, No.2, pp.97-116.

Yang, M.; Kriegman, D.J. & Ahuja, N. (2002). Detecting Faces in Images: A Survey, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.24, No.1, pp.34-58.

Zhang, Z.; Lyons, M., Schuster, M. & Akamatsu, S. (1998). Comparison Between Geometry-Based and Gabor-Wavelet-Based Facial Expression Recognition Using Multi-Layer Perceptron, *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, pp.454-459.

Zhao, W.; Chellappa, R., Rosenfeld, A. & Phillips, P.J. (2000). Face Recognition: A Literature Survey, *Technical Report 00-948*, University of Maryland.

13

Linear Subspace Learning for Facial Expression Analysis

Caifeng Shan
Philips Research
The Netherlands

1. Introduction

Facial expression, resulting from movements of the facial muscles, is one of the most powerful, natural, and immediate means for human beings to communicate their emotions and intentions. Some examples of facial expressions are shown in Fig. 1. Darwin (1872) was the first to describe in detail the specific facial expressions associated with emotions in animals and humans; he argued that all mammals show emotions reliably in their faces. Psychological studies (Mehrabian, 1968; Ambady & Rosenthal, 1992) indicate that facial expressions, with other non-verbal cues, play a major and fundamental role in face-to-face communication.

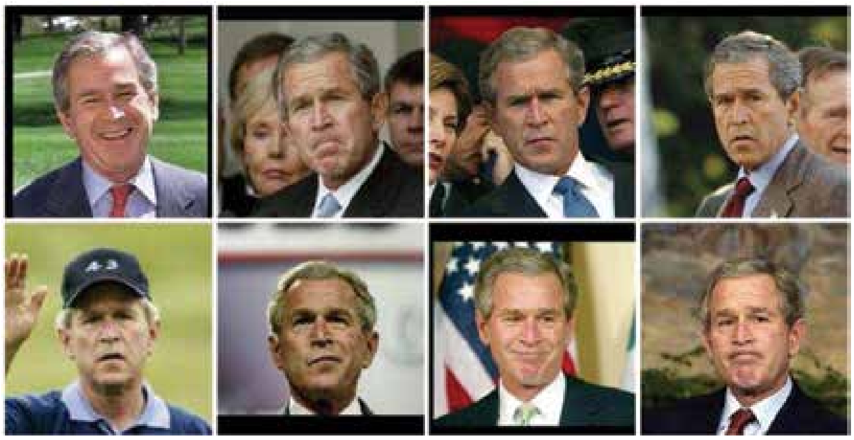


Fig. 1. Facial expressions of George W. Bush.

Machine analysis of facial expressions, enabling computers to analyze and interpret facial expressions as humans do, has many important applications including intelligent human-computer interaction, computer animation, surveillance and security, medical diagnosis, law enforcement, and awareness system (Shan, 2007). Driven by its potential applications and theoretical interests of cognitive and psychological scientists, automatic facial expression analysis has attracted much attention in last two decades (Pantic & Rothkrantz, 2000a; Fasel & Luettin, 2003; Tian et al, 2005; Pantic & Bartlett, 2007). It has been studied in multiple disciplines such as psychology, cognitive science, computer vision, pattern recognition, and human-computer interaction. Although much progress has been made, it is still difficult to design and develop an automated system capable of detecting and interpreting human facial expressions with high accuracy, due to their subtlety, complexity and variability.

Many machine learning techniques have been introduced for facial expression analysis, such as Neural Networks (Tian et al, 2001), Bayesian Networks (Cohen et al, 2003b), and Support Vector Machines (SVM) (Bartlett et al, 2005), to name just a few. Meanwhile, appearance-based statistical subspace learning has been shown to be an effective approach to modeling facial expression space for classification. This is because that despite a facial image space being commonly of a very high dimension, the underlying facial expression space is usually a sub-manifold of much lower dimensionality embedded in the ambient space. Subspace learning is a natural approach to resolve this problem. Traditionally, linear subspace methods including Principal Component Analysis (PCA) (Turk & Pentland, 1991), Linear Discriminant Analysis (LDA) (Belhumeur et al, 1997), and Independent Component Analysis (ICA) (Bartlett et al, 2002) have been used to discover both facial identity and expression manifold structures. For example, Lyons et al (1999) adopted PCA based LDA with the Gabor wavelet representation to classify facial images, and Donato et al (1999) explored PCA, LDA, and ICA for facial action classification.

Recently a number of nonlinear techniques have been proposed to learn the structure of a manifold, e.g., Isomap (Tenenbaum et al, 2000), Local Linear Embedding (LLE) (Roweis & Saul, 2000; Saul & Roweis, 2003), and Laplacian Eigenmaps (Belkin & Niyogi, 2001, 2003). These methods have been shown to be effective in discovering the underlying manifold. However, they are unsupervised in nature and fail to discover the discriminant structure in the data. Moreover, these techniques yield maps that are defined only on the training data, and it is unclear how to evaluate the maps for new test data. So they may not be suitable for pattern recognition tasks such as facial expression recognition. To address this problem, some linear approximations to these nonlinear manifold learning methods have been proposed to provide an explicit mapping from the input space to the reduced space (He & Niyogi, 2003; Kokiopoulou & Saad, 2005). He and Niyogi (2003) developed a linear subspace technique, known as Locality Preserving Projections (LPP), which builds a graph model that reflects the intrinsic geometric structure of the given data space, and finds a projection that respects this graph structure. LPP can be regarded as a linear approximation to Laplacian Eigenmaps; it can easily map any new data to the reduced space by using a transformation matrix. By incorporating the *priori* class information into LPP, we presented a Supervised LPP (SLPP) approach to enhance discriminant analysis on a manifold structure (Shan et al, 2005a). Cai et al (2006) further introduced a Orthogonal LPP (OLPP) approach to produce orthogonal basis vectors, which potentially have more discriminating power.

Orthogonal Neighborhood Preserving Projections (ONPP) is another interesting linear subspace technique proposed recently (Kokiopoulou & Saad, 2005, 2007). ONPP aims to preserve the intrinsic geometry of the local neighborhoods; it can be regarded as a linear approximation to LLE. ONPP constructs a weighted k-nearest neighbor graph which models explicitly the data topology, and, similarly to LLE, the weights are decided in a data-driven fashion to capture the geometry of local neighborhoods. In contrast to LLE, ONPP computes an explicit linear mapping from the input space to the reduced space. ONPP can be performed in either an unsupervised or a supervised setting. More recently Cai et al (2007) introduced a linear subspace method called Locality Sensitive Discriminant Analysis

(LSDA), which finds a projection that maximizes the margin between data points from different classes at each local area. LSDA constructs a nearest neighbor graph to model the geometrical structure of the underlying manifold, and then split it into *within-class* graph and *between-class* graph by using the class labels. LPP, ONPP, LSDA are all linear subspace learning techniques which aim at preserving locality of data samples, relying on a nearest neighbor graph to capture the data topology. However, they adopt totally different objective functions, so potentially they will provide different subspace learning power.

As different linear subspace techniques have been developed, the researchers are therefore confronted with a choice of algorithms with significantly different strengthes. However, to our best knowledge, there is no comprehensive comparative study on these linear subspace methods using the same data and experimental settings, although they were individually evaluated. In particular, for the task of facial expression analysis, it is necessary and important to identify the most effective linear subspace technique for facial expression representation and classification. In this chapter, we investigate and evaluate a number of linear subspace techniques for modeling facial expression subspace. Specifically we compare LPP and its variants SLPP and OLPP, ONPP, LSDA with the traditional PCA and LDA using different facial representations on several public databases. We find in our extensive study that the supervised LPP provides the best results in learning facial expression subspace, resulting in superior facial expression recognition performance. A short version of our work was presented in (Shan et al, 2006a).

The remainder of this chapter is organized as follows. We first survey the state of the art of facial expression analysis with machine learning (Section 2). Different linear subspace techniques compared in this chapter are described in Section 3. We present extensive experiments on different databases in Section 4, and finally Section 5 concludes the chapter.

2. State of the art

After Suwa et al (1978) made an early attempt to automatically analyze facial expressions from image sequences, machine analysis of facial expressions has received much attention in last two decades (Pantic & Rothkrantz, 2000a; Fasel & Luettin, 2003; Tian et al, 2005; Pantic & Bartlett, 2007). In this section, we review the state of the art on applying machine learning techniques for facial expression analysis.

Facial expressions can be described at different levels. Two mainstream description methods are facial affect (emotion) and facial muscle action (action unit) (Pantic & Bartlett, 2007). Most of facial expression analysis systems developed so far target facial affect analysis, attempting to analyze a set of prototypic emotional facial expressions (Pantic & Rothkrantz, 2000a, 2003). To describe subtle facial changes, Facial Action Coding System (FACS) (Ekman et al, 2002) has been used for manually labeling of facial actions. FACS associates facial changes with actions of the muscles that produce them. It defines 44 different action units (AUs). Another possible descriptor is the bipolar dimensions of *Valence* and *Arousal* (Russell, 1994). Valence describes the pleasantness, with positive (pleasant) on one end (e.g. happiness), and negative (unpleasant) on the other (e.g. disgust). The other dimension is arousal or activation, for example, sadness has low arousal, whereas surprise has a high arousal level.

The general approach to automatic facial expression analysis consists of three steps: face acquisition, facial data extraction & representation, and facial expression recognition. In the following sections, we discuss these steps respectively.

2.1 Face acquisition

Face acquisition is a pre-processing stage to automatically detect or locate the face region in the input images or sequences. Numerous techniques have been proposed for face detection (Yang et al, 2002), due to its practical importance in many computer vision applications. Most of existing methods emphasize statistical learning techniques and use appearance features. The real-time face detection scheme proposed by Viola and Jones (2001) is arguably the most commonly employed face detector, which consists of a cascade of classifiers trained by AdaBoost employing Harr-wavelet features. AdaBoost (Freund & Schapire, 1997; Schapire & Singer, 1999) is one of the most successful machine learning techniques applied in computer vision area, which provides a simple yet effective approach for stagewise learning of a nonlinear classification function. AdaBoost learns a small number of weak classifiers whose performance are just better than random guessing, and boosts them iteratively into a strong classifier of higher accuracy. Lienhart et al (2003) later made some extensions to this face detector. Many other machine learning techniques such as Neural Networks and SVM have also been introduced for face detection; details can be found in (Yang et al, 2002).

Most of face detectors can only detect faces in frontal or near-frontal view. To handle large head motion in video sequences, head tracking and head pose estimation can be adopted. The tasks of head tracking and pose estimation can be performed sequentially or jointly. Different approaches have been developed for head pose estimation (Murphy-Chutorian & Trivedi, 2008). Given the success of frontal face detectors, a natural extension is to estimate head pose by training multiple face detectors, each to specific a different discrete pose. For example, cascade AdaBoost detectors have been extended for pose estimation (Jones & Viola, 2003). Recently manifold learning approaches have been adopted to seek low-dimensional manifolds that model the continuous variation in head pose; new images can then be embedded into these manifolds for pose estimation. Nonlinear methods such as Isomap, LLE, and Laplacian Eigenmaps or their linear approximations have been exploited for pose estimation (Fu & Huang, 2006; Balasubramanian et al, 2008).

2.2 Facial feature extraction & representation

Facial feature extraction and representation is to derive a set of features from original face images which are used for representing faces. Two types of features, geometric features and appearance features, are usually considered for facial representation. Geometric features deal with the shape and locations of facial components (including mouth, eyes, brows, and nose), which are extracted to represent the face geometry (Zhang et al, 1998; Pantic & Rothkrantz, 2000b; Tian et al, 2001; Kaliouby & Robinson, 2004; Zhang & Ji, 2005; Pantic & Bartlett, 2007). Appearance features present the appearance changes (skin texture) of the face (including wrinkles, bulges and furrows), which are extracted by applying image filters to either the whole face or specific facial regions (Lyons et al, 1999; Donato et al, 1999; Bartlett et al, 2003; Shan et al, 2005c; Littlewort et al, 2006; Gritti et al, 2008). The geometric features based facial representations commonly require accurate and reliable facial feature detection and tracking, which is difficult to accommodate in real-world unconstrained scenarios, e.g., under head pose variation. In contrast, appearance features suffer less from issues of initialization and tracking errors, and can encode changes in skin texture that are critical for facial expression modeling. However, most of the existing appearance-based facial representations still require face registration based on facial feature detection, e.g., eye detection.

Machine learning techniques have been exploited to select the most effective features for facial representation. Donato et al (1999) compared different techniques to extract facial features, which include PCA, LDA, LDA, Local Feature Analysis, and local principal components. The experimental results provide evidence for the importance of using local filters and statistical independence for facial representation. Bartlett et al (2003, 2005) presented to select a subset of Gabor filters using AdaBoost. Similarly, Wang et al (2004) learned a subst of Harr features using Adaboost. Whitehill and Omlin (2006) compared Gabor filters, Harr-like filters, and the edge-oriented histogram for AU recognition, and found that AdaBoost performs better with Harr-like filters, while SVMs perform better with Gabor filters. Valstar and Pantic (2006) recently presented a fully automatic AU detection system that can recognize AU temporal segments using a subset of most informative spatio-temporal features selected by AdaBoost. In our previous work (Shan et al, 2005b; Shan & Gritti, 2008), we also adopted boost learning to learn discriminative Local Binary Patterns features for facial expression recognition.

2.3 Facial expression recognition

The last stage is to classify different expressions based on the extracted facial features. Facial expression recognition can be generally divided into image-based or sequence-based. The image-based approaches use features extracted from a single image to recognize the expression of that image, while the sequence-based methods aim to capture the temporal pattern in a sequence to recognize the expression for one or more images. Different machine learning techniques have been proposed, such as Neural Network (Zhang et al, 1998; Tian et al, 2001), SVM (Bartlett et al, 2005, 2003), Bayesian Network (Cohen et al, 2003b,a), and rule-based classifiers (Pantic & Rothkrantz, 2000b) for image-based expression recognition, or Hidden Markov Model (HMM) (Cohen et al, 2003b; Yeasin et al, 2004) and Dynamic Bayesian Network (DBN) (Kaliouby & Robinson, 2004; Zhang & Ji, 2005) for sequence-based expression recognition.

Pantic and Rothkrantz (2000b) performed facial expression recognition by comparing the AU-coded description of an observed expression against rule descriptors of six basic emotions. Recently they further adopted the rule-based reasoning to recognize action units and their combination (Pantic & Rothkrantz, 2004). Tian et al (2001) used a three-layer Neural Network with one hidden layer to recognize AUs by a standard back-propagation method. Cohen et al (2003b) adopted Bayesian network classifiers to classify a frame in video sequences to one of the basic facial expressions. They compared Naive-Bayes classifiers where the features are assumed to be either Gaussian or Cauchy distributed, and Gaussian Tree-Augmented Naive Bayes classifiers. Because it is difficult to collect a large amount of training data, Cohen et al (2004) further proposed to use unlabeled data together with labeled data using Bayesian networks. As a powerful discriminative machine learning technique, SVM has been widely adopted for facial expression recognition. Recently Bartlett et al (2005) performed comparison of AdaBoost, SVM, and LDA, and best results were obtained by selecting a subset of Gabor filters using AdaBoost and then training SVM on the outputs of the selected filters. This strategy is also adopted in (Tong et al, 2006; Valstar & Pantic, 2006).

Psychological experiments (Bassili, 1979) suggest that the dynamics of facial expressions are crucial for successful interpretation of facial expressions. HMMs have been exploited to capture temporal behaviors exhibited by facial expressions (Oliver et al, 2000; Cohen et al, 2003b; Yeasin et al, 2004). Cohen et al (2003b) proposed a multi-level HMM classifier, which allows not only to perform expression classification in a video segment, but also to automatically segment an arbitrary long video sequence to the different expressions segments without resorting to heuristic methods of segmentation. DBNs are graphical probabilistic models which encode dependencies among sets of random variables evolving in time. HMM is the simplest kind of DBNs. Zhang and Ji (2005) explored the use of multisensory information fusion technique with DBNs for modeling and understanding the temporal behaviors of facial expressions in image sequences. Kaliouby and Robinson (2004) proposed a system for inferring complex mental states from videos of facial expressions and head gestures in real-time. Their system was built on a multi-level DBN classifier which models complex mental states as a number of interacting facial and head displays. Facial expression dynamics can also be captured in low dimensional manifolds embedded in the input image space. Chang et al (2003, 2004) made attempts to learn the structure of the expression manifold. In our previous work (Shan et al, 2005a, 2006b), we presented to model facial expression dynamics by discovering the underlying low-dimensional manifold.

3. Linear subspace methods

The goal of subspace learning (or dimensionality reduction) is to map the data set in the high dimensional space to the lower dimensional space such that certain properties are preserved. Examples of properties to be preserved include the global geometry and neighborhood information. Usually the property preserved is quantified by an objective function and the dimensionality reduction problem is formulated as an optimization problem. The generic problem of linear dimensionality reduction is the following. Given a multi-dimensional data set $x_1, x_2, \dots, x_m$ in R^n, find a transformation matrix W that maps these m points to $y_1, y_2, \dots, y_m$ in $R^l (l \ll n)$, such that y_i represent x_i, where $y_i = W^T x_i$. In this section, we briefly review the existing linear subspace methods PCA, LDA, LPP, ONPP, LSDA, and their variants.

3.1 Principle Component Analysis (PCA)

Two of the most popular techniques for linear subspace learning are PCA and LDA. PCA (Turk & Pentland, 1991) is an eigenvector method designed to model linear variation in high-dimensional data. PCA aims at preserving the global variance by finding a set of mutual orthogonal basis functions that capture the directions of maximum variance in the data.

Let **w** denote a transformation vector, the objective function is as follows:

$$\max_{\mathbf{w}} \sum_i (y_i - \bar{y})^2 \quad \text{where} \quad \bar{y} = \frac{1}{m} \sum_i y_i \tag{1}$$

The solution $\mathbf{w}_0, \dots, \mathbf{w}_{l-1}$ is an orthonormal set of vectors representing the eigenvector of the data's covariance matrix associated with the l largest eigenvalues.

3.2 Linear Discriminant Analysis (LDA)

While PCA is an unsupervised method and seeks directions that are efficient for representation, LDA (Belhumeur et al, 1997) is a supervised approach and seeks directions

that are efficient for discrimination. LDA searches for the projection axes on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other.
Suppose the data samples belong to *c* classes, The objective function is as follows:

$$\max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \tag{2}$$

$$S_B = \sum_{i=1}^{c} n_i (\mathbf{m}^{(i)} - \mathbf{m})(\mathbf{m}^{(i)} - \mathbf{m})^T \tag{3}$$

$$S_W = \sum_{i=1}^{c} \left(\sum_{j=1}^{n_i} (x_j^{(i)} - \mathbf{m}^{(i)})(x_j^{(i)} - \mathbf{m}^{(i)})^T \right) \tag{4}$$

where $\mathbf{m}$ is the mean of all the samples, n_i is the number of samples in the *i*th class, $\mathbf{m}_{(i)}$ is the average vector of the *i*th class, and $x_j^{(i)}$ is the *j*th sample in the *i*th class.

3.3 Locality Preserving Projections (LPP)

LPP (He & Niyogi, 2003) seeks to preserve the intrinsic geometry of the data by preserving locality. To derive the optimal projections preserving locality, LPP employs the same objective function with Laplacian Eigenmaps:

$$\min_{\mathbf{w}} \sum_{i,j} \left(\mathbf{w}^T x_i - \mathbf{w}^T x_j\right)^2 S_{ij} \tag{5}$$

where $S_{i\,j}$ evaluates a local structure of the data space, and is defined as:

$$S_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{t}} & \text{if } x_i \text{ and } x_j \text{ are ``close''} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

or in a simpler form as

$$S_{ij} = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ are ``close''} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where "close" can be defined as $\|x_i - x_j\|^2 < \varepsilon$, where ε is a small constant, or x_i is among k nearest neighbors of x_j or x_j is among k nearest neighbors of x_i. The objective function with symmetric weights $S_{i\,j}(S_{i\,j} = S_{ji})$ incurs a heavy penalty if neighboring points x_i and x_j are mapped far apart. Minimizing their distance is therefore an attempt to ensure that if x_i and x_j are "close", $y_i(= \mathbf{w}^T x_i)$ and $y_j(= \mathbf{w}^T x_j)$ are also "close". The objective function of Eqn. (5) can be reduced to:

$$\begin{aligned} \frac{1}{2}\sum_{ij}(\mathbf{w}^T x_i - \mathbf{w}^T x_j)^2 S_{ij} &= \sum_i \mathbf{w}^T x_i D_{ii} x_i^T \mathbf{w} - \sum_{ij} \mathbf{w}^T x_i S_{ij} x_j^T \mathbf{w} \\ &= \mathbf{w}^T X(D - S)X^T \mathbf{w} \\ &= \mathbf{w}^T X L X^T \mathbf{w} \end{aligned} \tag{8}$$

where $X = [x_1, x_2, \ldots, x_m]$ and D is a diagonal matrix whose entries are column (or row, since S is symmetric) sums of S, $D_{ii} = \sum_j S_{ji}$. $L = D - S$ is a Laplacian matrix. D measures the local density on the data points. The bigger the value D_{ii} is (corresponding to y_i), the more important is y_i. Therefore, a constraint is imposed as follows:

$$\mathbf{y}^T D\mathbf{y} = 1 \Rightarrow \mathbf{w}^T XDX^T \mathbf{w} = 1 \tag{9}$$

The transformation vector **w** that minimizes the objective function is given by the minimum eigenvalue solution to the following generalized eigenvalue problem:

$$XLX^T \mathbf{w} = \lambda XDX^T \mathbf{w} \tag{10}$$

Suppose a set of vectors $\mathbf{w}_0, \ldots, \mathbf{w}_{l-1}$ is the solution, ordered according to their eigenvalues, $\lambda_0, \ldots, \lambda_{l-1}$, the transformation matrix is derived as $W = [\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{l-1}]$.

3.3.1 Supervised Locality Preserving Projections (SLPP)

When the class information is available, LPP can be performed in a supervised manner. We introduced a Supervised LPP to encode more discriminative power than the original LPP for improving classification capacity (Shan et al, 2005a). SLPP preserves the class information when constructing a neighborhood graph such that the local neighborhood of a sample x_i from class c is composed of samples belonging to class c only. This can be achieved by increasing the distances between samples belonging to different classes, but leaving them unchanged if they are from the same class. Let $Dis(i, j)$ denote the distance between x_i and x_j, the distance after incorporating the class information is then

$$SupDis(i,j) = Dis(i,j) + M\delta(i,j) \tag{11}$$

where $M = \max_{i,j} Dis(i, j)$, and $\delta(i, j) = 1$ if x_i and x_j belong to different classes, and 0 otherwise. In this way, distances between samples in different classes will be larger than the maximum distance in the entire data set, so neighbors of a sample will always be picked from the same class. SLPP preserves both local structure and class information in the embedding, so that it better describes the intrinsic structure of a data space containing multiple classes.

3.3.2 Orthogonal Locality Preserving Projections (OLPP)

The basis vectors derived by LPP can be regarded as the eigenvectors of the matrix $(XDX^T)^{-1}XLX^T$ corresponding to the smallest eigenvalues. Since $(XDX^T)^{-1}XLX^T$ is not symmetric in general, these basis vectors are non-orthogonal. Cai et al (2006) presented Orthogonal LPP to enforce the mapping to be orthogonal. The orthogonal basis vectors $\{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{l-1}\}$ are computed as follows.

- Compute $\mathbf{w}_0$ as the eigenvector of $(XDX^T)^{-1}XLX^T$ associated with the smallest eigenvalue.
- Compute $\mathbf{w}_k$ as the eigenvector of

$$M^{(k)} = \left(I - (XDX^T)^{-1}A^{(k-1)}(B^{(k-1)})^{-1}(A^{(k-1)})^T\right)(XDX^T)^{-1}XLX^T \tag{12}$$

associated with the smallest eigenvalue, where

$$A^{(k-1)} = [\mathbf{w}_0, \dots, \mathbf{w}_{k-1}] \tag{13}$$

$$B^{(k-1)} = (A^{(k-1)})^T (XDX^T)^{-1} A^{(k-1)} \tag{14}$$

OLPP can be applied under supervised or unsupervised mode. In this chapter, for facial expression analysis, OLPP is performed in the supervised manner as described in Section 3.3.1.

3.4 Orthogonal Neighborhood Preserving Projections (ONPP)

ONPP (Kokiopoulou & Saad, 2005, 2007) seeks an orthogonal mapping of a given data set so as to best preserve the local geometry. The first step of ONPP, identical with that of LLE, builds an affinity matrix by computing optimal weights which reconstruct each sample by a linear combination of its k nearest neighbors. The reconstruction errors are measured by minimizing

$$\varepsilon(V) = \sum_i |x_i - \sum_j v_{ij} x_j|^2 \tag{15}$$

The weights v_{ij} represent the linear coefficient for reconstructing the sample x_i from its neighbors $\{x_j\}$. The following constraints are imposed on the weights:

1. $v_{ij} = 0$, if x_j is not one of the k nearest neighbors of x_i.
2. $\sum_j v_{ij} = 1$, that is x_i is approximated by a convex combination of its neighbors.

In the second step, ONPP derives an explicit linear mapping from the input space to the reduced space. ONPP imposes a constraint that each data sample y_i in the reduced space is reconstructed from its k nearest neighbors by the same weights used in the input space, so it employs the same objective function with LLE:

$$\min_Y \sum_i |y_i - \sum_j v_{ij} y_j|^2 \tag{16}$$

where the weights v_{ij} are fixed. The optimization problem can be reduced to

$$\begin{aligned} \min_Y \sum_i |y_i - \sum_j v_{ij} y_j|^2 &= \min_W \sum_i |W^T x_i - \sum_j v_{ij} W^T x_j|^2 \\ &= \min_W \|W^T X(I - V^T)\|_F^2 \\ &= \min_W tr(W^T X(I - V^T)(I - V)X^T W) \\ &= \min_W tr(W^T XMX^T W) \end{aligned} \tag{17}$$

where $M = (I - V^T)(I - V)$. By imposing an additional constraint that the columns of W are orthogonal, the solution to the above optimization problem is the eigenvectors associated with the d smallest eigenvalues of the matrix

$$\tilde{M} = X(I - V^T)(I - V)X^T. \tag{18}$$

ONPP can be performed in either an unsupervised or a supervised setting. In the supervised ONPP, when building the data graph, an edge exists between x_i and x_j if and only if x_i and x_j belong to the same class. This means that the adjacent data samples in the nearest neighbor graph belong to the same class. So there is no need to set parameter k in the supervised ONPP.

3.5 Locality Sensitive Discriminant Analysis (LSDA)

Given a data set, LSDA (Cai et al, 2007) constructs two graphs, *within-class graph* G_w and *between-class graph* G_b, in order to discover both geometrical and discriminant structure of the data. For each data sample x_i, let $N(x_i)$ be the set of its k nearest neighbors. $N(x_i)$ can be naturally split into two subsets, $N_b(x_i)$ and $N_w(x_i)$. $N_w(x_i)$ contains the neighbors sharing the same label with x_i, while $N_b(x_i)$ contains neighbors have different labels. Let S_w and S_b be the weight matrices of G_w and G_b respectively, which can be defined as follows

$$S_{b,ij} = \begin{cases} 1 & \text{if } x_i \in N_b(x_j) \text{ or } x_j \in N_b(x_i) \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

$$S_{w,ij} = \begin{cases} 1 & \text{if } x_i \in N_w(x_j) \text{ or } x_j \in N_w(x_i) \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

To derive the optimal projections, LSDA optimizes the following objective functions

$$\min_{\mathbf{w}} \sum_{i,j} \left(\mathbf{w}^T x_i - \mathbf{w}^T x_j\right)^2 S_{w,ij} \tag{21}$$

$$\max_{\mathbf{w}} \sum_{i,j} \left(\mathbf{w}^T x_i - \mathbf{w}^T x_j\right)^2 S_{b,ij} \tag{22}$$

Similar to Eqn (8), the objective function (21) can be reduced to

$$\begin{aligned} \frac{1}{2}\sum_{ij}(\mathbf{w}^T x_i - \mathbf{w}^T x_j)^2 S_{w,ij} &= \sum_i \mathbf{w}^T x_i D_{w,ii} x_i^T \mathbf{w} - \sum_{ij} \mathbf{w}^T x_i S_{w,ij} x_j^T \mathbf{w} \\ &= \mathbf{w}^T X (D_w - S_w) X^T \mathbf{w} \\ &= \mathbf{w}^T X L_w X^T \mathbf{w} \end{aligned} \tag{23}$$

where D_w is a diagonal matrix, and its entries $D_{w,ii} = \sum_j S_{w,ji}$. Similarly, the objective function (22) can be reduced to

$$\frac{1}{2}\sum_{ij}(\mathbf{w}^T x_i - \mathbf{w}^T x_j)^2 S_{b,ij} = \mathbf{w}^T X (D_b - S_b) X^T \mathbf{w} = \mathbf{w}^T X L_b X^T \mathbf{w} \tag{24}$$

Similar to LPP, a constraint is imposed as follows:

$$\mathbf{y}^T D_w \mathbf{y} = 1 \Rightarrow \mathbf{w}^T X D_w X^T \mathbf{w} = 1 \tag{25}$$

The transformation vector **w** that minimizes the objective function is given by the maximum eigenvalue solution to the generalized eigenvalue problem:

$$X(\alpha L_b + (1-\alpha)S_w)X^T\mathbf{w} = \lambda X D_w X^T \mathbf{w} \quad (26)$$

In practice, the dimension of the feature space (n) is often much larger than the number of samples in a training set (m), which brings problems to LDA, LPP, ONPP, and LSDA. To overcome this problem, the data set is first projected into a lower dimensional space using PCA.

4. Experiments

In this section, we evaluate the above linear subspace methods for facial expression analysis with the same data and experimental settings. We use implementations of LPP, SLPP, OLPP, ONPP and LSDA provided by the authors.
Psychophysical studies indicate that basic emotions have corresponding universal facial expressions across all cultures (Ekman & Friesen, 1976). This is reflected by most current facial expression recognition systems that attempt to recognize a set of prototypic emotional expressions including disgust, fear, joy, surprise, sadness and anger (Lyons et al, 1999; Cohen et al, 2003b; Tian, 2004; Bartlett et al, 2005). In this study, we also focus on these prototypic emotional expressions. We conducted experiments on three public databases: the Cohn-Kanade Facial Expression Database (Kanade et al, 2000), the MMI Facial Expression Database (Pantic et al, 2005), and the JAFFE Database (Lyons et al, 1999), which are the most commonly used databases in the current facial-expression-research community.
In all experiments, we normalized the original face images to a fixed distance between the two eyes. Facial images of 110×150 pixels, with 256 gray levels per pixel, were cropped from original frames based on the two eyes location. No further alignment of facial features such as alignment of mouth (Zhang et al, 1998), or removal of illumination changes (Tian, 2004) was performed in our experiments. Fig. 2 shows an example of the original image and the cropped face image.

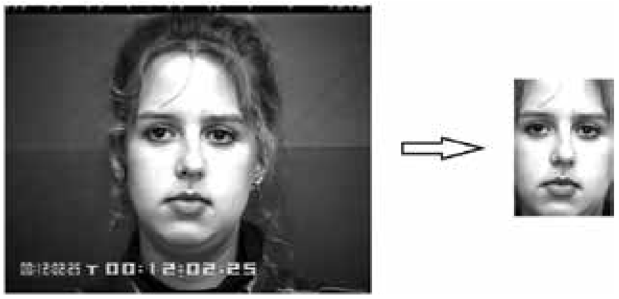


Fig. 2. The original face image and the cropped image.

4.1 Facial representation

To perform facial expression analysis, it is necessary to derive an effective facial representation from original face images. Gabor-wavelet representations have been widely adopted to describe appearance changes of faces (Tian, 2004; Bartlett et al, 2005). However, the computation of Gabor features is both time and memory intensive. In our previous work (Shan et al, 2005c), we proposed Local Binary Patterns (LBP) features as low-cost discriminant appearance features for facial expression analysis. The LBP operator, originally introduced by Ojala et al (2002) for texture analysis, labels the pixels of an image by thresholding a neighborhood of each pixel with the center value and considering the results as a binary number. The histogram of the labels computed over a region can be used as a texture descriptor. The most important properties of the LBP operator are its tolerance against illumination changes and its computational simplicity. LBP features recently have been exploited for face detection and recognition (Ahonen et al, 2004).

In the existing work (Ahonen et al, 2004; Shan et al, 2005c), the face image is equally divided into small regions from which LBP histograms are extracted and concatenated into a single feature histogram (as shown in Fig. 3). However, this LBP feature extraction scheme suffers from fixed LBP feature size and positions. By shifting and scaling a sub-window over face images, many more LBP histograms could be obtained, which yields a more complete description of face images. To minimize the large number of LBP histograms necessarily introduced by shifting and scaling a sub-window, we proposed to learn the most effective LBP histograms using AdaBoost (Shan et al, 2005b). The boosted LBP features provides a compact and discriminant facial representation. Fig. 4 shows examples of the selected subregions (LBP histograms) for each basic emotional expression. It is observed that the selected sub-regions have variable sizes and positions.

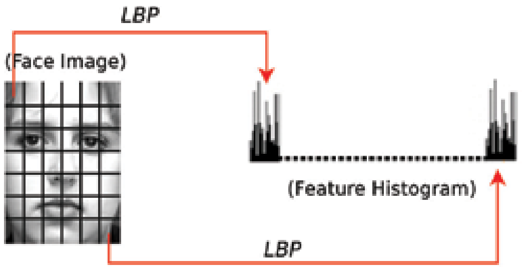


Fig. 3. A face image is divided into small regions from which LBP histograms are extracted and concatenated into a single, spatially enhanced feature histogram.

In this study, three facial representations were considered: raw gray-scale image (IMG), LBP features extracted from equally divided sub-regions (LBP), and Boosted LBP features (BLBP). On IMG features, for computational efficiency, we down-sampled the face images to 55×75 pixels, and represented each image as a 4,125(55×75)-dimensional vector. For LBP features, as shown in Fig. 3, we divided facial images into 42 sub-regions; the 59-bin $LBP_{8,2}^{u2}$ operator (Ojala et al, 2002) was applied to each sub-region. So each image was represented by a LBP histogram with length of 2,478(59×42). For BLBP features, by shifting and scaling a sub-window, 16,640 sub-regions, i.e., 16,640 LBP histograms, were extracted from each face image; AdaBoost was then used to select the most discriminative LBP histograms. AdaBoost training continued until the classifier output distribution for the positive and negative samples were completely separated.

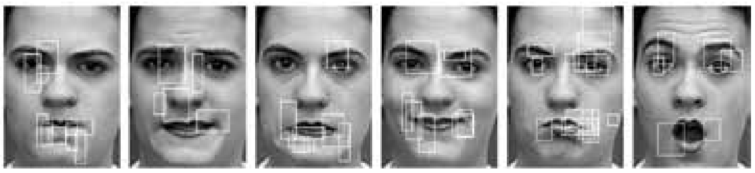


Fig. 4. Examples of the selected sub-regions (LBP histograms) for each of the six basic emotions in the Cohn-Kanade Database (from left to right: Anger, Disgust, Fear, Joy, Sadness, and Surprise).

4.2 Cohn-Kanade database

The Cohn-Kanade Database (Kanade et al, 2000) consists of 100 university students in age from 18 to 30 years, of which 65% were female, 15% were African-American, and 3% were Asian or Latino. Subjects were instructed to perform a series of 23 facial displays, six of which were prototypic emotions. Image sequences from neutral face to target display were digitized into 640×490 pixel arrays. Fig. 5 shows some sample images from the database.

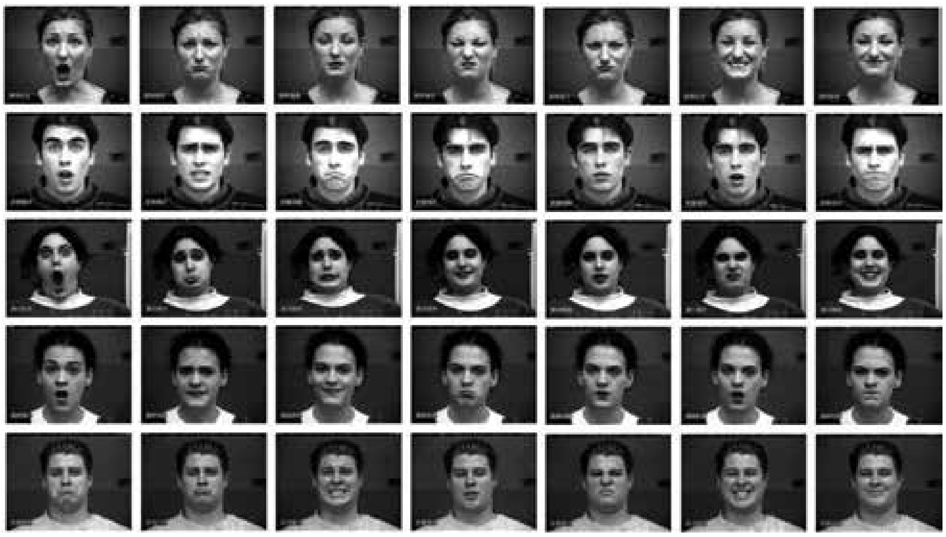


Fig. 5. The sample face expression images from the Cohn-Kanade Database.

In our experiments, 320 image sequences were selected from the database. The only selection criterion is that a sequence can be labeled as one of the six basic emotions. The sequences come from 96 subjects, with 1 to 6 emotions per subject. Two data sets were constructed: (1) **S1**: the three peak frames (typical expression at apex) of each sequence were used for 6-class expression analysis, resulting in 960 images (108 Anger, 120 Disgust, 99 Fear, 282 Joy, 126 Sadness, and 225 Surprise); (2) **S2**: the neutral face of each sequence was further included for 7-class expression analysis, resulting in 1,280 images (960 emotional images plus 320 neutral faces).

4.2.1 Comparative evaluation on subspace learning

As presented in (Shan et al, 2006a), we observed in our experiments on all databases that ONPP and the supervised ONPP achieve comparable performance in expression subspace learning and expression recognition. It seems that the label information used in the supervised ONPP does not provide it with more discriminative power than ONPP for facial expression analysis. Therefore, in this chapter, we focus on the evaluation of the supervised ONPP. We also found in our experiments that the supervised OLPP provides similar results with SLPP, so we mainly focus on the evaluation of SLPP in this chapter.

The 2D visualization of embedded subspaces of data set **S1** is shown in Fig. 6. In the six methods compared, PCA and LPP are unsupervised techniques, while LDA, SLPP, ONPP, and LSDA perform in a supervised manner. It is evident that the classes of different expressions are heavily overlapped in 2D subspaces generated by unsupervised methods PCA and LPP (with all three facial representations), therefore are poorly represented. The

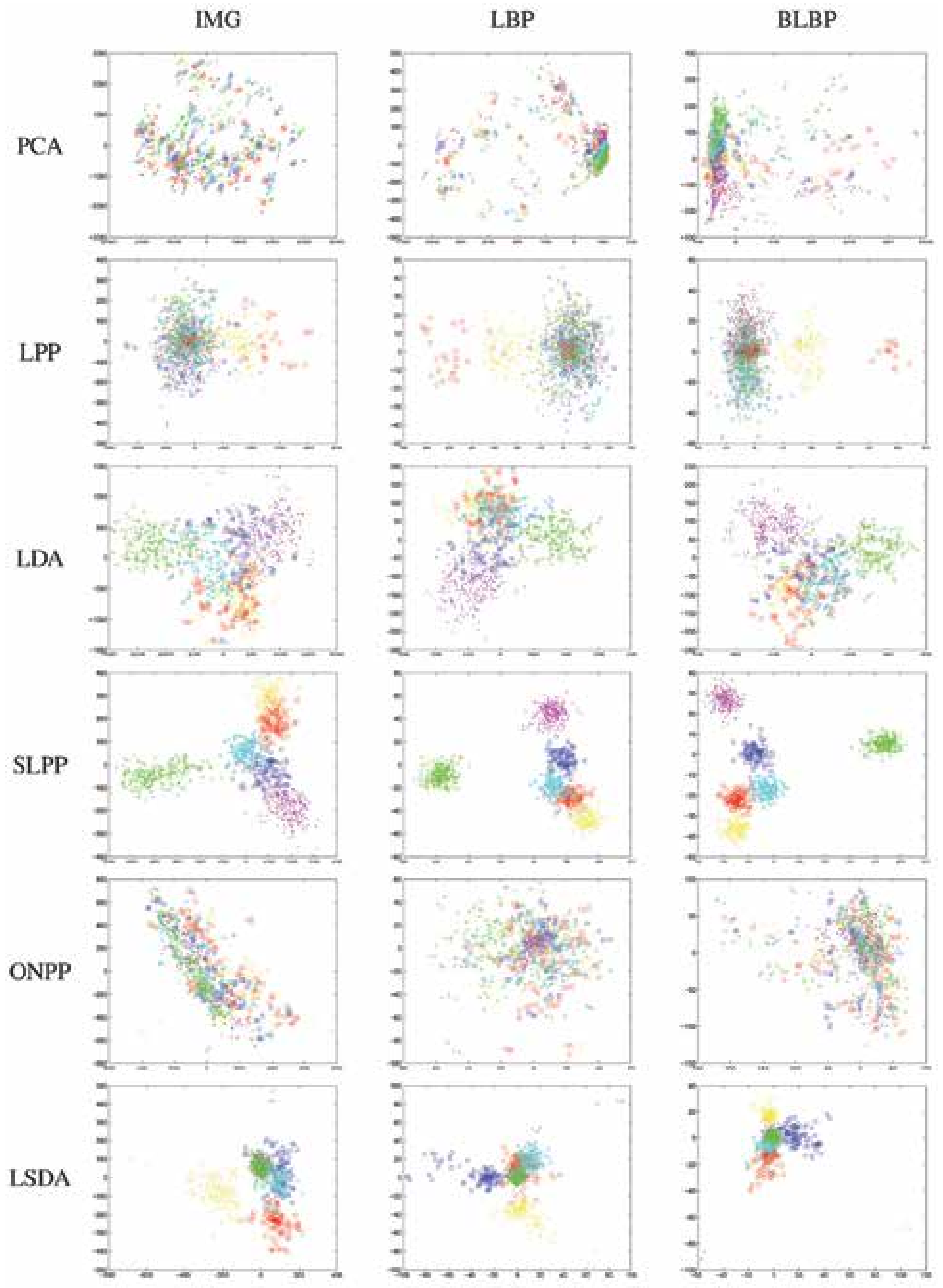


Fig. 6. (Best viewed in color) Images of data set **S1** are mapped into 2D embedding spaces. Different expressions are color coded as: Anger (red), Disgust (yellow), Fear (blue), Joy (magenta), Sadness (cyan), and Surprise (green).

projections of PCA are spread out since PCA aims at maximizing the variance. In the cases of LPP, although it preserves local neighborhood information, as expression images contain complex variations and significant overlapping among different classes, it is difficult for

LPP to yield meaningful projections in the absence of class information. For supervised methods, it is surprising to observe that different expressions are still heavily overlapped in the 2D subspace derived by ONPP. In contrast, the supervised methods LDA, SLPP and LSDA yield much meaningful projections since images of the same class are mapped close to each other. SLPP provides evidently best projections since different classes are well separated and the clusters appear cohesive. This is because SLPP preserves the locality and class information simultaneously in the projections. On the other hand, LDA discovers only the Euclidean structure therefore fails to capture accurately any underlying nonlinear manifold that expression images lie on, resulting in its discriminating power being limited. LSDA obtains better projections than LDA as the clusters of different expressions are more cohesive. On comparing facial representation, BLBP provides evidently the best performance with projected classes more cohesive and clearly separable in the SLPP subspace, while IMG is worst.

Fig. 7 shows the embedded OLPP subspace of data set **S1**.We can see that OLPP provides much similar projections to SLPP. The results obtained by SLPP and OLPP reflect human observation that Joy and Surprise can be clearly separated, but Anger, Disgust, Fear and Sadness are easily confused. This reenforces the findings in other published work (Tian, 2004; Cohen et al, 2003a).

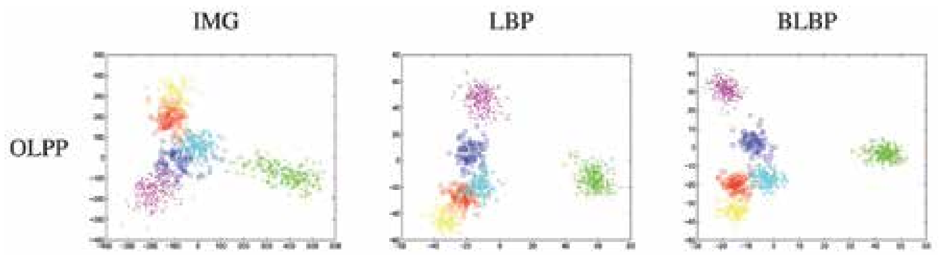


Fig. 7. (Best viewed in color) Images of data set **S1** are mapped into 2D embedding spaces of OLPP.

For a quantitative evaluation of the derived subspaces, following the methodology in (Li et al, 2003), we investigate the histogram distribution of within-class pattern distance and between-class pattern distance of different techniques. The former is the distance between expression patterns of the same expression class, while the latter is the distance between expression patterns belonging to different expression classes. Obviously, for a good representation, the within-class distance distribution should be dense, close to the origin, having a high peak value, and well-separated from the between-class distance distribution.We plot in Fig. 8 the results of different methods on **S1**. It is observed that SLPP consistently provides the best distributions for different facial representations, while those of PCA, LPP, and ONPP are worst. The average within-class distance d_w and between-class distance d_b are shown in Table 1. To ensure the distance measures from different methods are comparable, we compute a normalized difference between the within- and between-class distances of each method as $dif = \frac{d_b - d_w}{d_w}$, which can be regarded as a relative measure on how widely the within-class patterns are separated from the between-class patterns. A high value of this measure indicates success. It is evident in Table 1 that SLPP has the best separating power whilst PCA, LPP and ONPP are the poorest. The separating power of

LDA and LSDA is inferior to that of SLPP, but always outperform those of PCA, LPP, and ONPP. Both Fig. 8 and Table 1 reinforce the observation in Fig. 6.

	IMG			LBP			BLBP		
	d_w	d_b	dif	d_w	d_b	dif	d_w	d_b	dif
PCA	3921.8	4219.9	*0.0760*	542.7	591.3	*0.0897*	480.12	532.55	*0.1092*
LPP	216.3	241.4	*0.1164*	48.7	52.2	*0.0723*	42.304	46.016	*0.0877*
LDA	2195.3	2684.3	*0.2228*	288.5	377.1	*0.3071*	170.01	279.44	*0.6436*
SLPP	284.6	587.4	***1.0636***	18.1	86.4	***3.7741***	26.032	68.837	***1.6443***
ONPP	3069.8	3317.8	*0.0808*	633.3	673.5	*0.0636*	396.07	442.80	*0.1180*
LSDA	361.9	561.7	*0.5520*	44.3	76.1	*0.7189*	29.1	45.6	*0.5665*

Table 1. The average within-class and between-class distance and their normalization difference values on data set **S1**.

The 2D visualization of embedded subspaces of data set **S2** with different subspace techniques and facial representations is shown in Fig. 9. We observe similar results to those obtained in 6-class problem. SLPP outperforms the other methods in derive the meaningful projections. Different expressions are heavily overlapped in 2D subspaces generated by PCA, LPP, and ONPP, and the discriminating power of LDA is also limited.We further show in Fig. 10 the embedded OLPP subspace of data set **S2**, and also observe that OLPP provides much similar projections to SLPP. Notice that in the SLPP and OLPP subspaces, after including neutral faces, Anger, Disgust, Fear, Sadness, and Neutral are easily confused, while Joy and Surprise still can be clearly separated.

4.2.2 Comparative evaluation on expression recognition

To further compare these methods, we also performed facial expression recognition in the derived subspaces. We adopted the k nearest-neighbor classifier for its simplicity. The Euclidean metric was used as the distance measure. The number of nearest neighbors was set according to the size of the training set. To evaluate the algorithms' generalization ability, we adopted a 10-fold cross-validation test scheme.

That is, we divided the data set randomly into ten groups of roughly equal numbers of subjects, from which the data from nine groups were used for training and the left group was used for testing. The process was repeated ten times for each group in turn to be tested. We reported the average recognition results (with the standard deviation) here.

The recognition performance of subspace learning techniques varies with the dimensionality of subspace (note that the dimension of the reduced LDA subspace is at most c–1, where c is the number of classes). Moreover, the graph-based techniques rely on the parameter k, the number of nearest neighbors used when building the graph; how to set the parameter is still an open problem. In our cross-validation experiments, we tested different combinations of the parameter k with the subspace dimensionality, and the best performance obtained are shown in Tables 2 and 3. It is observed that the supervised approaches perform robustly better than the unsupervised methods. For unsupervised methods, PCA performs better than LPP, with all three facial representations. For supervised methods, it is evident that SLPP has a clear margin of superiority over LDA (12-38% better), ONPP (25-64% better), and LSDA (6-13% better). Both LSDA and LDA perform better than ONPP, and LSDA outperforms LDA. On comparing the standard deviation of 10-fold cross validation, SLPP

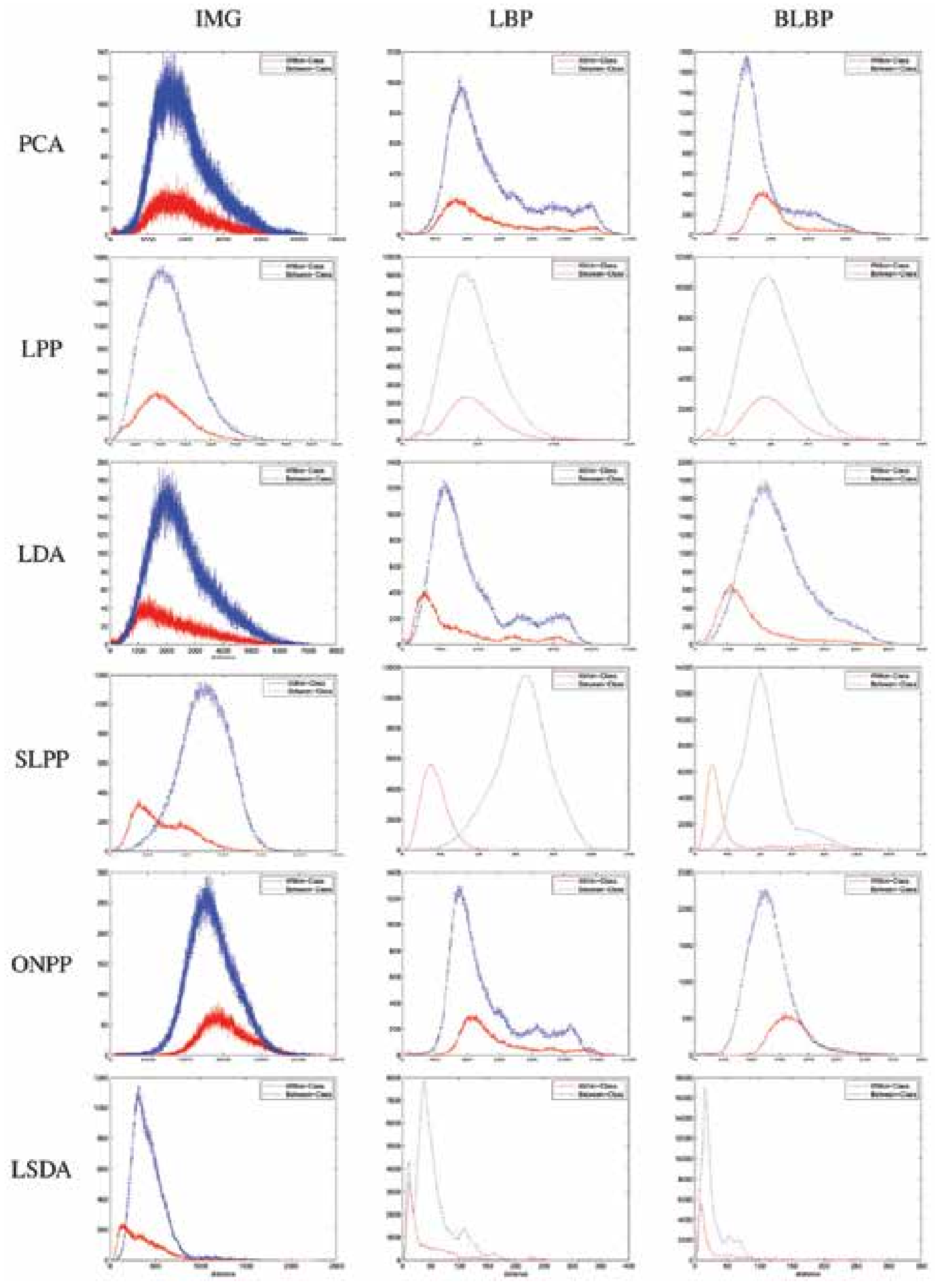


Fig. 8. (Best viewed in color) Histogram distribution of within-class pattern distance (solid red lines) and between-class pattern distances (dotted blue line) on data set **S1**

always produces the smallest deviation (one exception with IMG on **S2**). This demonstrates that SLPP is much more robust than other methods. The recognition results reinforce our early observations shown in Fig. 6, Fig. 8 and Table 1. To clearly compare recognition rates of different methods with different facial representations, we plot the bar graphes of

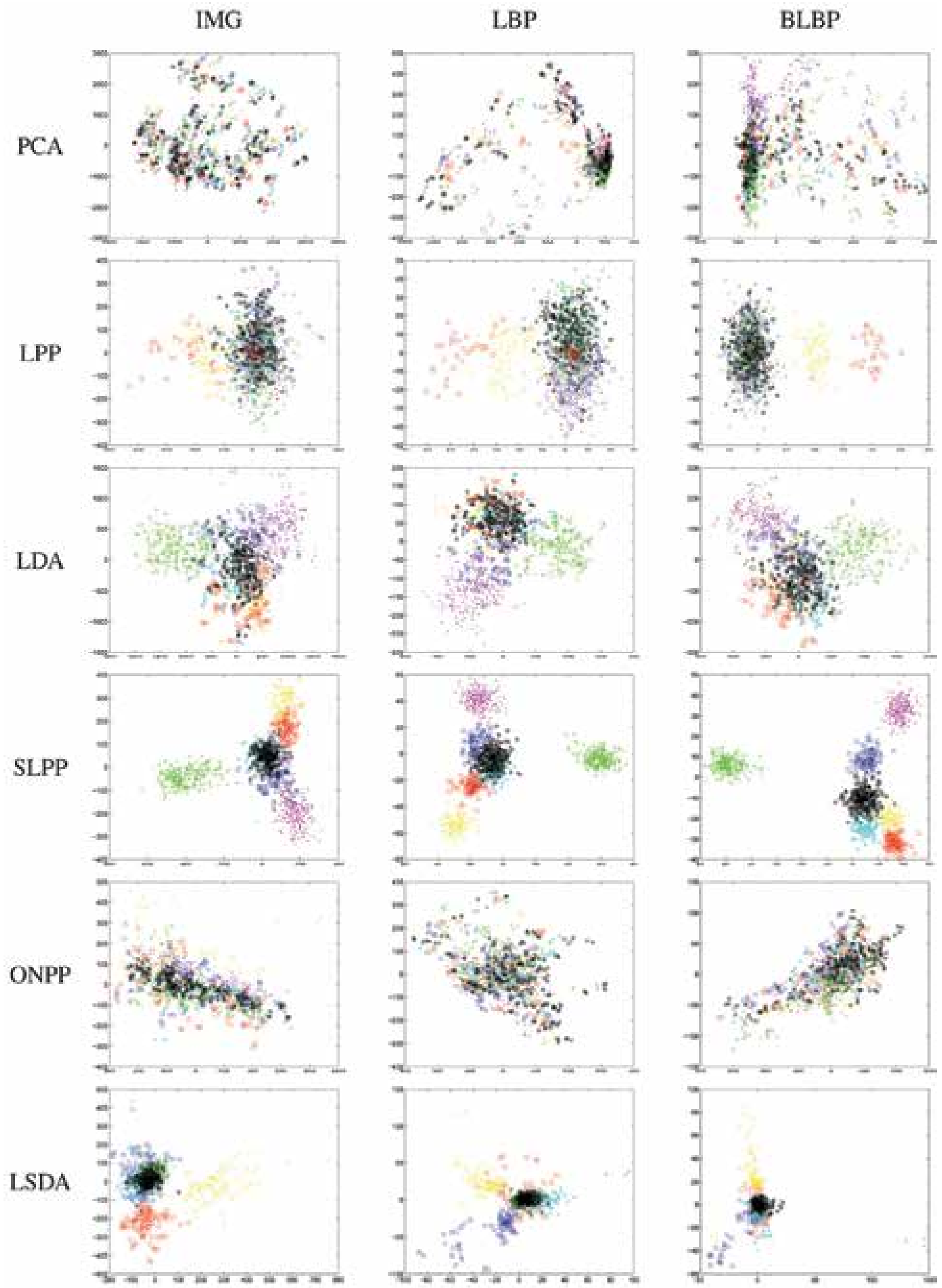


Fig. 9. (Best viewed in color) Images of data set **S2** are mapped into 2D embedding spaces. Neutral expression is color coded as black.

recognition rates in Fig. 11. On comparing feature representations, it is clearly observed that BLBP features perform consistently better than LBP and IMG features. LBP outperforms IMG most of the time except with LPP, IMG has a slight advantage over LBP.

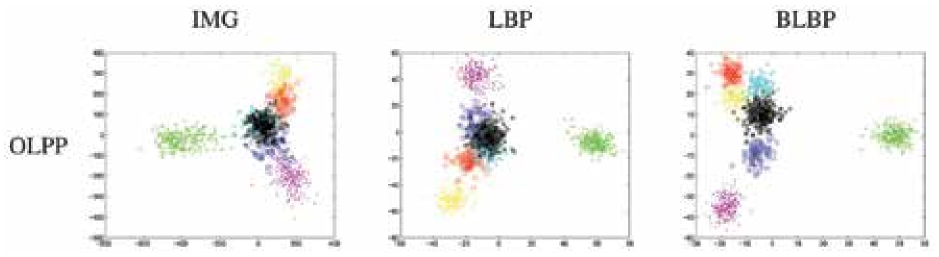


Fig. 10. (Best viewed in color) Images of data set **S2** are mapped into 2D embedding spaces of OLPP.

	PCA (%)	LPP (%)	LDA (%)	SLPP (%)	ONPP (%)	LSDA (%)
IMG	49.3±7.1	48.9±6.2	67.5±7.6	**86.4±5.0**	61.9±6.8	80.3±5.2
LBP	49.6±8.9	42.6±5.5	73.4±7.1	**90.4±2.9**	63.1±9.4	82.0±4.8
BLBP	74.9±9.1	48.9±9.0	84.2±6.7	**94.7±3.5**	75.9±6.9	87.2±5.4

Table 2. Averaged recognition rates (with the standard deviation) of 6-class facial expression recognition on data set **S1**.

	PCA (%)	LPP (%)	LDA (%)	SLPP (%)	ONPP (%)	LSDA (%)
IMG	41.4±7.0	37.9±8.4	59.5±**5.4**	**82.2**±6.2	50.0±6.8	77.3±5.7
LBP	50.6±6.4	36.6±5.0	67.9±5.5	**87.1±3.0**	57.8±9.4	76.9±6.5
BLBP	65.9±8.8	44.0±7.3	75.9±5.8	**92.0±3.9**	68.3±6.9	82.3±5.2

Table 3. Averaged recognition rates (with the standard deviation) of 7-class facial expression recognition on data set **S2**.

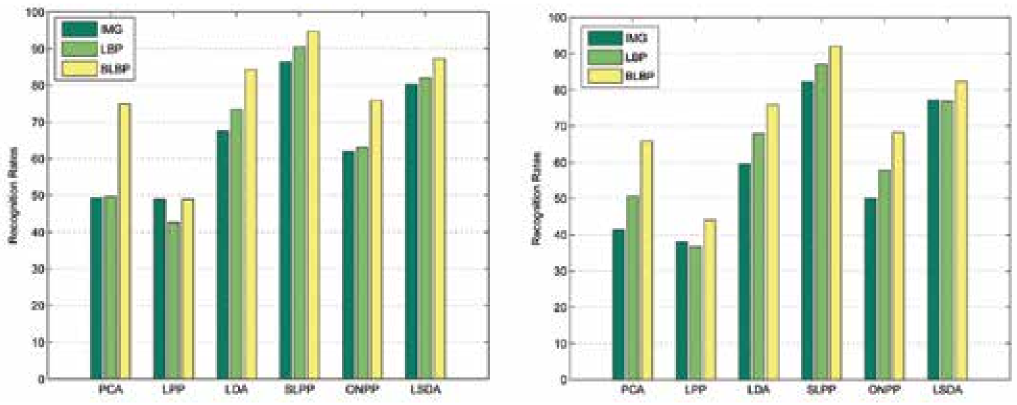


Fig. 11. Comparison of recognition rates using different subspace methods with different features. Left: data set **S1**; Right: data set **S2**.

We show in Fig. 12 the averaged recognition rates versus dimensionality reduction by different subspace schemes using BLBP features. As the dimension of the reduced subspace of LDA is at most c–1, we plot only the best achieved recognition rate by LDA across the various values of the dimension of subspace.We observe that SLPP outperforms other methods. The performance difference between SLPP and LDA is conspicuous when the dimension of subspace is small. But when the dimension increases, their performances become rather similar. The performances of PCA, LPP, and ONPP is inferior to that of LDA

consistently across all values of the subspace dimension. LSDA has similar trend with SLPP, but much worse performance. The performance of PCA and ONPP eventually become stable and similar when the dimension increase. On the other hand, the performance of LPP degrades when the dimension increases, and is the worst overall.
The best result of 94.7% in 6-class facial expression recognition, achieved by BLBP based SLPP, is to our best knowledge the best recognition rate reported so far on the database in the published literature. Previously Tian (2004) achieved 94% performance using Neural Networks with combined geometric features and Gaborwavelet features. With regard to 7-class facial expression recognition, BLBP based SLPP achieves the best performance of 92.0%, which is also very encouraging given that previously published 7-class recognition performance on this database were 81- 83% (Cohen et al, 2003a). The confusion matrix of 7-class facial expression in data set **S2** is shown in Table 4, which shows that most confusion occurs between Anger, Fear, Sadness, and Neutral.

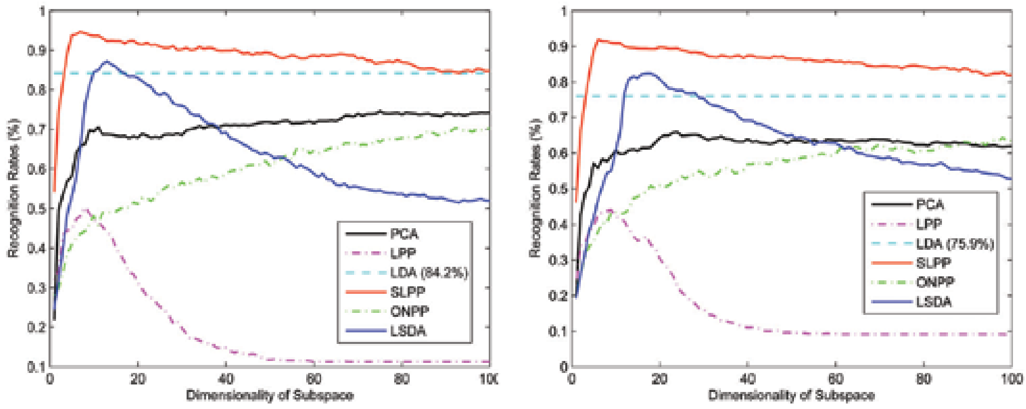


Fig. 12. (Best viewed in color) Averaged recognition accuracy versus dimensionality reduction (with BLBP features). Left: data set **S1**; Right: data set **S2**.

	Anger	Disgust	Fear	Joy	Sadness	Surprise	Neutral
Anger	85.2%	2.8%	0	0	7.4%	0	4.6%
Disgust	0	97.5%	2.5%	0	0	0	0
Fear	0	0	81.8%	11.1%	1.0%	1.0%	5.1%
Joy	0	0	0	97.5%	0	0	2.5%
Sadness	4.0%	0	0.8%	0	84.9%	0	10.3%
Surprise	0	0	2.7%	0	0	96.9%	0.4%
Neutral	0.8%	0	0.4%	2.8%	5.2%	0.4%	90.4%

Table 4. Confusion matrix of 7-class expression recognition on data set **S2**.

4.3 MMI database

The MMI Database (Pantic et al, 2005) includes more than 20 subjects of both sexes (44% female), ranging in age from 19 to 62, having either a European, Asian, or South American ethnic background. Subjects were instructed to display 79 series of facial expressions that included a single AU or a combination of AUs, or a prototypic emotion. Image sequences have neutral faces at the beginning and at the end, and were digitized into 720×576 pixels. Some sample images from the database are shown in Fig. 13. As can be seen, the subjects displayed facial expressions with and without glasses, which make facial expression analysis more difficult.

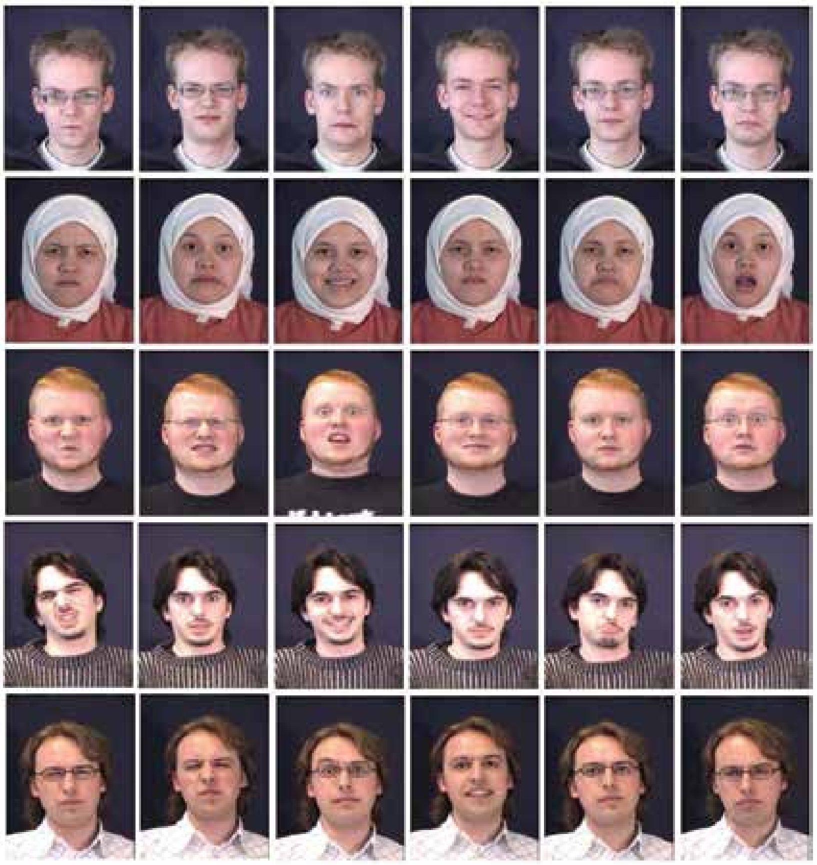


Fig. 13. The sample face expression images from the MMI Database.

In our experiments, 96 image sequences were selected from the MMI Database. The only selection criterion is that a sequence can be labeled as one of the six basic emotions. The sequences come from 20 subjects, with 1 to 6 emotions per subject.
The neutral face and three peak frames of each sequence (384 images in total) were used to form data set **S3** for 7-class expression analysis.

4.3.1 Comparative evaluation on subspace learning

The 2D visualization of embedded subspaces of data set **S3** is shown in Fig. 14. We observe similar results to those obtained in the Cohn-Kanade Database. SLPP consistently has the best performance, and different facial expressions are well clustered and represented in the derived 2D subspaces. In contrast, different expressions are heavily overlapped in 2D subspaces generated by PCA, LPP, and ONPP. The LDA and LSDA projections can not represent different facial expressions clearly, either. Notice also that in the SLPP subspaces, Anger, Disgust, Fear, Sadness, and Neutral are easily confused, while Joy and Surprise can be clearly separated.

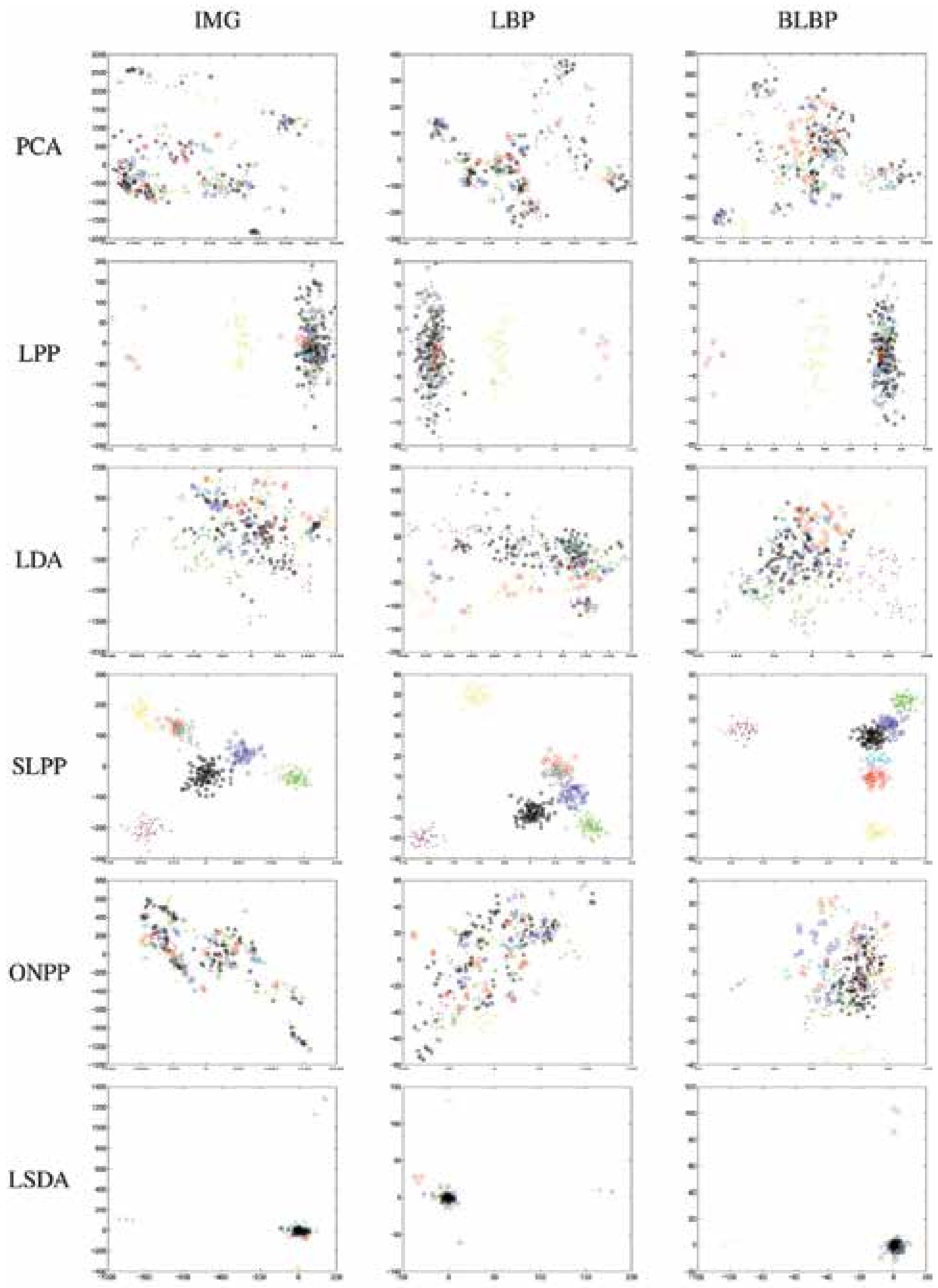


Fig. 14. (Best viewed in color) Images of data set **S3** are mapped into 2D embedding spaces.

4.3.2 Comparative evaluation on expression recognition

We report the average recognition results in Table 5. We observe similar recognition results to that in the Cohn-Kanade Database. With regard to unsupervised methods, PCA outperforms LPP with all three facial representations. For supervised methods, it is seen that

SLPP has a clear margin of superiority over LDA (19-50% better), ONPP (28-52% better) and LSDA (16-33% better). We further plot the bar graphes of recognition rates in the left side of Fig. 15, which demonstrate that BLBP features perform better than LBP and IMG features (except with LPP and LSDA), while LBP features have better or comparable performance with IMG features.

	PCA (%)	LPP (%)	LDA (%)	SLPP (%)	ONPP (%)	LSDA (%)
IMG	54.3±17.2	38.8±8.7	55.0±15.2	**81.2**±10.1	54.3±16.1	62.2±13.6
LBP	59.5±16.6	38.8±10.7	63.7±14.1	**80.7**±9.1	60.2±15.4	69.6±8.4
BLBP	60.2±15.3	35.1±7.3	71.4±11.7	**84.6**±8.8	66.2±12.1	63.5±8.7

Table 5. Averaged recognition rates (with the standard deviation) of 7-class facial expression recognition on data set **S3**.

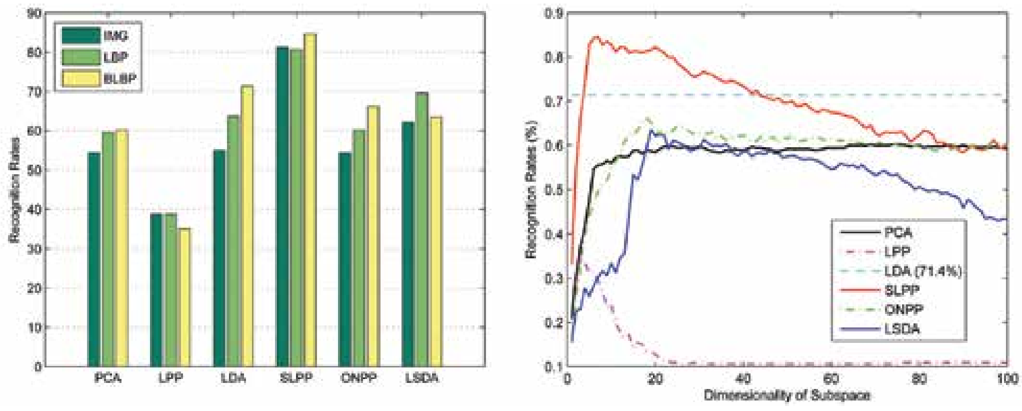


Fig. 15. (Best viewed in color) (*Left*) Comparison of recognition rates on data set **S3**; (*Right*) Averaged recognition accuracy versus dimensionality reduction (with BLBP features) on data set **S3**.

We show in the right side of Fig. 15 the averaged recognition rates with respect to the reduced dimension of different subspace techniques using BLBP features. We observe that SLPP performs much better than LDA when the reduced dimension is small, but their performance become similar, and SLPP is even inferior to LDA when the subspace dimension increases. LSDA provides consistently worse performance than LDA. The performances of PCA and ONPP are similar and stable consistently. In contrast, the performance of LPP degrades when the dimension increases, and is the worst overall. The plot in the right side of Fig. 15 is overall consistent with that of the Cohn-Kanade Database shown in Fig. 12.

4.4 JAFFE database

The JAFFE Database (Lyons et al, 1999) consists of 213 images of Japanese female facial expression. Ten expressers posed 3 or 4 examples for each of the seven basic expressions (six emotions plus neutral face). The image size is 256×256 pixels. Fig. 16 shows some sample images from the database.

In our experiments, all 213 images of the JAFFE database were used to form data set **S4** for 7-class facial expression analysis.

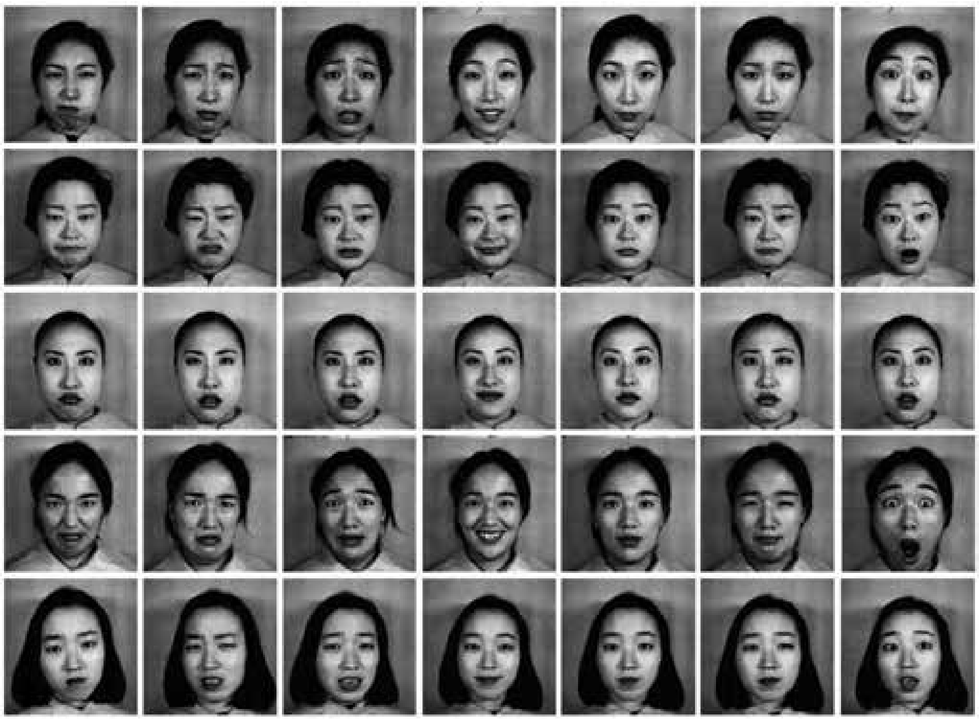


Fig. 16. The sample face expression images from the JAFFE Database.

4.4.1 Comparative evaluation on subspace learning

The 2D visualization of embedded subspaces of data set **S4** is shown in Fig. 17. Once again we observe that SLPP provides the best projections, in which different facial expressions are well separated. Similar to those in the Cohn-Kanade Database and the MMI Database, PCA, LPP, and ONPP do not provide meaningful projections, as different expressions are heavily overlapped in their 2D subspaces.

4.4.2 Comparative evaluation on expression recognition

The facial expression recognition results are reported in Table 6. We once again observe that SLPP outperform other subspace techniques with a clear margin of superiority, e.g., 14-38% better than LDA, 11-46% better than ONPP, and 22-38% better than LSDA. In this data set, LDA and ONPP have parallel performance, and are all superior to PCA and LPP. LPP still provides the worst results. The bar graphes of recognition rates is plotted in the left side of Fig. 18, which once again demonstrate that BLBP features provide the best performance, and LBP features perform better or comparably to IMG features.

Recognition performance on data set **S4** is much poorer than that on data sets **S1**, **S2**, and **S3**, and this is possibly because that there are fewer images in the data set resulting in a poor sampling of the underlying latent space. The effect of the small training set size may be also reflected on the standard deviation of 10-fold crossvalidation, as the standard deviations on data set **S4** are larger than those of data sets **S1**, **S2**, and **S1**, and the standard deviations of **S3** are larger than those of **S1** and **S2** as well. So the recognition performance of linear subspace methods on the small training sets is not robust and reliable.

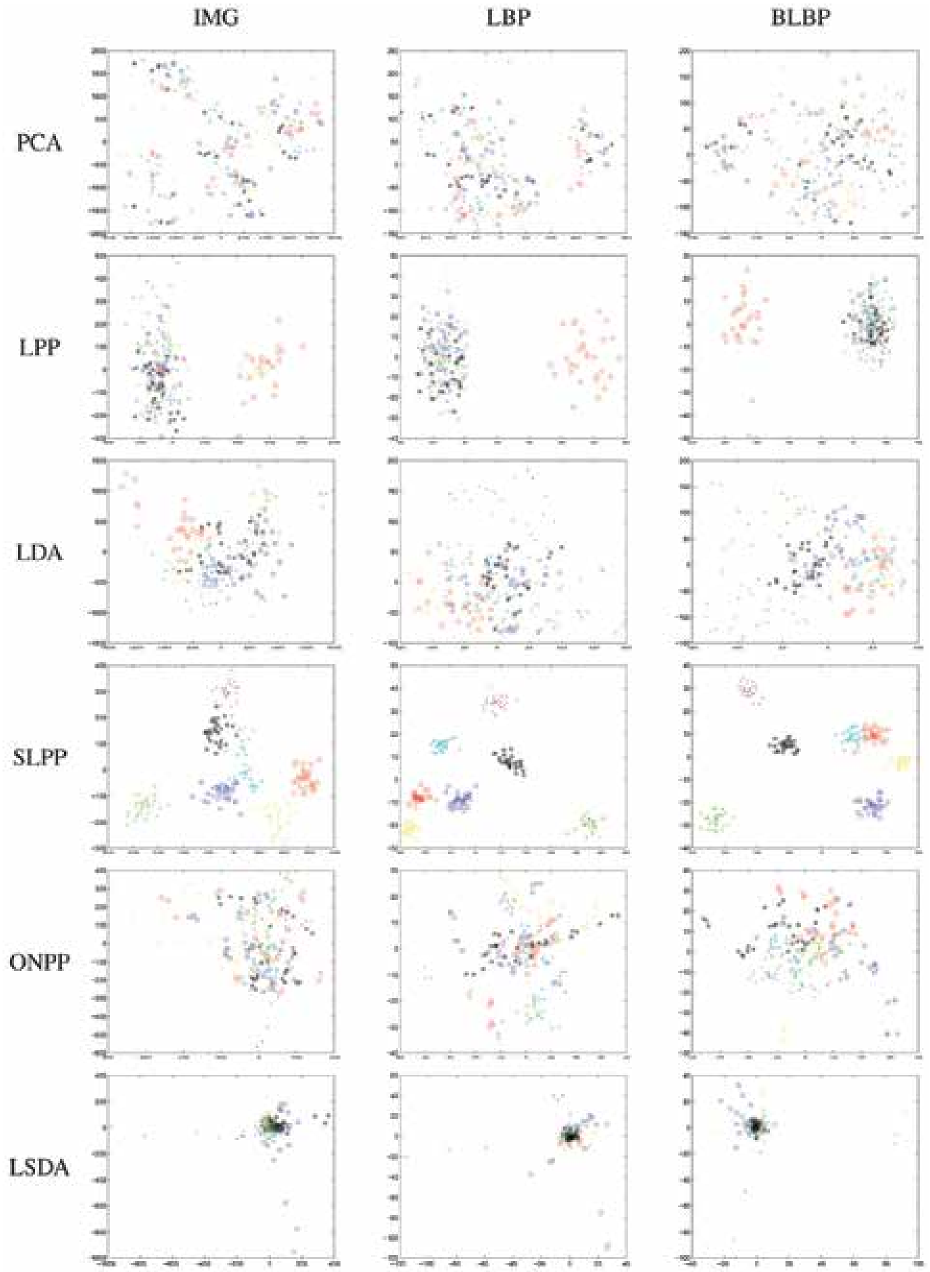


Fig. 17. (Best viewed in color) Images of data set **S4** are mapped into 2D embedding spaces.

We also plot in the right side of Fig. 18 the averaged recognition rates of different subspace techniques as the function of the reduced dimension when using BLBP features. It is observed the performances of SLPP and LDA become comparable when the reduced dimension increases. On the other hand, ONPP and PCA have similar performance. The plots for **S4** shows greater variations compared to those of **S1** and **S2** (shown in Fig. 12). This may also be due to the small size of the training set.

	PCA (%)	LPP (%)	LDA (%)	SLPP (%)	ONPP (%)	LSDA (%)
IMG	39.5±6.3	31.6±14.5	51.2±11.0	**69.2**±15.7	48.5±12.2	56.3±22.8
LBP	51.8±13.2	31.0±10.5	56.4±16.1	**72.4**±18.1	54.2±16.0	52.3±16.6
BLBP	62.6±17.5	32.4±9.8	65.4±15.5	**74.2**±16.1	66.8±13.7	54.9±14.5

Table 6. Averaged recognition rates (with the standard deviation) of 7-class facial expression recognition on data set **S4**.

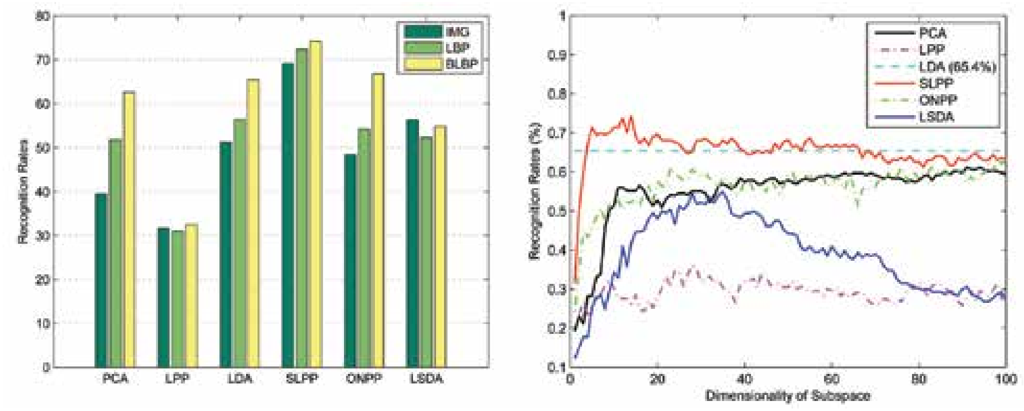


Fig. 18. (Best viewed in color) (*Left*) Comparison of recognition rates on data set **S4**; (*Right*) Averaged recognition accuracy versus dimensionality reduction (with BLBP features) on data set **S4**.

5. Conclusions and discussions

In this chapter, we review and evaluate a number of linear subspace methods in the context of automatic facial expression analysis, which included recently proposed LPP, SLPP, OLPP, ONPP, LSDA, and the traditional PCA and LDA. These techniques are compared using different facial feature representations on several databases. Our experiments demonstrate that the supervised LPP performs best in modeling the underlying facial expression subspace resulting in the best expression recognition performance. We believe that this study is helpful and necessary for further research in linear subspace methods and facial expression analysis.

It is believed that images of facial expressions lies on a non-linear low-dimensional manifold. Therefore, although linear subspace learning methods have been shown to be effective , non-linear manifold learning could potentially perform better for modeling facial expression space. For future work, we would expect to see research on discriminant non-linear manifold learning techniques for facial expression analysis.

6. Acknowledgments

We sincerely thank Prof. Jeffery Cohn, Dr. Maja Pantic, and Dr. Michael J. Lyons for granting access to the Cohn-Kanade database, the MMI database, and the JAFFE database respectively. We also thank E. Kokiopoulou for providing the source code of ONPP.

7. References

Ahonen T, Hadid A, Pietik¨ainen M (2004) Face recognition with local binary patterns. In: European Conference on Computer Vision (ECCV), pp 469–481

Ambady N, Rosenthal R (1992) Thin slices of expressive behaviour as predictors of interpersonal consequences: A meta-analysis. Psychological Bulletin 111(2):256–274

Balasubramanian VN, Krishna S, Panchanathan S (2008) Person-independent head pose estimation using biased manifold embedding. EURASIP Journal of Advances in Signal Processing 8(1)

Bartlett M, Littlewort G, Fasel I, Movellan R (2003) Real time face detection and facial expression recognition: Development and application to human computer interaction. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop, pp 53–53

Bartlett MS, Movellan JR, Sejnowski TJ (2002) Face recognition by independent component analysis. IEEE Transactions on Neural Networks 13(6):1450–1464

Bartlett MS, Littlewort G, Frank M, Lainscsek C, Fasel I, Movellan J (2005) Recognizing facial expression: Machine learning and application to spotaneous behavior. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 568–573

Bassili JN (1979) Emotion recognition: The role of facial movement and the relative importance of upper and lower area of the face. Journal of Personality and Social Psychology 37(11):2049–2058

Belhumeur PN, Hespanha JP, Kriegman DJ (1997) Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7):711–720

Belkin M, Niyogi P (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems (NIPS), pp 585–591

Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15(6):1373–1396

Cai D, He X, Han J, Zhang H (2006) Orthogonal laplacianfaces for face recognition. IEEE Transactions on Image Processing 15(11):3608–3614

Cai D, He X, Zhou K, Han J, Bao H (2007) Locality sensitive discriminant analysis. In: International Joint Conference on Artificial Intelligence (IJCAI), pp 708–713

Chang Y, Hu C, TurkM(2003) Mainfold of facial expression. In: IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG), pp 28–35

Chang Y, Hu C, Turk M (2004) Probabilistic expression analysis on manifolds. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 520– 527

Cohen I, Sebe N, Cozman F, Cirelo M, Huang T (2003a) Learning baysian network classifiers for facial expression recognition using both labeled and unlabeled data. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Cohen I, Sebe N, Garg A, Chen L, Huang TS (2003b) Facial expression recognition from video sequences: Temporal and static modeling. Computer Vision and Image Understanding 91:160–187

Cohen I, Cozman F, Sebe N, Cirelo M, Huang TS (2004) Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(12):1553–1566

Darwin C (1872) The Expression of the Emotions in Man and Animals. John Murray, London

Donato G, Bartlett M, Hager J, Ekman P, Sejnowski T (1999) Classifying facial actions. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(10):974–989

Ekman P, Friesen W (1976) Pictures of Facial Affect. Consulting Psychologists

Ekman P, Friesen WV, Hager JC (2002) The Facial Action Coding System: A Technique for the Measurement of Facial Movement. San Francisco: Consulting Psychologist

Fasel B, Luettin J (2003) Automatic facial expression analysis: a survey. Pattern Recognition 36:259–275

Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1):119–139

Fu Y, Huang TS (2006) Graph embedded analysis for head pose estimation. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp 3–8

Gritti T, Shan C, Jeanne V, Braspenning R (2008) Local features based facial expression recognition with face registration errors. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), Amsterdam, The Netherlands

He X, Niyogi P (2003) Locality preserving projections. In: Advances in Neural Information Processing Systems (NIPS)

Jones M, Viola P (2003) Fast multi-view face detection. Tech. Rep. 096, MERL

Kaliouby RE, Robinson P (2004) Real-time inference of complex mental states from facial expressions and head gestures. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop, pp 154–154

Kanade T, Cohn J, Tian Y (2000) Comprehensive database for facial expression analysis. In: IEEE International Conference on Automatic Face& Gesture Recognition (FG), pp 46–53

Kokiopoulou E, Saad Y (2005) Orthogonal neighborhood preserving projections. In: IEEE International Conference on Data Mining (ICDM)

Kokiopoulou E, Saad Y (2007) Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(12):2143–2156

Li Y, Gong S, Liddell H (2003) Constructing facial identity surfaces for recognition. International Journal of Computer Vision 53(1):71–92

Lienhart R, Kuranov D, Pisarevsky V (2003) Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In: DAGM'03, 25th Pattern Recognition Symposium, Madgeburg, Germany, pp 297–304

Littlewort G, Bartlett M, Fasel I, Susskind J, Movellan J (2006) Dynamics of facial expression extracted automatically from video. Image and Vision Computing 24(6):615–625

Lyons MJ, Budynek J, Akamatsu S (1999) Automatic classification of single facial images. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(12):1357–1362

Mehrabian A (1968) Communication without words. Psychology Today 2(4):53–56

Murphy-Chutorian E, Trivedi M (2008) Head pose estimation in computer vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence

Ojala T, Pietikäinen M, Mäenpää T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971–987

Oliver N, Pentland A, Berard F (2000) Lafter: a real-time face and lips tracker with facial expression recognition. Pattern Recognition 33:1369–1382

Pantic M, Bartlett MS (2007) Machine analysis of facial expressions. In: Kurihara K (ed) Face Recognition, Advanced Robotics Systems, Vienna, Austria, pp 377– 416

Pantic M, Rothkrantz L (2000a) Automatic analysis of facial expressions: the state of art. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(12):1424–1445

Pantic M, Rothkrantz L (2000b) Expert system for automatic analysis of facial expression. Image and Vision Computing 18(11):881–905

Pantic M, Rothkrantz L (2003) Toward an affect-sensitive multimodal humancomputer interaction. In: Proceeding of the IEEE, vol 91, pp 1370–1390

Pantic M, Rothkrantz LJM (2004) Facial action recognition for facial expression analysis from static face images. IEEE Transactions on Systems, Man, and Cybernetics 34(3):1449–1461

Pantic M, Valstar M, Rademaker R, Maat L (2005) Web-based database for facial expression analysis. In: IEEE International Conference on Multimedia and Expo (ICME), pp 317–321

Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323–2326

Russell JA (1994) Is there univeral recognition of emotion from facial expression. Psychological Bulletin 115(1):102–141

Saul LK, Roweis ST (2003) Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research 4:119–155

Schapire RE, Singer Y (1999) Improved boosting algorithms using confidencerated predictions. Maching Learning 37(3):297–336

Shan C (2007) Inferring facial and body language. PhD thesis, Queen Mary, University of London

Shan C, Gritti T (2008) Learning discriminative lbp-histogram bins for facial expression recognition. In: British Machine Vision Conference (BMVC), Leeds, UK

Shan C, Gong S, McOwan PW (2005a) Appearance manifold of facial expression. In: Sebe N, Lew MS, Huang TS (eds) Computer Vision in Human-Computer Interaction, Lecture Notes in Computer Science, vol 3723, Springer, pp 221–230

Shan C, Gong S, McOwan PW (2005b) Conditional mutual information based boosting for facial expression recognition. In: British Machine Vision Conference (BMVC), Oxford, UK, vol 1, pp 399–408

Shan C, Gong S, McOwan PW (2005c) Robust facial expression recognition using local binary patterns. In: IEEE International Conference on Image Processing (ICIP), Genoa, Italy, vol 2, pp 370–373

Shan C, Gong S, McOwan PW (2006a) A comprehensive empirical study on linear subspace methods for facial expression analysis. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop, New York, USA, pp 153–158

Shan C, Gong S, McOwan PW (2006b) Dynamic facial expression recognition using a bayesian temporal manifold model. In: British Machine Vision Conference (BMVC), Edinburgh, UK, vol 1, pp 297–306

Suwa M, Sugie N, Fujimora K (1978) A preliminary note on pattern recognition of human emotional expression. In: International Joint Conference on Pattern Recognition, pp 408–410

Tenenbaum JB, Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290:2319–2323

Tian Y (2004) Evaluation of face resolution for expression analysis. In: International Workshop on Face Processing in Video, pp 82–82

Tian Y, Kanade T, Cohn J (2001) Recognizing action units for facial expression analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(2):97– 115

Tian Y, Kanade T, Cohn J (2005) Handbook of Face Recognition, Springer, chap 11. Facial Expression Analysis

Tong Y, Liao W, Ji Q (2006) Inferring facial action units with causal relations. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1623–1630

Turk M, Pentland AP (1991) Face recognition using eigenfaces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Valstar M, Pantic M (2006) Fully automatic facial action unit detection and temporal analysis. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop, p 149

Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 511–518

Wang Y, Ai H, Wu B, Huang C (2004) Real time facial expression recognition with adaboost. In: International Conference on Pattern Recognition (ICPR), pp 926– 929

Whitehill J, Omlin C (2006) Harr features for facs au recognition. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp 97–101

Yang MH, Kriegman DJ, Ahuja N (2002) Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(1):34–58

Yeasin M, Bullot B, Sharma R (2004) From facial expression to level of interests: A spatio-temporal approach. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 922–927

Zhang Y, Ji Q (2005) Active and dynamic information fusion for facial expression understanding from image sequences. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(5):1–16

Zhang Z, Lyons MJ, Schuster M, Akamatsu S (1998) Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp 454–461

Resampling Methods for Unsupervised Learning from Sample Data

Ulrich Möller
Leibniz Institute for Natural Product Research and Infection Biology – Hans Knöll Institute, Germany

1. Introduction

Two important tasks of machine learning are the *statistical learning* from sample data (SL) and the *unsupervised learning* from unlabelled data (UL) (Hastie et al., 2001; Theodoridis & Koutroumbas, 2006). The synthesis of the two parts – the *unsupervised statistical learning* (USL) – is frequently used in the cyclic process of inductive and deductive scientific inference. This applies especially to those fields of science where promising, testable hypotheses are unlikely to be obtained based on manual work, the use of human senses or intuition. Instead, huge and complex experimental data have to be analyzed by using machine learning (USL) methods to generate valuable hypotheses. A typical example is the field of functional genomics (Kell & Oliver, 2004).
When machine learning methods are used for the generation of hypotheses, human intelligence is replaced by artificial intelligence and the proper functioning of is this type of 'intelligence' has to be validated. This chapter is focused on the validation of *cluster analysis* which is an important element of USL.
It is assumed that the data set is a sample from a mixture population which is statistically modeled as a mixture distribution. Cluster analysis is used to 'learn' the number and characteristics of the components of the mixture distribution (Hastie et al., 2001). For this purpose, similar elements of the sample are assigned to groups (clusters).
Ideally, a cluster represents all of the elements drawn from one population of the mixture. However, clustering results often contain errors due to lacking robustness of the algorithms. Rather different partitions may result even for samples with small differences. That is, the obtained clusters have a random character. In this case, the generalization from clusters of a sample to the underlying populations is inappropriate. If a hypothesis derived from such clustering results is used to design an experiment, the outcome of this experiment will hardly lead to a model with a high predictive power. Thus, a new study has to be performed to find a better hypothesis. Even a single cycle of hypothesis generation and hypothesis testing can be time-consuming and expensive (e.g., a gene expression study in cancer research, with 200 patients, lasts more than a year and costs more than 100.000 dollars). Therefore, it is desirable to increase the efficiency and effectiveness of the scientific progress by using suitable validation tools.
An approach for the statistical validation of clustering results is data resampling (Lunneborg, 2000). It can be seen as a special Monte Carlo method that is, as a method for

finding solutions to statistical problems by simulation (Borgelt & Kruse, 2006). The choice of a *suitable* resampling method for any cluster validation task is not trivial. On the one hand, such a method is expected to simulate random samples that have the same structure that underlies the original sample – even though the true structure is unknown. On the other hand, it is undesired that the method introduces any additional structure into the simulated data, because this kind of error can not be recognized from the clustering results in the absence of the ground truth.

Once, clustering results (partitions) have been generated for a set of resamples, three steps are usually performed. i) The stability of the partitions under the influence of resampling is calculated. When desired, stability scores can be obtained also for single clusters and individual assignments of data points to clusters. ii) A *consensus partition* is determined that best possible represents the characteristics which are common to the resample partitions. iii) The number of clusters is estimated, typically based on the maximization of a partition stability score. For methods that can be used to perform the steps i) to iii) see, for example, (Strehl & Gosh, 2002; Topchy et al., 2005; Fred & Jain, 2006 and Ayad & Kamel, 2008).

Resampling-based cluster validation is not yet common standard. In many software tools for cluster analysis, resampling methods are missing. Some new methods were published only recently. The choice of the appropriate resampling technique depends on the data properties, the goal and constraints of the study and on the clustering methods used. The purpose of this contribution is to review available techniques, to summarize existing benchmark results and to give recommendations for the selection and use of the methods. Furthermore, a new method called *nearest neighbor resampling* is presented.

In statistics resampling schemes are subdivided into parametric and non-parametric methods. The use of parametric methods for cluster validation will be briefly characterized in section 2. In section 3 non-parametric methods will be reviewed. Section 4 is a summary of benchmarking tests of different resampling techniques. Section 5 refers to results of the new resampling method previously described in section 3.5. Finally, section 6 contains a discussion of the described methods and conclusions for their future application.

2. Parametric resampling

Parametric resampling is also known as *parametric bootstrapping*. Methods of this type are used to fit a parametric model to the data. That is, the hypothesis is made that the data follow a theoretical distribution and certain parameters of this distribution (mean, variance etc.) are estimated. Then resample data sets are sampled from the distribution with the parameter values set to the obtained estimates. In cluster analysis a mixture distribution $P = \Sigma_i \varepsilon_i P_i$ is assumed, where P_i, $i = 1,\ldots,C$, are the C distributions generating C "true" clusters respectively, and ε_i is the probability that a sample point from P_i is drawn.

In principle, this approach has attractive properties. Examples for the validation of clustering results obtained from gene expression data are contained in (McLachlan & Khan, 2004). However, there exist also arguments against the use of parametric resampling for cluster validation. One argument concerns the lack of justification for the (more or less arbitrary) selection of a particular theoretical distribution as a model for real data with an unknown distribution (Yu et al. 1999; Lunneborg, 2000). Hennig (2007) argued that parametric bootstrapping does not suggest itself for the aim of cluster validation, because parametric methods discover structures generated by the assumed model much better than patterns in real data for which the model does not hold. This could lead to overoptimistic assessments of the stability of clustering structures. If the original sample has clearly more dimensions than data points, model fitting may be impaired by the "curse of dimensionality". Further arguments can be found in (Tseng & Wong, 2005). In the sequel, we consider non-parametric resampling methods that may be used in cluster analysis.

3. Non-parametric resampling

3.1 Sampling from a sample

Several methods can be referred to as re-sampling in the literal sense according to the common (non-statistical) definition of the word *sample* [1]. In such methods the data points of a resample are drawn from the set of data points contained in the original sample.

Bootstrapping. The non-parametric version of bootstrapping is usually described as "drawing with replacement". That is, each bootstrap sample is obtained by drawing N data points randomly and with replacement from the original sample, where N is the number of data points in this sample. If the population size N_p is finite and relatively small compared to N, (i.e., N_P / N < 20), another procedure is conventionally used (see Lunneborg, 2000). This procedure guarantees that the empirical distribution of the union of all bootstrap resamples agrees accurately with the empirical distribution of the original sample. In any case some original data points are likely represented more than once in a bootstrap sample, while accordingly, other original points are missing in the resample.

It has been shown that for increasing values of N, the percentage of original data which are not contained in a bootstrap sample converges to about 37%. If this information loss is considered to large for an adequate recognition of the data structure, the bootstrap scheme could be applied to M randomly selected points of the sample X ($M < N$), while the resample is completed by the $N–M$ points of X not used for the bootstrapping. This modification would allow to control the degree of information loss associated with the bootstrap scheme (Möller & Radke, 2006a). Moreover, this resampling version could be performed by using random numbers M_r for the generation of $r = 1, 2, \ldots$ bootstrap samples with reasonable boundaries of the interval from which the values M_r are drawn. This selection could make the results less depending on the heuristic choice of parameter M.

Subsampling. The original data set X is used to draw random subsets $Y_r \subset X$, $r = 1, 2, \ldots$ The size of a drawn subset, $S = \text{card}(Y_r)$, is a control parameter. Usually, S is fixed for all subsamples to be used in an application. If S is not much smaller than the original sample size N, clustering results of different subsamples may be very similar and not informative. The choice of S clearly smaller than N can be recommended if the information retained on average in a subsample is sufficient to obtain reasonable estimates of the unknown underlying distribution. Resampling-based clustering methods have been introduced including the subsampling of 70% (Tseng & Wong, 2005), 80% (Monti et al., 2003) and 90% (Fred & Jain, 2006) of the data. It may not be easy to select an optimal subsample size in a particular application. To avoid an inappropriate choice for this parameter, the subsample size could be varied from subsample to subsample. For example, the subsample size is uniformly drawn from an interval that represents 75-90% of the size of the original sample.

[1] A sample of things is a number of them that are chosen at random out of a larger group and then used to test ideas or to provide information about the whole group (Collins Cobuild Dictionary, 1987).

An alternative way, without an explicit specification of the subsampling size, would be to generate a bootstrap sample and to discard the identically replicated points (Hennig, 2007).
Subdivision. The original sample X is split into two disjoint subsets $Y \cup Z = X$. Clustering is used to generate the partitions π_Y and π_Z, from Y and Z, respectively. In addition, a classifier C_Y is build from the subset Y and the label set π_Y. Then C_Y is applied to the subset Z providing the partition π_{YZ}. Finally, the predictability of π_Z based on π_{YZ} is assessed. For the success of this strategy it has to be ensured that in general each subset Y and Z contain sufficient information about the underlying distribution necessary to infer a reasonable model from the data. Dudoit and Fridlyand (2003) presented an example, where the 'training' set Y and the 'test' set Z consist respectively of 2/3 and 1/3 of the original sample.

3.2 Jittering

Real data samples contain random measurement errors. Even if the same objects were observed multiple times under the same experimental conditions, the data are likely to be different. These differences can be simulated by generating copies of the original sample and adding random values to each of these data sets. The normal distribution with zero mean is traditionally used for this purpose. If estimates of the measurement error exist, these information can be utilized to define the parameters of the error distribution. Otherwise, heuristic rules can be applied.
Hennig (2007) defined such a resampling scheme as follows. 1) For all p dimensions of the original sample data $X = (x_1,\ldots, x_N)$, compute the $N-1$ differences d_{ij} between neighboring data values in dimension p: for $i = 1,\ldots, N-1$, $j = 1,\ldots, p$, d_{ij} is the difference between the $(i+1)$-th and the i-th order statistic of the j-th dimension. For $j = 1,\ldots, p$, let q_j be the empirical quantile of the d_{ij}, where q is a tuning constant. 2) Draw noise e_n, $n = 1,\ldots, N$, independent and identically distributed from a normal distribution with a zero mean and a diagonal matrix as covariance matrix with diagonal elements $\sigma_1^2 = q_1^2,\ldots, \sigma_p^2 = q_p^2$ and compute the resample points $y_n = x_n + e_n$ for $n = 1,\ldots, N$. (For an example see section 4).

3.3 Combination of bootstrapping and jittering

When using the (non-parametric) bootstrapping scheme, about one third of the resample points will be identical replicates of original sample points. Each group of such identical points could be seen as a mini-cluster. The occurrence of these artificial clusters, generated by a statistical analysis tool, may induce inappropriate models of the true data structure. In particular, when clustering the resample data, the artificially replicated data points may be misinterpreted as true clusters (Monti et al., 2003). Moreover, for some implementations of clustering and multidimensional scaling methods the identical bootstrap replicates may cause numerical problems. Hennig (2007) proposed the combination of bootstrapping and jittering as a way to avoid or to reduce these problems.

3.4 Perturbation

Data sets for applications of statistical machine learning are usually generated with a precision that is high enough to measure intra-population variability. Therefore, any data point of a sample is likely to be different from any data point of another (disjoint) sample – even if the measurement error was zero. This type of inter-sample differences is not realistically simulated when using the above non-parametric methods. (*Sampling from a sample* provides highly overlapping data sets that all consist of random selections from the *same* set of original points, while *jittering* leads to data sets that simulate differences comparable to those caused by measurement errors.) Another resampling strategy may be desired for a better (non-parametric) simulation of inter-sample differences due to intra-population variability. Estimates of intra-population variability that could be used for such a simulation are usually unavailable prior to cluster analysis. Under these circumstances, a simple simulation is the addition of random values onto the data. Here this approach is called 'perturbation'.

Let $X \in \Re^{N \times p}$ be the original p-dimensional sample consisting of N data points. Then for $r = 1, 2, \ldots$, resample r is obtained as follows. $Y_r = X + \xi_r$, where $\xi_r \in \Re^{N \times p}$ is a sample of size N from a p-dimensional distribution. The parameters of this distribution, such as variance, can be specified based on an estimate obtained from the original sample. For example, the random variable ξ may be selected to have a normal distribution with zero mean vector and $c \cdot \sigma \in \Re^p$, where c denotes a constant, $\sigma = (\sigma_1, \ldots, \sigma_p)$ is an empirical estimate of the variability of the data. Bittner et al. (1999), chose $c = 0.15$ and σ being the median standard deviation of the entire sample. Möller and Radke (2006a) used several values of c equal to 0.01, 0.05 and 0.1, where σ represented the standard deviation from the grand mean of the data.

Perturbation and *jittering* are conceptually similar resampling techniques. However, their implementation may differ quantitatively in the values of statistical parameters used to simulate intra-population variability and measurement error based on external knowledge, estimates or assumptions.

3.5 Nearest neighbor resampling

The *perturbation* technique has two shortcomings in a cluster validation study. First, the method will induce inappropriate inter-resample differences if the true intra-population variability *differs* between several populations of the mixture population. The reason is that the random values used to perturb every data point are drawn from the *same* distribution. Therefore, the data points originally drawn from some populations are perturbed too strongly or too weakly or both types of error may occur simultaneously. Second, even if the intra-population variability is constant across all populations within the mixture, it is difficult to adjust the parameter(s) of the distribution used for drawing the random values. An overestimation of the proper perturbation strength would have the consequence that true data structures which are present in the original sample may not be retained in any resample. Otherwise, an underestimation of the perturbation strength would lead to very similar resamples and spurious, high cluster stability. To avoid false interpretations of a perturbation-based clustering study, it may be appropriate to repeat the analysis with different values of the perturbation strength (e.g., Möller & Radke, 2006a). A non-parametric resampling approach where the choice of the perturbation strength is less critical is nearest neighbor resampling (NNR).

The idea behind NNR can be explained as follows. A high intra-population variability is characterized by a wide distribution and a low probability of drawing a point from the respective part of the hyperspace. Accordingly, the distances between sample points in this part of the hyperspace are high. For low intra-population variability the opposite is true. Clearly, if two or more populations of a mixture population have overlapping distributions, the total probability is increased and sample points will have decreased inter-point distances compared to those obtained from any single population. The relationship between population variability and inter-point distances can be utilized to simulate random samples, where the advantages of a perturbation approach are utilized and knowledge, estimates or assumptions about the distributions of existing populations are not required.

Here we consider the following strategy for NNR. 1) For each original sample point x_n, $n = 1,\ldots, N$, an estimate of the inter-point distances in the neighborhood of x_n is obtained. This neighborhood is defined by the k nearest neighbors of x_n according to a user-selected metric. 2) The direction vector for the perturbation of x_{nr} with respect to x_n is selected. 3) Resample point x_{nr} is generated by adding a random vector to x_n with the direction as selected in step two and the vector length being a function of the estimated inter-point distances in the neighborhood of x_n. The rationale underlying the choice of a k-NN approach is the same as in supervised learning. Most of the k nearest neighbors of data point x_n are assumed to belong to the same class (population) as x_n. Therefore, the neighboring points of x_n are assumed to provide an estimate of intra-population variability. Below, two versions of NNR are described.

Nearest neighbor resampling 1 (NNR1). (Möller & Radke, 2006b)

0. Let $X = (x_1,\ldots, x_N) \subset \Re^p$ be the original sample. Chose $k \geq 2$ and a metric for calculating the distance between elements of X.
1. For each sample point x_n, $n = 1,\ldots, N$, determine Y_n, that is, the set containing x_n and its k nearest neighbors. Calculate d_n, the mean of the distances between each member and the center (mean) of the set Y_n.
2. For each resample r, $r = 1, 2, \ldots$, and each sample point x_n, $n = 1,\ldots, N$, perform the following steps.
3. Chose a random direction vector ξ_{nr} in the p-dimensional data space (i.e., ξ_{nr} is a p-dimensional random variable uniformly distributed over the hyper-rectangle $[-1, 1]^p$).
4. Rescale the direction vector ξ_{nr} to have the vector length equal to d_n (calculated in step 1).
5. Generate point n of resample r: $x_{nr} = x_n + \xi_{nr}$.

The fixed point-wise perturbation strength (d_n) has been selected to ensure an effective perturbation of each sample point (i.e., to avoid spurious high cluster stability). The method NNR1 can be used to simulate random samples from an unknown mixture population with different intra-population variability and a diagonal matrix as covariance matrix of each population. However, the latter assumption may be too strong for a number of real data sets. For example, the NNR1 method may simulate resample clusters with a hyper-globular shape also in cases where the corresponding clusters in the original sample have a hyper-ellipsoidal shape. (This is a consequence of the fixed perturbation strength in conjunction with the uniformly distributed direction vector.)

Therefore, the user should have other choices for calculating the amount and direction of the perturbation. Experiments have shown that the unintentional generation of artificial outliers by the resampling method may prevent reasonable clustering results of the resamples, while the original sample may have been clustered appropriately. For example, in some cases the fuzzy C-means (FCM) clustering algorithm provided 'missing clusters' for NNR1-type resamples, but not for the original sample (data not shown). Missing clusters were introduced in (Möller, 2007) as being inappropriate clustering results of the FCM. As a conclusion, another method, NNR2, was developed for the analysis of high-dimensional data sets. In NNR2, a data point can be 'shifted' only towards and not beyond one of its nearest neighbors (i.e., into a region of the feature space that actually contains some data).

Furthermore, the mean-based estimate d_n in step 2 of the NNR1 method could be biased if the neighbors of x_n contain outliers or if they contain data points which have been drawn from a population different than the one from which x_n has been drawn. This source of bias can be reduced or avoided by using a robust estimate of the typical inter-point distance such as the median.

Nearest neighbor resampling 2 (NNR2).

0. Let $X = (x_1,\dots, x_N) \subset \Re^p$ be the original sample. Chose $k \geq 2$, two constants $c_1 \geq 0$ and $c_2 > c_1$ for a data-specific calibration of the perturbation strength and a metric for calculating the distance between elements of X.
1. For each sample point x_n, $n = 1,\dots, N$, determine the k nearest neighbors of x_n and calculate d_n, the median distance between all pairs of these k neighbors.
2. For each resample r, $r = 1, 2, \dots$, and each sample point x_n, $n = 1,\dots, N$, perform the following steps.
3. Chose one of the k nearest neighbors of x_n at random. This data point is denoted by x_m. The direction vector from x_n to x_m is used as the direction vector ξ_{nr} for the perturbation of the sample point x_n to generate the resample point x_{nr}.
4. Draw the value c_{nr} from the uniform distribution over the interval $[c_1, c_2]$. Calculate the distance d_{nm} between the sample points x_n and x_m. If d_{nm} is larger than d_n, set the amount of perturbation $|\xi_{nr}| = c_{nr} \cdot d_n$, otherwise set $|\xi_{nr}| = c_{nr} \cdot d_{nm}$, where $|.|$ denotes the vector length. Briefly, $|\xi_{nr}| = c_{nr} \cdot \min(d_n, d_{nm})$.
5. Generate point n of resample r: $x_{nr} = x_n + \xi_{nr}$.

The NNR2 method restricts the positions of simulated (resample) points to the set of points that lie on the lines interconnecting an original sample point and its k nearest neighbors. Real samples are not constrained in this way. However, the application of this constraint leads to the simulation of resample points that cover only those regions of the feature space which are actually occupied by observed data. NNR2 has two advantages in cluster validation studies. Artificial outliers and resulting biases of resample clusterings can be largely avoided. More importantly, there may be data structures which are recognized from a clustering of the original sample, but are no longer separable after a perturbation like that in section 3.4 or that induced by the NNR1 method. The constrained perturbation by the NNR2 method is likely to simulate samples in which such (weakly separable) structures are preserved.

NNR2-type perturbation can be calibrated by adjusting the parameters k, c_1 and c_2. A higher maximal perturbation strength is achieved by increasing the values of k and/or c_2. When choosing $c_2 = 1$ the maximum amount of perturbation for each point equals the median distance between the k nearest neighbors of the respective point. The minimum amount of perturbation of each point can be adjusted by choosing $c_1 > 0$.

3.6 Outlier simulation

Real data sets may contain outliers – even though the data has been processed by a method for the detection and removal of outliers. Therefore, it is desirable to know how robust the result of a clustering algorithm is with respect to the presence of outliers. This knowledge can then be used to select a robust result among a number of candidate results obtained by different clustering algorithms or the same algorithm with different settings of a control parameter (especially, the number of clusters).

For the investigation of cluster stability with respect to outliers Hennig (2007) proposed the replacement of a subset of data points by noise, where "noise points should be allowed to lie far away from the bulk (or bulks) of the data, but it may also be interesting to have noise points in between the clusters, possibly weakening their separation". The author cited Donoho's and Huber's concept of the finite sample replacement breakdown point as a related methodological basis.

Replacing points by noise. Choose M, the number of data points to be replaced by noise, where $1 \leq M < N$ with N being the size of the original sample X. Select a noise distribution and replace M elements of X by points drawn from the noise distribution. For example, the uniform distribution on a hyperrectangle $[-c, c]_p \subset \Re^p$, $C > 1$, may be used, where X had been transformed before the replacement to have a zero mean vector and the identity matrix as covariance matrix.

Addition of noise points. The replacement of original points by noise causes a loss of information which may impair the modeling of the data structure based on a resample clustering. Therefore, an alternative method is proposed here. The M points drawn from the noise distribution could also be added to the data set (i.e., without eliminating any original point). The artificial increase of the resample size in comparison to the original sample size may be less problematic for the purpose of cluster validation than it could be for other resampling applications. It is also possible to find a balance between the artificial increase of the resample size and the information loss: M_R points are replaced, while M_A points are added, where $M_R + M_A = M$. Reasonable choices for M_R and M_A may have to be sought experimentally by the user.

3.7 Feature resampling

Data randomization schemes can also be applied to the set of features used to characterize the population. Such methods will be subsumed below under the term 'feature resampling'. Two of the subsequently described methods (feature subsampling and leave on feature out) leave the information about one or more features unused when generating a resample. These methods may be useful if the number of features p is larger than the number of data points N, where the N points in the p-dimensional coordinate system actually span a data space with less than p (i.e., at most N–1) dimensions. An example is the clustering of biological tissues based on gene expression data, where often $40 \leq N \leq 300$ and $p \geq 1000$ (cf. Monti et al. 2007). In such cases the clustering may become a more effective (because redundant information are eliminated) and the computational effort of the clustering would decrease (owing to the dimension reduction).

Feature subsampling. For $r = 1, 2, \ldots$, select a subset of s_r features randomly from the entire set of p features ($1 \leq s_r < p$). Resample r is obtained by extracting the data of the original sample for the selected features only. The value of s_r can be fixed for generating all resamples (e.g., Smolkin & Gosh, 2003). Alternatively, s_r can be a random variable. Yu et al. (2007) defined the value of s_r to be uniformly distributed over the integer range between $0.75p$ and $0.85p$.

Feature multiscale bootstrapping. There exists a version of bootstrapping which is similar to feature subsampling with variable subsampling size. In this method bootstrap resamples of a variable size $M \leq N$ are drawn from the original sample. This method has been applied to the set of features (gene expression values) when clustering tumor samples (Suzuki & Shimodaira, 2004). An implementation of the method is available in the free statistical software R (Suzuki & Shimodaira, 2006).

Leave one feature out. Generate a set of $i = 1,\ldots, p$ resamples, where p is the number of all features. Resample i contains the original data of all features except feature i. If the number of features is large, the p resamples are relatively similar. Accordingly, a resample clustering is likely to generate p similar partitions and a cluster stability assessment of these partitions may not be informative. A cluster validation approach developed for 'leave one feature out' resamples is the 'figure of merit' (FOM), motivated by Efron's jackknife approach. The FOM quantifies how well the data clustering based on all features except feature i can predict the clustering based on only the data of feature i. For the details see (Yeung et al., 2001).

Feature mapping. Several methods exist for the mapping of a data set into a lower-dimensional space. Among these methods randomized maps suggest themselves for the application to resampling-based cluster validation due to their attractive properties. First, these projections generate random variations of the input data, where the strength of variation can be adjusted almost arbitrarily. Second, some characteristics of the data in the original space, such as the distances between points, are approximately preserved in the projected space (i.e., metric distortions are bounded according to the Johnson-Lindenstrauss theory). Third, the number of dimensions of the projected space can be slightly or considerably smaller than the number of dimensions of the original space. The dimensionality of the projected subspace in which a limited distortion can be obtained depends only on the cardinality of the data and the magnitude of the admissible distortion. For details see (Bertoni & Valentini, 2006). For potential users an implementation of some of these methods is available in the free statistical software R (Valentini, 2006).

Feature weighting. The features to be included into a resample data set can also be randomly weighted. When using continuous positive weights, the information of every feature is included at a certain degree. The lognormal distribution with the mean $\mu = -\log 2$ and the variance $\sigma^2 = 2*\log 2$ can be used for the drawing of the weights. The method can be interpreted as an alternative approach to bootstrapping. The use of the lognormal distribution can be motivated based on relationships of this distribution with the Poisson distribution and the binomial distribution, where the latter is the underlying distribution of a drawing with replacement. The authors of this method (Gana Dresen et al., 2008) called their approach *resampling based on continuous weights*.

4. Results of benchmarking studies

The performance of the above resampling methods is not easily predicted based on a theoretical analysis. Therefore, empirical comparisons of different methods provide useful information for the selection of a method in future applications. This section is a summary of main results reported in five studies which included benchmarking tests of different resampling schemes in a clustering context. In the next section these results will be discussed aiming at general suggestions for the use and choice of resampling methods applied to cluster validation.

In the sequel, the term *bootstrapping* always refers to its non-parametric version. The bootstrap scheme (drawing with replacement) was always applied to the full original sample. To keep the reported information concise the following symbols will be used.

Symbols / abbreviations

N	number of observations (data points) in an original sample
p	number of dimensions (i.e., features used to describe the members of a population)
R	number of resamples generated by using one of the resampling schemes

S subsampling size (percentage of data randomly drawn from the original data sample)
K_R number of clusters generated when clustering each resample data set
K number of clusters of a consensus partition obtained from the set of resample partitions
K_t true (known) number of classes (populations) represented by a benchmarking data set
Minaei-Bidgoli et al. (2004) compared bootstrapping and subsampling for five benchmarking data sets with $N >> p$. The number of resamples R varied from 5 to 1000 and $S \in [5\%, 75\%]$. All resample partitions were obtained by using the K-means clustering algorithm. Resampling performance was measured based on the misassignment (error) obtained for the clustering partitions in comparison to the a priori known class structure of benchmark data sets. The error rate was always calculated for a partition representing the consensus of the R resample partitions. Four different methods from the literature were used providing four consensus partitions in each case. While the generation of resample partitions was repeated for different pre-specified values of the number of clusters (K_R = [2, 20], $K_R > K_t$), each consensus partitions was calculated to have exactly the true number of clusters ($K = K_t$). The error was calculated after finding the optimal assignment between the obtained consensus clusters and the known classes. All experiments were repeated at least 10 times and average errors were reported for some of the best parameter settings of the entire procedure (resampling, resample clustering and consensus clustering).
The error rates obtained for bootstrapping and subsampling were similar. Because the results for subsampling were based on only 5 to 75% of the data sets (parameter S), the authors considered subsampling as a flexible method that can be used to reduce the computational cost in many data mining tasks.
Möller & Radke (2006) compared bootstrapping, subsampling (S = 80%) and perturbation (with three values of the perturbation strength, see section 3.4). R = 20 was fixed in all experiments. Resampling performance was measured based on the rate of false estimates of the number of clusters obtained for the set of the R resample partitions. For each data set 458 estimates of the number of clusters were obtained, resulting from the application of 12 clustering techniques and 41 cluster validity indices. The clustering methods included different hierarchical agglomeration schemes and different metrics, a so-called K-medoid clustering and two versions of fuzzy C-means clustering. Only those of the 458 results were used for the final interpretation where the correct (a priori known) number of clusters was obtained for the original sample as well as for the majority of the resamples. (These constraints were used to exclude errors due to poor original sampling, poor cluster analysis and/or poor configuration of the resampling scheme.) The following data were analyzed: five realizations of each of the stochastic models 2, 3, 4, 6 and 7 described in (Dudoit and Fridlyand, 2003), three microarray data sets with the 200 most differentially expressed genes (*Leukemia*, *CNS* and *Novartis* data described in Monti et al., 2003), the data sets *Iris*, *Liver*, *Thyroid* and *Wine* from the UCI repository (Asuncion & Newman, 2007), and a data set of functional magnetic resonance imaging data. Data sets with $N >> p$ as well as $N << p$ were included.
In general, the error rates obtained for the perturbation technique were smaller than the error rates for subsampling. Both perturbation and subsampling led to clearly smaller error rates than bootstrapping. The same ranking was obtained when considering all (about 15.000) estimates of the number of clusters without applying the mentioned constraints. The occurrence of false estimates even for a perturbation with 1% noise indicated that the small errors obtained for the perturbation scheme are not spurious results (i.e., the perturbation

was effective). The authors concluded that the increased errors for subsampling and bootstrapping may have been a consequence of the information loss (i.e., 20% and about 37% of the original sample were not used for the generation of a resample in the subsampling and bootstrapping schemes, respectively). The authors further concluded that resampling schemes without this information loss are more useful in cluster validation studies, in particular, when the original samples have a small size.

Hennig (2007) compared bootstrapping, subsampling (S = 50%), the replacement of sample points by noise ($M = 0.05N$, $c = 3$ and $M = 0.2N$, $c = 4$, see section 3.6), two versions of jittering (parameter q was set respectively to the 0.1- and 0.25-quantiles of the values d_{ij}, see section 3.2), and the combination of bootstrapping and jittering ($q = 0.1$). $R = 20$ was fixed in all experiments. Resampling performance was measured based on several types of results. First, cluster stability was assessed by calculating the agreement between the partition generated from each resample and the partition obtained for the original sample (The agreement between clusters of two partitions was measured by the Jaccard index (cf. Theodoridis & Koutroumbas, 2006).) Second, for model data with true cluster memberships, it was measured how well the clustering of an original sample represented the model structure. (The Jaccard index was applied to the cluster memberships of each original sample and the true cluster memberships.) Third, the correlation between the two aforementioned types of results was calculated. Different clustering methods were used, namely, a method called *normal mixture plus noise*, K-means, 10% trimmed K-means and average linkage hierarchical agglomeration. 50 original samples were generated for each of two stochastic models ($K_t = \{3, 6\}$, $N >> p$). One model included outliers. One biological data set ($N = 366$, $p = 306$) was analyzed that was known to contain substructure – without exact knowledge about the 'true' cluster composition.

Due to the choice of the analysis design, three types of results were distinguished. 1) partitions of original samples with a fairly good representation of the model structure and a stable clustering of the resample data that corresponded to this model structure, 2) partitions of original samples with a relatively poor representation of the model structure and an unstable clustering of the resample data and 3) partitions of original samples with a relatively poor representation of the model structure and, nevertheless, a stable clustering of the resample data. The results of the types 1 and 2 are desirable, because they permit appropriate conclusions about the performance of clustering of unknown data based on resample cluster stability scores. Results of type 3 are problematic. If the original sample does not adequately represent the true population structure, also the clustering of this sample may not represent the true structure. Even though it is desirable to obtain an indication of the poor modeling result, namely, an *unstable* clustering for the resample data. Otherwise, this kind of inappropriate modeling cannot be distinguished from proper clustering models when the true population is unknown.

Based on all results, subsampling was considered as being the best method, followed by the combination of bootstrapping/jittering and bootstrapping alone. The replacement of data points by noise was also useful in a number of case, including some cases where the other methods did not perform well (i.e., they provided a number of type-3 results). Jittering showed generally a poor performance (i.e., a relatively large fraction of type-3 results for most of the data sets and clustering algorithms). The author concluded that a good strategy in practice can be the use of *one* of the schemes bootstrapping, bootstrapping/jittering and subsampling together with *one* scheme for replacing data by noise.

Gana Dresen et al. (2008) compared bootstrapping and feature weighting. R = 1000 was fixed in all experiments. Resampling performance was measured based on the stability of branches of cluster trees (dendrograms) obtained from hierarchical agglomerative clustering of the resample data sets. Furthermore, a majority consensus tree was generated from the resample cluster trees and this consensus tree was compared with the cluster tree obtained from the original sample (based on the Rand index; cf. (Theodoridis & Koutroumbas, 2006)). For the comparison, gene expression data from 24 chromosomes (p = 8 to 648 probe sets) of N = 20 tumor patients were used. For a subset of the data, knowledge about actual clustering structure was available. A data set containing p = 7 features of N = 22 primates was also analyzed. In addition, it was investigated how well groups of simulated differentially expressed genes can be robustly detected based on bootstrapping and feature weighting.
In a number of cases bootstrapping and feature weighting showed comparable performance. However, in several cases bootstrapping led to inappropriate consensus cluster trees. That is, the structure was inappropriate, many spurious singleton clusters were obtained and especially the false clusters proved to be stable under the bootstrap procedure. The authors concluded that resampling with continuous weights is strongly recommended because it performed at least as well as bootstrapping and in some cases it surpassed bootstrapping. In particular, feature weighting was more appropriate than bootstrapping to cluster small size samples.
Möller and Radke (2006b) reported results of estimating the number of clusters based on two different approaches, denoted here by A and B. In approach A (Monti et al., 2003) resampling is performed by subsampling (S = 80%). In approach B (Möller & Radke, 2006b) nearest neighbor resampling (NNR1) was used. Approach B led to better results than A on high-dimensional gene expression benchmark data ($N << p$). In particular, a fairly good recovery of known tumor classes was possible based on just R = 10 nearest neighbor resamples in approach B, while approach A led to similar or worse results based on R = 200 or R = 500 subsamples (with R depending on the clustering algorithm). These results indicated the usefulness of nearest neighbor resampling; however, the performance differences may partly be attributable to the different methods selected in the approaches A and B, respectively, for clustering and for estimating the number of clusters.

5. Results of nearest neighbor resampling

Results of a direct benchmarking of NNR and other resampling methods are currently not available. However, several cluster validation results based on NNR have been obtained. Ulbrich (2006) used the NNR1 algorithm to identify robust and prognostic gene expression patterns by clustering of tumor patients. Guthke et al. (2007) performed clustering to find co-expression patterns of genes for the subsequent utilization in systems biology. They showed that the NNR1-based cluster stability analysis can be used to complement and confirm the results of a different quality assessment, namely the vote of so-called cluster validity indices (Bezdek and Pal, 1998).
The use of the NNR2 method has provided strong indications that (estrogen receptor positive) breast cancer can be robustly subdivided into three, perhaps four, classes which are represented by different prognostic gene expression profiles. This result has been consistently obtained for gene expression data and survival time data generated in four different studies based on two different DNA microarray platforms and including the data from more than 700 tumor patients (Iffert, 2007).
In combination with methods presented by Fred and Jain (2006), the NNR2 algorithm was recently applied to the gene expression benchmark data sets of known tumor classes

published by Monti et al. (2003). In several cases the obtained class recovery scores were higher than those obtained by Monti et al. based on subsampling and those obtained by Yu et al. (2007) who analyzed the same data based on feature subsampling (Möller, 2008). However, the cluster analysis methods used in these studies were also different.

6. Discussion and conclusions

Bootstrapping (drawing with replacement) is perhaps the most widely known and recommended resampling approach, because it is a standard approach in for statistical inference methods (Efron & Tibshirani, 1993). If the sample size is large and the true distribution is well represented by the data, bootstrapping may also be useful for the validation of clustering results. That is, other resampling schemes may not lead to more accurate results (cf. Minaei-Bidgoli et al., 2004). Under these circumstances the user may prefer bootstrapping, because no control parameter has to be set.

However, as shown in complementary investigations (section 4), for statistical cluster validation it is recommended to prefer other methods than bootstrapping. When the sample size is large, subsampling is likely to perform as well as bootstrapping (Minaei-Bidgoli et al., 2004; Hennig, 2007) or even better (Möller & Radke, 2006a), where the clustering of subsamples requires a lower computing effort. If the clustering result is to be used as the basis for a classifier of unknown samples, the subdivision scheme (e.g., Dudoit & Fridlyand, 2003) may be the best choice, because it is focused on minimizing the prediction error, while subsampling results are commonly used for assessing cluster stability (e.g., Tseng and Wong, 2005; Fred & Jain, 2006). When the sample size was small, perturbation and resampling with continuous weights have been shown to outperform bootstrapping (Radke & Möller, 2006a; Gana Dresen et al., 2008).

If the sample size is small, a further decrease by drawing subsamples prevents the "learning" of a good model from the resample data. In this case, perturbation methods are more suitable than *sampling from a sample* (Radke & Möller, 2006a). However, the user should be aware that this type of perturbation works best only if all populations of the hypothesized mixture population have equal variability. Furthermore, this method requires an estimate or guess of the proper perturbation strength. Therefore, it may be recommended to search for stable clusters by using different values of the perturbation strength. This could increase the confidence in the validity of the obtained clusters and their completeness with respect to the true structures.

Nearest neighbor resampling (NNR) is an attractive alternative to the perturbation described in section 3.4. In the absence of prior knowledge, the parameter setting for the NNR2 method is less critical than the specification of a global perturbation strength. According to the author's knowledge, the NNR methods were described here for the first time in detail. Especially, the NNR2 method has provided promising results when clustering data with complex structures (see section 5). Therefore, based on practical experience, the author recommends the NNR approach for applications of unsupervised machine learning. Even though, more comprehensive simulations and benchmarking studies with other methods are desired know the performance of the NNR approach in a more general context.

Feature resampling may be a way to bypass some of the problems associated with the above resampling schemes. However, the successful use of some of these techniques is limited to applications where the assumptions underlying these techniques are fulfilled. This argument applies, for example, to *feature subsampling* and *leave one feature out* which involve a loss of original information (cf. Yeung et al., 2001). *Feature mapping* (Bertoni & Valentini, 2006) appears to be a promising approach due to the combination of dimension reduction and the distance–preserving character of the mapping. It would be interesting to have empirical results indicating the relative merits of this kind of mapping in comparison to several other methods presented above. Another promising method is *resampling with continuous weights* (Gana Dresen et al., 2008). As stated by the authors it would be interesting to investigate the performance of this method in combination with other clustering algorithms than the hierarchical ones used.

The resampling methods for the simulation of measurement errors (*jittering*) and outliers are useful if the user wants to confirm the robustness of the final clustering result with respect to these factors of influence. However, robust results of such an analysis are only a pre-condition for a good clustering model. The fact that clusters are stable under jittering and the insertion of artificial outliers must not be interpreted over-optimistically as the indication of a real mixture population.

Hennig (2007) argued that "Generally, large stability values do not necessarily indicate valid clusters, but small stability values are informative. Either they correspond to meaningless clusters (in terms of the true underlying models), or they indicate inherent instabilities in clusters or clustering methods." Following this view, any stable cluster and any good prediction based on the *subdivision* approach (section 3.1) may have to be verified by repeating the cluster analysis with an increasing amount of (random) change made to the data. One criterion for stopping these repetitions is that some clusters 'disappear' under the influence of resampling, while other clusters can still be recovered. This observation would not be expected in the absence of any true structure. Another termination criterion is fulfilled if the clustering structures 'disappear' only if the amount of random change has become clearly larger compared to the effect of the measurement error. This fact may be deducible even if the measurement error can only be roughly estimated.

An inevitable decision that has to be made by the user is the selection of the number of resamples, R. A proper value of R depends on both the structure of the investigated data and the resampling method used. In fact, compact and well separated clusters would be robustly detected based on fewer resamples than overlapping, noisy clusters. In addition, the more original sample information is utilized for generating each resample, the fewer resamples are likely to be required. For example, R = 10,..., 30 resamples obtained from NNR methods have been sufficient to robustly recover clustering structures of small high-dimensional samples (Ulbrich, 2006; Iffert, 2007; Möller & Radke, 2006b). In contrast, R = 100,…, 1000 resamples have often been used for the cluster validation based on bootstrapping or subsampling (cf. section 4). If the information loss of the mapping from the original sample to the resample exceeds a data specific-threshold, the lack of information in the individual resamples may not be compensable by any increase in the number of resamples.

Computerized observation techniques in an increasing number of research areas generate high-dimensional data (e.g., DNA microarray data, spectral data with a high frequency resolution and complex image and video data). High-dimensional data sets are more likely than others to provide clusterings which are not significant and meaningful. Especially in those cases, but also when clustering any other sample data, the use of resampling methods is recommend as a valuable aid for a statistical model quality assessment.

The above description and review of resampling schemes and their performance as well as the presentation of a new approach (NNR) may help users to select an appropriate method in future studies.

8. Acknowledgement

This study was supported by the HKI Jena and the International Leibniz Research School for Microbial and Biomolecular Interactions (ILRS) Jena (http://www.ilrs. hki-jena.de).

9. References

Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics. uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

Ayad, H.G. & Kamel, M.S. (2008). Cumulative voting consensus method for partitions with variable number of clusters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 1, 160-173

Bertoni, A. & Valentini, G. (2006). Randomized maps for assessing the reliability of patients clusters in DNA microarray data analyses. *Artificial Intelligence in Medicine*, Vol. 37, 85-109

Bezdek, J.C. & Pal, N.R. (1998). Some new indexes of cluster validity, *IEEE Transactions on Systems, Man and Cybernetics – Part B*, Vol. 28, 301-315

Bittner, M. & 27 co-authors (2000). Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, Vol. 406, 536-540

Borgelt, C. & Kruse, R. Finding the number of fuzzy clusters by resampling. *Proceedings of the IEEE Int. Conf. on Fuzzy Systems*, pp. 48-54, ISBN: 0-7803-9488-7, Vancouver, Canada, Sept. 2006, IEEE Press, Piscataway, NJ, USA

Dudoit, S. & Fridlyand J. (2003). A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology* 3:7, Online ISSN 1465-6914

Efron, B. & Tibshirani, R.J. (1993). *An introduction to the bootstrap.* Chapman & Hall/CRC, ISBN 0-412-04231-2

Fred, A. & Jain, A.K. (2006). Learning pairwise similarity. *Proceedings of the Int. Conf. on Pattern Recognition (ICPR)*, pp. 892-895, ISBN 0-7695-2521-0, Hong-Kong, August 2006, IEEE Computer Society Press

Gana Dresen, I.M.; Boes, T.; Huesing, J.; Neuhaeuser, M. & Joeckel, K.-H. (2008). New resampling method for evaluating stability of clusters. *BMC Bioinformatics*, 9:42, doi:10.1186/1471-2105-9-42

Guthke, R.; Kniemeyer, O.; Albrecht, D.; Brakhage, A.A. & Möller, U. (2007). Discovery of Gene Regulatory Networks in Aspergillus fumigatus, In: *Knowledge discovery and emergent complexity in bioinformatics,* Tuyls, K. et al. (Ed.), *Lecture Notes in Bioinformatics 4366*, 22-41, Springer, ISBN 978-3-540-71036-3, Berlin/Heidelberg

Hastie, T.; Tibshirani, R. & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer, ISBN 978-0-387-95284-0

Hennig, C. (2007). Cluster-wise assessment of cluster stability. *Computational Statistics and Data Analysis* Vol. 52, 258-271

Iffert, W. (2007). Investigations for the prognosis of diseases by simultaneous analysis of gene expression data and survival time data. (in German), Diploma Thesis in Bioinformatics, August 2007, Friedrich Schiller University, Jena, Germany

Kell, D.B. & Oliver, S.G. (2004). Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. *BioEssays,* Vol. 26, 99-105

Lunneborg, C.E. (2000). *Data Analysis by Resampling. Concepts and Applications*, Duxbury Press, ISBN 0-534-22110-6, Pacific Grove, CA, USA

McLachlan, G.J. & Khan, N. (2004). On a resampling approach for tests on the number of clusters with mixture model-based clustering of tissue samples. *Journal of Multivariate Analysis*, Vol. 90, 90-105

Minaei-Bidgoli, B.; Topchy, A. & Punch, W.F. (2004). A comparison of resampling methods for clustering ensembles. *Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications (MLMTA)*, pp. 939-945, Las Vegas, Nevada, June 2004

Möller, U. & Radke, D. (2006a). Performance of data resampling methods for robust class discovery based on clustering. *Intelligent Data Analysis* Vol. 10, No. 2, 139-162

Möller, U. & Radke, D. (2006b). A cluster validity approach based on nearest neighbor resampling, *Proceedings of the Int. Conf. on Pattern Recognition (ICPR)*, pp. 892-895, ISBN 0-7695-2521-0, Hong-Kong, August 2006, IEEE Computer Society Press

Möller, U. (2007). Missing clusters indicate poor estimates or guesses of a proper fuzzy exponent. In: *Applications of Fuzzy Sets Theory*, Masulli, F.; Mitra, S.; Pasi, G. (Ed.), *Lecture Notes in Artificial Intelligence 4578,* 161-169, Springer, ISBN 978-3-540-73399-7, Berlin-Heidelberg

Möller, U. (2008). Methods for robust class discovery in gene expression profiles of tissue samples. Poster presentation at the conference *Bioinformatics Research and Development (BIRD*), July 2008, Vienna, Austria

Monti, S.; Tamayo, P.; Mesirov, J. & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, Vol. 52, 91–118

Smolkin, M. & Ghosh, D. (2003). Cluster stability scores for microarray data in cancer studies. *BMC Bioinformatics*, 4:36, www.biomedcentral.com/1471-2105/4/36

Strehl, A. & Gosh, J. (2002). Cluster ensembles: A knowledge reuse framework for combining multiple partitions, *J. of Machine Learning Research,* Vol. 3, 583–617

Suzuki, R. & Shimodaira, H. (2004). An application of multiscale bootstrap resampling to hierarchical clustering of microarray data: How accurate are these clusters? *Proceedings of the Int. Conf. on Genome Informatics (GIW),* p. P034

Suzuki, R. & Shimodaira, H. (2006). Pvclust: an R package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, Vol. 22, No. 12, 1540-1542

Theodoridis S. & Koutroumbas, K. (2006). *Pattern recognition.* 3rd ed., Academic Press, ISBN 0-12-369531-7, San Diego

Topchy, A.; Jain, A.K. & Punch, W. (2005). Clustering ensembles: models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 12, 1866-1881

Tseng, G.C. & Wong, W.H. (2005). Tight clustering: a resampling-based approach for identifying stable and tight patterns in data. *Biometrics*, Vol. 61, 10-16

Ulbrich, B. (2006). Improvements of tumor classification based on molecular-biological patterns by using new methods of unsupervised learning. (in German), Diploma Thesis in Bioinformatics, August 2007, Friedrich Schiller University, Jena, Germany

Valentini, G. (2006). Clusterv: a tool for assessing the reliability of clusters discovered in DNA microarray data. *Bioinformatics*, Vol. 22 No. 3, 369-370

Yeung, K.Y.; Haynor, D.R. & Ruzzo, W.L. (2001). Validating clustering for gene expression data. *Bioinformatics*, Vol. 17, No. 4, 309-318

Yu, Z.; Wong, H.-S. & Wang, H. (2007). Graph-based consensus clustering for class discovery from gene expression data. *Bioinformatics*, Vol. 23, No. 21, 2888-2896

15

3D Shape Classification and Retrieval Using Heterogenous Features and Supervised Learning

Hamid Laga
Tokyo Institute of Technology
Japan

1. Introduction

Content-based 3D model retrieval (CB3DMR) aims at augmenting the text-based search with the ability to search 3D data collections by using examples, sketches, as well as geometric and structural features. In recent years there is an increasing demand on such tools as 3D graphics technology is becoming widely accessible and a large amount of 3D contents is being created and shared.
Usually an algorithm for 3D model classification and retrieval requires: (1) an efficient representation of the 3D data that is suited for search, and (2) a good similarity function in order to measure distances between entities in the feature space. The first step involves feature extraction, feature selection strategy to keep only the most relevant features, and a method for encoding the features as real-valued vectors called *shape descriptors*. Shape descriptors provide a numerical representation of the salient features of the data. They should be an abstraction of the semantics of the shape and shape category. Many descriptors have been proposed for content-based 3D model classification and retrieval but none of them has achieved high-level performance on all shape classes. For instance:

- Global geometric features, which are easy to compute and compare, are poor in terms of discrimination power since they are unable to capture the intra-class shape variability. Alternatively, local features, such as spin images (Johnson, 1997) and shape contexts (M.Kortgen et al., 2003) are more effective for intraclass retrieval. However, their extraction and comparison are expensive in terms of computation and storage requirements. A key observation is that many of these features are redundant and only a small subset of them, called representative feature set, is really discriminative. Thus, there is a need for selecting automatically the optimal set of features. The selected set should be specific to each class of shapes, and adapted to different types of user queries and data classifications.
- Geometry-based features, such as Light Fields (LFD) (Chen et al., 2003) and spherical harmonic (Funkhouser et al., 2003) descriptors, represent shapes with their global geometric characteristics. On the other hand, graph-based descriptors, such as Reeb-graphs and skeleton representations (Hilaga et al., 2001; T.Tung & F.Schmitt, 2005), encode the structural characteristics of the shape, and therefore are more suitable for indexing articulated shapes. Consequently, there is a need for combining *heterogeneous*

features in order to achieve best performance. By heterogeneous we mean features of different types and scales.

From the machine learning point-of-view, efficient selection and combination of heterogeneous features for classification and retrieval poses many challenges. The first issue is how to choose among a large set of features, a subset that allows to achieve high-level performance. The second issue is the feature normalization problem. Heterogenous features are often of different scales. Therefore, incorporating them directly into the similarity function will result in low retrieval performance as higher scale features will influence more the similarity. This issue is related to the feature weighting strategy.

The goal of this chapter is to develop an effective 3D shape classification and retrieval method that uses discriminative shape features automatically selected from a large set of heterogeneous features. The construction of the representative set can be regarded as a machine learning task. Particularly, supervised learning allows to capture the high-level semantic concepts of the data using low-level geometric features. Our key idea is to use a large set of local and global features, eventually not orthogonal, then use a supervised learning algorithm to select only the most efficient ones. We experimented with AdaBoost which provides a mean for feature selection and classifier combination. Boosting, like many machine-learning methods, is entirely data-driven in the sense that the classifier it generates is derived exclusively from the evidence present in the training data itself (Schapire, 2003). Moreover, allowing redundancy and overlapping in the feature set has been proven to be very efficient in recognition and classifications tasks than orthogonal features (Tieu & Viola, 2004). Specifically, we make the following contributions:

- An algorithm for learning the discriminative features of a class of shapes from a training set. The algorithm allows also to quantify the discrimination ability of a shape feature with respect to the underlying classification. Features of high discrimination ability of each class of shapes will be used for processing unseen objects (classification of the query, and retrieving the most similar shapes to the query).
- A method for matching shapes using only the most relevant features to each class of shapes. This approach can be used with either a flat or a hierarchical classification of the data resulting in a multi-scale organization of the feature space.
- The ability to use heterogeneous features for classification is a major deviation from previous work.

The remainder of this paper is organized as follows: the next section reviews the related work. Section 2.3 gives and overview of the proposed framework and outlines the main contributions. In Section 3 we describe the type of 3D shape descriptors we will use in this chapter. Section 4 details the developed algorithm for feature selection and combination in the case of a binary classification (Section 4.1), and its generalization to a multi-class problem (Section 4.2). In Section 5 we detail the query processing method for classification and retrieval. Experimental results and evaluations are given in Section 6. Section 7 concludes the paper and outlines the major issues for future work.

2. Related work

3D shape analysis, classification and retrieval received significant attention in recent years. In the following we review the most relevant techniques to our work. For more details, we refer the reader to the recent surveys of the topic (Lew et al., 2006; Tangelder & Veltkamp, 2004; Iyer et al., 2005).

2.1 Descriptors for 3D model retrieval

For efficient comparison and similarity estimation, 3D models can be represented with a set of meaningful descriptors that encode the salient geometric and topological characteristics of their shapes. The database objects are then ranked according to their distance to the descriptors of the query model. These descriptors are either global, local, or structural. Structural descriptors such as Reeb graphes (Hilaga et al., 2001; T.Tung & F.Schmitt, 2005) aim at encoding the topological structure of the shape. They can be used for global matching as well as partial matching (Biasotti et al., 2006).

Global descriptors describe an entire 3D shape with one single feature vector. In this family, the Light Fields (LFD) (Chen et al., 2003) are reported to be the most effective (Shilane et al., 2004). (Funkhouser et al., 2003) map a 3D shape to unit spheres and use spherical harmonics (SH) to analyze the shape function. Spherical harmonics can achieve rotation invariance by taking only the power spectrum of the harmonic representation, and therefore, discarding the rotation dependent information (Kazhdan et al., 2003). (Novotni & Klein, 2003) use 3D Zernike moments (ZD) as a natural extension of SH. (Laga et al., 2006) introduced flat octahedron parameterization and spherical wavelet descriptors to eliminate the singularities that appear in the two poles when using latitude-longitude parameterization, and therefore, achieve a fully rotation invariant description of the 3D shapes. Recently, (Reuter et al., 2006) introduced the notion of shape DNA as fingerprints for shape matching. The fingerprints are computed from the spectra of the Laplace-Beltrami operators. They are invariant under similarity transformations and are very efficient in matching 2D and 3D manifold shapes. However, it is not clear how they can be extended to polygon soup models.

Global descriptors are very compact, easy to compute, and efficient for broad classification of 3D shapes. However, they cannot capture the variability of the shapes and their subtle details necessary for intra-class retrieval. Local feature-based methods can overcome these limitations by computing a large set of features at different scales and locations on the 3D model. Spin images (Johnson, 1997) and shape contexts (M.Kortgen et al., 2003) have been used for shape retrieval as well as for matching and registering 3D scans. Local features are very efficient to discriminate objects within the same class. However, similarity estimation requires combinatorial comparison, making them not suitable for realtime applications such as retrieval.

2.2 Feature selection and relevance feedback

3D model retrieval by matching low level features does not fully reflect the semantics of the data. For instance, most of the previous techniques cannot distinguish between a flying bird and a commercial airplane. This is commonly known as the *semantic gap*. Recent progress in pattern recognition suggested the use of supervised learning to narrow the semantic gap. This allows the automatic selection of salient features of a single 3D model within a class of shapes, and also the use of the results of classification to improve the performance of retrieval algorithms.

The basic learning approach is the Nearest Neighbor classifier. It has been used for the classification of 3D protein databases (Ankerst et al., 1999), and also 3D engineering parts (Ip et al., 2003).

Hou et al. (2005) introduced a semi-supervised semantic clustering method based on Support Vector Machines (SVM) to organize 3D models semantically. The query model is first labeled with some semantic concepts such that it can be assigned to a single cluster.

Then the search is conducted only in the corresponding cluster. Supervised learning and ground-truth data are used to learn the patterns of each semantic cluster off-line. Later, (Hou & Ramani, 2006) combine both semantic concepts and visual content in a unified framework using a probability-based classifier. They use a linear combination of several classifiers, one per descriptor. The individual classifiers are trained in a supervised manner, and output an estimate of the probability of data being classified to a specific class. The output of the training stage is used to estimate the optimal weights of the combination model. The retrieval is performed in two stages; first they begin by estimating the conditional probability of each class of shapes given the query. Then they perform shape search inside each candidate class. The new similarity measure is a unified distance that integrates the probability estimation from the classifiers, a combination of classifiers learned off-line, and a shape similarity distance. This is the closest work to ours. In this approach features and type of classifiers are set manually. In our case, we aim at selecting automatically the most salient features.

(Shilane & Funkhouser, 2006) investigated on how to select local descriptors from a query shape that are most distinctive and therefore most relevant for retrieval. Their approach uses supervised learning to predict the retrieval performance of each feature, and select only a small set of the most effective ones to be used during the retrieval. (Funkhouser & Shilane, 2006) introduced priority-driven search for partial matching of 3D shapes. The algorithm produces a ranked list of *c*-best target objects sorted by how well any subset of *k* features on the query matches features on the target object. As reported by the authors, the timing results is dominated by the number of features for each target object and the number of scales for each feature. The algorithm we propose can deal with large set of features while maintaining the processing time at interactive rates.

The approach most similar to our own is that of (Tieu and Viola, 2004) where they applied the AdaBoost algorithm (Schapire, 2003) to online learning of the similarity of a given query to the target objects in image retrieval. It has been recently extended to learn the intrinsic features for boosting 3D face recognition (Xu et al., 2006). AdaBoost enables the use of a very large set of features while keeping the processing time at the run-time very attractive. We improve over this approach in two important ways. First we investigate the application of AdaBoost to the general problem of 3D model retrieval. Second, we learn, off-line, the optimal salient and discriminative set of features for each class of shapes with respect to objects in the entire database. These improvements allow our 3D model retrieval algorithm to achieve high-level performance in terms of retrieval efficiency and computation time.

2.3 Overview

Figure 1 gives an overview of our approach. At the training stage a strong classifier is learned using AdaBoost. The classifier returns the likelihood that a given 3D model *O* belongs to a class of shapes *C*. First a large set of features are extracted from every model in the database. Then a set of binary classifiers are trained using AdaBoost. Each binary classifier learns one class of shapes and its optimal set of salient features. Finally, the binary classifiers are combined into one multi-class classifier. In our implementation we experimented with the Light Field Descriptors (LFD) (Chen et al., 2003) (100 descriptors per-shape), the Gaussian Euclidean Distance transform (GEDT) (Shilane et al., 2004) (32

descriptors per-shape, each descriptor is computed on a concentric sphere of radius r, $0 \leq r \leq 1$), and a combination of the two descriptors which will be referred by LFD-GEDT.
At the run-time, given a query model Q, a ranked list of k–best matches is produced in a two-stage process that involves classification and search. First, a large set of features are computed from the query model Q, in the same manner as for the database models. Then in the classification stage, a set of highly relevant classes to Q is found. Each binary classifier C_i decides wether the class C_i is relevant to the query Q or not. In the retrieval stage, the similarity between Q and the models in every relevant class C_i is estimated and a ranked list of the best matches is returned.

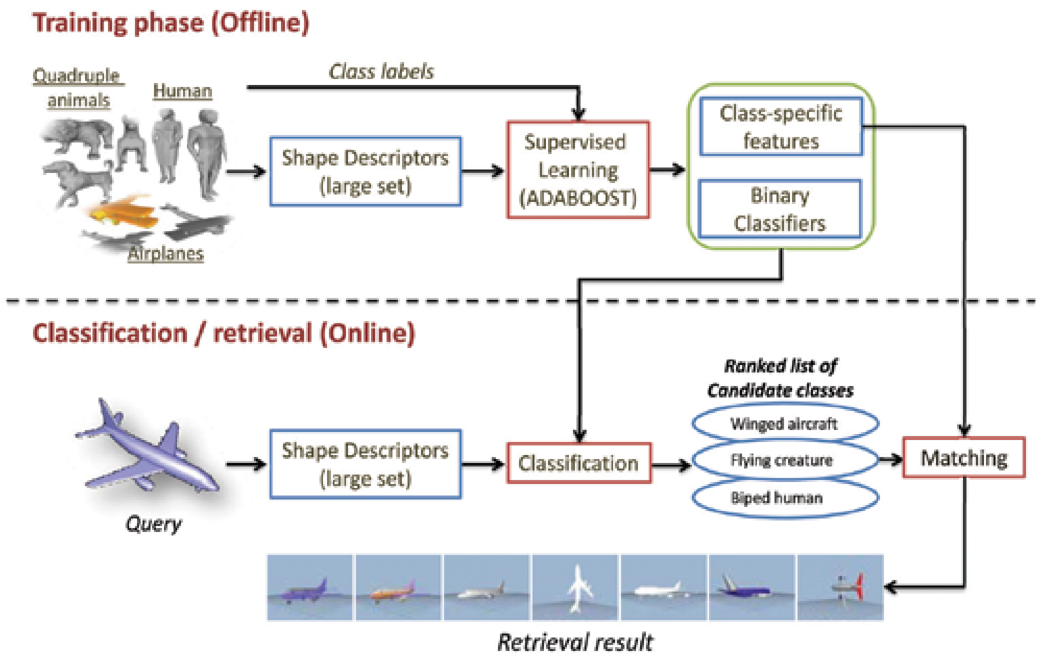


Fig. 1. Overview of AdaBoost-based 3D model classification and retrieval. At the training stage a strong classifier is learned using AdaBoost. The strong classifier is based on a combination of the most discriminative features of the shape. At the run-time, a query is first classified into a set of candidate classes, then the search for the best matches is performed inside the candidate classes.

The key step in this process is the way we predict the saliency of each feature with respect to a class of shapes in the training set. More formally, the saliency of a feature $\vec{v}$ with respect to a class of shapes C is the ability of this feature to discriminate the shapes of class C from the shapes of other classes in the database. Mathematically, given the binary classifier $C_{\vec{v}}$ trained with the feature $\vec{v}$, the saliency of $\vec{v}$ is directly related to the overall classification error of $C_{\vec{v}}$ on the data set. However, none of the existing classifiers that are based on a single feature can achieve zero classification error. Therefore none of the features is sufficiently salient. AdaBoost provides a way for combining weak classifiers and shape features, eventually of different types and saliency degrees, into a single strong classifier with high classification performance. There are several advantages of this approach:

Although a large set of features is extracted both at the training and online stages, only a small subset of the features (between 10 to 50) is used during the similarity estimation. This allows retrieval at interactive rates.

- The algorithm selects automatically the representative set of features for each class of shapes, and provides a mean for automatic combination of the selected features. This has potential applications in shape classification and recognition.
- The algorithm provides an automatic way to truncate the list of the $k-$best matches, i.e, it provides a mean for saying wether the database contains models which are similar to a given query or not.
- This approach allows to perform both inter-class and intra-class retrieval. AdaBoostbased classifier allows to find the relevant classes to the query. Then, in a second step, the search can be performed inside each relevant class using, eventually, different types of descriptors.

For feature extraction, we use the Light Field descriptors (LFD) (Chen et al., 2003) and Gaussian Euclidean Distance Transform (GEDT) (Shilane et al., 2004). However, a further investigation is required to test the efficiency of other descriptors when boosted, which is beyond the scope of this paper.

3. 3D shape descriptors

The process starts by computing a large set of features for each model in the training set, which is the content of the database to search. There are many requirements that the features should fulfill: (1) compactness, (2) computation speed, and (3) the ability to discriminate between dissimilar shapes. However, in real applications it is hard to fulfill these requirements when the goal is to achieve high retrieval accuracy. In fact, compact features, which are easy to compute, are not discriminative enough to be used for high accuracy retrieval. We propose to extract a large set of features following the same idea as in (Tieu & Viola, 2004).

There are many shape descriptors that can be computed from a 3D model. A large set of Spherical harmonics (Funkhouser & Shilane, 2006) and spherical waveletbased descriptors (Laga et al., 2006) can be computed by moving the center of the sphere across different locations on the shape's surface or on a 3D grid. However, in the literature, it has been proven that view-based descriptors outperform significantly the spherical descriptors. In our implementation we considered two shape descriptors evaluated in the Princeton Shape Benchmark: the Light Fields Descriptor (LFD), and the Gaussian Euclidean Distance Transform Descriptor (GEDT). For the completeness purpose we give a brief overview of these descriptors but the reader can find further details in the original paper (Shilane et al., 2004):

- **Light Field Descriptor (LFD) (Chen et al., 2003):** a view-based descriptor computed from 100 images rendered from cameras positioned on the vertices of a regular dodecahedron. Each image is encoded with 35 Zernike moments, and 10 Fourier coefficients. In this paper we use our own implementation.
- **Gaussian Euclidean Distance Transform (GEDT) (Shilane et al., 2004):** a 3D function whose values at each point is given by composition of a Gaussian with the Euclidean Distance Transform of the surface. It is computed on 64×64×64 axial grid, translated such as the origin is at the point (32, 32, 32), scaled by a factor of 32, and then

represented by 32 spherical descriptors representing the intersection of the voxel grid with concentric spherical shells. Values within each shell were scaled by the square-root of the corresponding area and represented by their spherical harmonic coefficients up to order 16.

To evaluate the performance of the feature selection algorithm we will consider also a combination of the two descriptors, herein after referred by LFD-GEDT. Notice that these two descriptors are encoding different properties of the shape and may have different scales. Also, the set of features contains many redundancies: in the case of LFD for example, two symmetric view points will have the same 2D projection, and close points in the Euclidean sense will have their associated LFDs very similar. On one hand, this will increase significantly the storage and computation time required for matching and retrieval. However, on the other hand, it will guarantee that the feature set can capture the shape variability. Therefore, we rely on the learning stage to select the salient ones that achieve best classification and retrieval performance.

4. Supervised classification

The first task in our approach is to build a classifier $\mathcal{C}$ that decides wether a given 3D model O belongs to a class of shapes C or not. The challenge is to define a feature space such that 3D shapes belonging to the same class are mapped into points close to each other in the new feature space. Clusters in the new space correspond to classes of 3D models. There are many feature spaces that have been proposed in the literature, but it has been proven that none of them achieved best performance on all classes of shapes. We propose to follow a machine learning approach where each classifier is obtained by the mean of training data.

4.1 Boosting the binary classification

A brute force approach for comparing a large set of features is computationally very expensive, and in the best case, it requires $M \times d \times N$ comparisons, where M is the number of feature vectors used to describe each 3D model, d is the dimension of the feature space, and N is the number of models in the database.

Previous works consider this problem from the dimensionality reduction point of view. (Ohbuchi et al., 2007) provide an overview and performance evaluation of six linear and non-linear dimensionality reduction techniques in the context of 3D model retrieval. They demonstrated that non-linear techniques improve significantly the retrieval performance. There have been also a lot of research in classifiers that have a good generalization performance by maximizing the margin. Speed is the main advantage of boosting over other classification algorithms such as Support Vector Machines (SVM) (Hou et al., 2005), and non-linear dimensionality reduction techniques (Ohbuchi et al., 2007; Ohbuchi & Kobayashi, 2006). It can be also used as a feature selection algorithm, and provides a good theoretical quantification of the upper bound of the error rate, therefore a good generalization performance.

We use AdaBoost version of boosting. Every weak classifier is based on a single feature of a 3D shape (recall that we have computed a large set of features for each 3D model). The final strong classifier, a weighted sum of weak classifiers, is based on the most discriminating features weighted by their discriminant power. The algorithm is summarized in Algorithm 1.

```
Algorithm 1: AdaBoost algorithm for binary classification
Input: Training set $S_C = \{(V_i, y_i), i = 1 \dots N\}$, where $V_i = \{\vec{v}_1, \dots, \vec{v}_K\}$ is a
       large set of $K$ features computed from the 3D object $O_i$, and
       $y_i \in \{+1, -1\}$ is the targeted classification of $O_i$.
Output: The decision function $f_C$, such that, $f_C(O) > 0$ is $O \in C$, and
        $f_C(O) < 0$ if $O \notin C$.
  1. Initialize the sample weights: $w_0 = \{w_i^0, i = 1, \dots, N\}$:
         $w_i^0 = \begin{cases} \frac{1}{N^+}, \text{if } O_i \text{ is a positive example} \\ \frac{1}{N^-}, \text{otherwise.} \end{cases}$
     where $N^+$ and $N^-$ are, respectively, the number of positive and negative
     examples.
  2. for $t=1, \dots, T$ do
         (a) Train one weak classifier $h_k, k = 1 \dots K$ for each
             feature vector $v_k$, using the training set $S_C$
             sampled according to the probability distribution
             $w_t = \{w_i^t, i = 1, \dots, N\}$.
         (b) Choose the hypothesis $h_t$ with the lowest classification error $\epsilon_t$
         (c) Update the sample weights: $w_i^{t+1} = \frac{1}{Z_t} w_i^t e^{-\alpha_t h_t(O_i) \cdot y_i}$
             where $h_t(O_i) \in \{+1, -1\}$ wether $O_i$ is correctly or incorrectly
             classified by the weak hypothesis $h_t$,
             $\alpha_t = 0.5 \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$, and $Z_t$ is a normalizing constant
             so that $w_{t+1}$ is a distribution.
     end
  3. Final classifier: $f_C(O) = \sum_{t=1}^{T} \alpha_t h_t(O)$
```

The sample weights w_i^t, $i = 1, \dots, N$, $t = 1, \dots, T$ are very important; at step t, the weights of the samples with high classification error at step $t - 1$ is increased, while the weights of samples with smaller classification error is decreased (Algorithm 1). This will let the classifier at step t focus on difficult samples which have not been correctly classified in the previous step. The output of the strong classifier can be interpreted as the posterior probability of a class C given the shape O and it is given by:

$$P(C|O) = \frac{e^{f_C(O)}}{e^{f_C(O)} + e^{-f_C(O)}}. \tag{1}$$

The AdaBoost algorithm requires two parameters to tune: the type of weak classifier, and the maximum number of iterations. The weak classifier is required to achieve better classification than random. We experimented with the decision stumps and Least Mean Squares (LMS) classifier for their simplicity. The parameter T can be set such that $E[f_C]$, the upper bound of the classification error on the training data of the strong classifier f_C, is less than a threshold θ. In our experiments we found that a value of T between 20 and 50 is sufficient to achieve an upper bound of the classification error on the training set less than 1.0%.

Building the training set

We use as positive and negative examples for our training set the relevant and nonrelevant models provided in the Princeton Shape Benchmark (PSB) classification. For example, to build a strong classifier that learns the decision boundary between the *biped human* objects and *non-biped human* objects, the positive examples are set to all models that belong to the class *biped human*, while the negative examples are the remaining models in the database. The PSB is provided with a train and test classifications. We use the train classification to train our classification and the test classification to assess the performance of the classification and retrieval.

4.2 Generalization to multiple classes

Two straightforward extension schemes are the one-vs-all classifier and the pairwise classifier (Hao & Luo, 2006). The pairwise classifier uses $L(L-1)/2$ binary classifiers where L is the number of classes in the training set, to separate every two classes. A voting scheme at the end is used to determine the correct classification. With the one-vs-all classifier, L binary classifiers are trained, each of which is able to distinguish one class from all the others. The pairwise classifier has a smaller area of confusion in the feature space compared to the one-vs-all. However, the number of the required binary classifiers increases quadratically with the number of classes in the database, while the one-vs-all increases linearly.

In our implementation we used a one-vs-all classifier for its simplicity. The output of the training stage is a set of L binary classifiers, where L is the number of classes in the database. Given a query model Q each binary classifier will return a vote for a certain class. We use the positive votes to construct the set of candidate classes to which the query Q may belong. Notice that when a new 3D model or a new class of models are added to the database, only the classifier that corresponds to the model's class that needs training.

It is important to outline that the algorithm is data-driven that is different classifiers are obtained when given a different classification of the data. This allows to capture the semantics of the data. Furthermore, existing 3D model collections are often provided with multiple classifications. We plan in the future to extend the framework to handle hierarchical classifications of the data.

```
Algorithm 2: One-vs-all extension of binary AdaBoost for multi-class problem.

Input:  Training set $S_{C_l} = \{(V_i^l, y_i^l), i = 1 \dots N\}, l = 1, \dots, L$, is the class
        index and $V_i^l = \{\vec{v}^l_1, \dots, \vec{v}^l_K\}$ is a large set of $K$ features computed
        from the 3D object $O_i$, and $y_i^l \in \{+1, -1\}$ is the targeted classification
        of $O_i$.
Output: $L$ binary decision functions $f_{C_l}$, such that, $f_{C_l}(O) > 0$ is $C_l$ is a
        candidate class for the 3D model $O$, and $f_{C_l}(O) < 0$ otherwise.

for $l=1, \dots, L$ do
    Train one strong binary classifier $C_l$, using Algorithm 1.
        $f_{C_l}(O) > 0$ if $O \in C_l$, and $f_{C_l}(O) < 0$ otherwise
end
Collect the binary classifiers $\mathcal{C} = \{C_l, l = 1, \dots, L\}$.
```

4.3 Interpretation of the selected features

Boosting algorithm can be used as a feature selection and combination technique. Each iteration learns a new weak classifier that is based on the most discriminative feature according to the probability distribution of the training data. In the case of LFD, the selected feature is the descriptor of a 2D projection of the 3D model. Therefore, by adopting a Boosting approach we provide a tool for best view selection and view ordering based on their ability to discriminate the shapes of a certain class from the other classes in the database. Here we assume that the quality of a view is quantified as its discrimination ability, i.e, the ability of the 2D view to discriminate the shape from other shapes that belong to different classes.

The interpretation of the weak classifier may differ according to the type of descriptor used for training. In the case of the GEDT, which computes the restriction of the shape to concentric sphere, the selected feature can be seen as the radius of the concentric sphere on which the most important features of the class lie. Furthermore, the weight of each weak classifier can be considered as a measure of the saliency of the selected feature. Recall also that AdaBoost is a stochastic approach. Therefore, different runs of the algorithm on the same data will generate different sets of selected features. This is the case when the problem has many solutions (local optima). At each run it finds a different solution but with similar performance.

Figure 2 shows the top-best views selected with our algorithm. We can see that the important features of each class of shapes are visible from the selected views. This shows first that the selected views are consistent across all models of a same class, and the selected views are visually plausible. Hence, boosting captures some high semantic features of the data set. Best view selection has many applications in Computer Graphics and also online browsing of digital media contents. The framework we proposed provides an easy method to achieve this. We plan in the future to evaluate the quality of the selected views compared to other algorithms (Lee et al., 2005; Yamauchi et al., 2006).

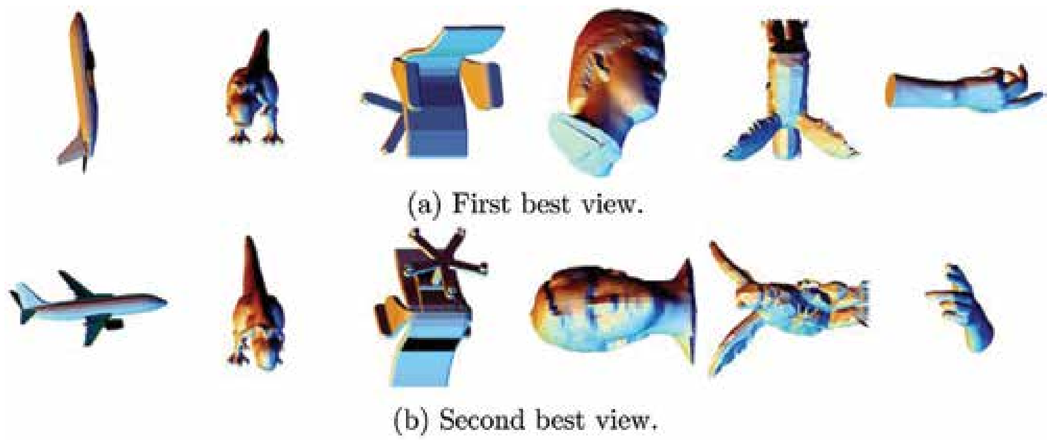


Fig. 2. Boosted LFD descriptor allows for automatic best-view selection. The first and second rows show respectively the first and second best views of objects belonging to different classes of shapes. Automatic best view selection can be used for visual browsing of large collections of 3D models.

4.4 Combining heterogeneous features

One important property of the developed classification algorithm is its ability to combine heterogeneous features in a straightforward manner; at each step of the training process, a set of weak classifiers are trained on the features, one-per basic feature, and the one with minimum training error is picked. Therefore, the features are considered independently. Although this may ignore possible correlations between basic features, it allows however to handle features of different types. We use this property to combine heterogeneous features for efficient classification.

5. Query processing

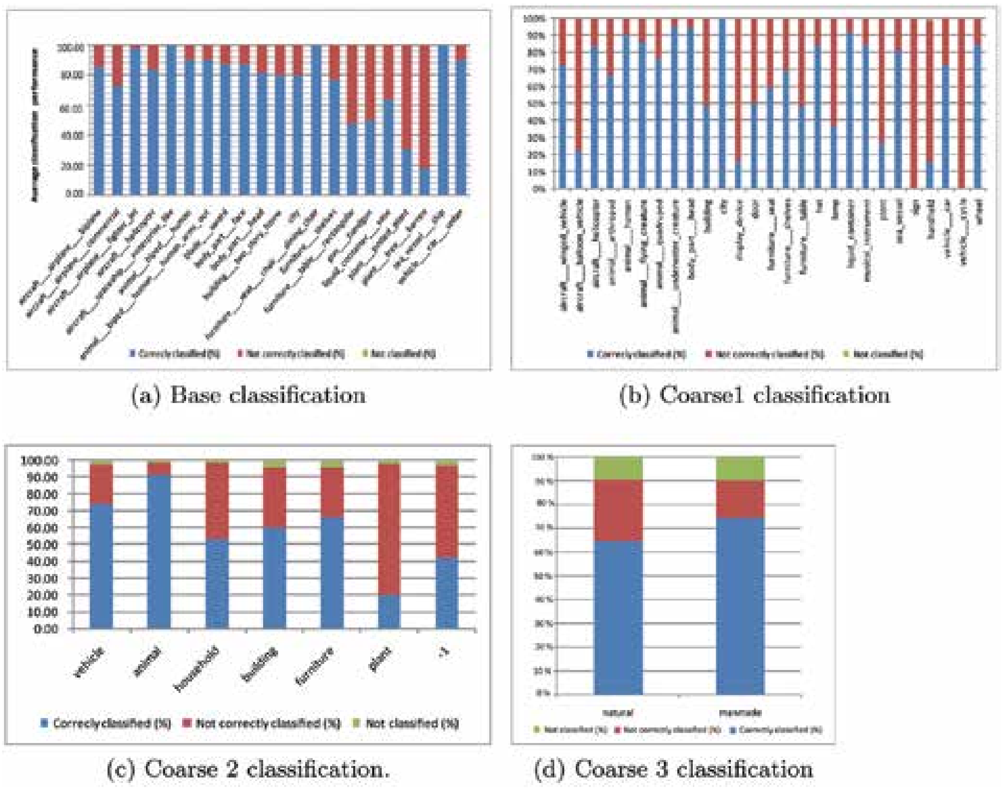
At the run time, the user specifies a query Q and seeks either to classify it into one of the shape classes (classification), or retrieve models in the database that are most similar to the query (retrieval).
To classify the query Q, we compute a set of M feature vectors (LFD and GEDT descriptors in our case) in the same manner as in the training stage (Section ??). Then we let each binary classifier $\mathcal{C}_l$ vote for a the class C_l, $l = 1, \ldots, L$. The candidate classes are determined by the classifiers that have positive response to the query Q. We build the candidate classes set by collecting the indices of classes whose classifiers gave positive response, and we order them in descending order of the class posterior probabilities given in Equation 1.
We perform the retrieval in two steps combining classification and search: first we find the candidate classes C_i to which the query Q may belong. Then, we run a search operation inside each candidate class by computing the similarity between the query Q and every model in the candidate class C_i. The 3D models of the candidate classes are sorted according to their similarity to the query model. We return one ranked list per class. The ranked lists are merged to form the k-best matches to the query. Here we use only the salient features of the class C_i, and the matching is performed only on a subset of the entire database. This reduces significantly the computation time.
Search inside classes requires the use of a distance function which measures the distance between the descriptor of the query and the descriptors of the class's shapes. In our implementation we used the Euclidean distance when working with a single descriptor type, i.e., LFD or GEDT. When using heterogeneous features however(ex. LFD-GEDT), the descriptor with larger scale will have higher impact on the Euclidean distance. To overcome this limitation we modify slightly the distance measure as follows; first we compute the Euclidean distance between the query model and the candidate models using each descriptor independently. The final distance is then taken as the minimum over the computed distances.
Examples of retrieval results are shown in Figures 4 and 5 with queries that are not part of the database. In these examples the query models (first column) do not belong to the database and have not been used during the training phase.

6. Results

To evaluate the performance of the proposed framework for 3D model classification and retrieval, we we use the Princeton Shape Benchmark (PSB) Shilane et al. (2004) as a ground truth, and the Shape Retrieval Evaluation Contest (SHREC2006) (Veltkamp et al., 2006) query set and performance evaluation tools. The Princeton Shape Benchmark contains 1814 polygon soup models, divided into the training set (907 models) and the test set (907 models). Every set contains four classification levels; the base train classification contains 129 classes while the coarsest classification (coarse3) contains two classes: man-made vs. natural objects.

6.1 Classification performance

Figure 3 summarizes the classification performance of the developed AdaBoost classifier. In this figure, the average classification performance is the ratio between the number of correctly classified models of a class *C* to the total number of models in the class. We see that, for the coarse3 classification (Figure 3-(d)), which contains only two classes with very high shape variability within each class, the classification performance is at 65.3% for natural shape and 73% for man-made models. This clearly proves that the training procedure captures efficiently the semantic concepts of the shape classes and generalizes relatively well to unseen samples.



(a) Base classification (b) Coarse1 classification

(c) Coarse 2 classification. (d) Coarse 3 classification

Fig. 3. Average classification performance of the Boosted-LFD for each class of shapes in the test set of the Princeton Shape Benchmark. Class labeled by (-1) contains models that cannot be classified to any of the other classes.

The performance on the other classification levels: base, coarse1 and coarse2 are shown in Figure 3-(a),(b) and (c). In this experiments we show only the classification results on the classes of the test set that exist in the training set. On the base classification (Figure 3-(a)), we can see that the classifiers achieve 100% classification performance on *space ship entreprise* *like, dining chair* and *sea vessel*. The worst performance is on the *plant tree* models. This is probably because of the high shape variability within the class , which cannot be captured by the LFD descriptors.

6.2 Retrieval performance

To evaluate the retrieval performance we train our classifier with the entire base classification (train and test sets) of the PSB. This classification contains 160 shape categories with varying number of samples in each class. For testing, we use the 30 queries of the SHREC2006. Each query contains a set of highly relevant and relevant models in the database. Recall that these queries do not belong the database, and therefore, have not been used during the training of the AdaBoost classifiers. We measure the retrieval performance using the SHREC2006 tools and compare to the other descriptors used in contest.

Tables 1, 2, and 3 summarize the performance of our descriptors on the mean average precision, mean first tier, mean second tier, dynamic average recall, mean normalized cumulative gain (MNCG), and the mean normalized discounted cumulative gain (MNDCG) measures. We tested the GEDT and LFD descriptors without boosting, the Boosted-GEDT and Boosted-LFD (i.e., the GEDT and LFD descriptors after boosting), and combination of LFD and GEDT denoted by Boosted-LFD-GEDT.

We can see first that the boosted versions of the LFD and GEDT algorithms perform much better than before boosting. This confirms that learning the salient features of the data by the mean of supervised learning improves the performance of the descriptors as it captures the semantic structure of the database to query. Although we tested only the LFD and GEDT, our approach is more general and it can be applied to other types of descriptors.

The second observation is that the Boosted-LFD-GEDT descriptor outperforms the Boosted-LFD and Boosted-GEDT in most of the measures. This shows that combining different types of features precision as well as the the mean dynamic average recall of the retrieval algorithm. In our implementation we used a simple similarity measure for intra-class search for the combined descriptor. We believe that there is a window for improvement by investigating more elaborated similarity measures.

Finally, Figures 4 and 5 some retrieval results of the Boosted-LFD and Boosted-GEDT descriptors. We use the SHREC2006 queries (first column) and we show the top-10 best matches. Notice that for some queries (the dolphine for the Boosted-LFD and the horse for the Boosted-GEDT), the algorithm returned less then 10 results. This is an important property of our algorithm: it is able to say whether a model is relevant to the query or not and therefore discard irrelevant models from the retrieval list.

7. Conclusion

We proposed in this chapter a new framework for 3D model retrieval based on an off-line learning of the most salient features of the shapes. By using a boosting approach we are able to use a large set of features, which can be heterogeneous, in order to capture the high-level semantic concepts of different shape classes. The retrieval process is a combination of classification and intra-class search. The experimental results showed that (1) the boosted descriptors outperform their non-boosted counter part, and (2) an efficient combination of descriptors of different types improves significantly the retrieval performance.

	Methods	Value
1	Boosted-LFD-GEDT	0.64
2	Boosted-GEDT	0.59
3	Boosted-LFD	0.55
4	Shilane et al. (R3)	0.53
5	Zaharia et al. (R1)	0.50
6	Makadia et al. (R2)	0.48
7	Shilane et al. (R2)	0.48
8	Makadia et al. (R1)	0.47
9	GEDT	0.35
10	LFD	0.28

(a) Mean Average Precision(highly relevant).

	Participant	Value
1	Boosted-LFD-GEDT	0.74
2	Boosted-LFD	0.67
3	Boosted-GEDT	0.60
4	Shilane et al. (R3)	0.52
5	Zaharia et al. (R1)	0.51
6	Shilane et al. (R2)	0.49
7	Makadia et al. (R2)	0.43
8	Makadia et al. (R1)	0.42
9	GEDT	0.28
10	LFD	0.22

(b) Mean Average Precision (Relevant)

	Methods	Value
1	Boosted-LFD-GEDT	61.9%
2	Boosted-GEDT	56.2%
3	Boosted-LFD	51.95%
4	Makadia et al. (R2)	44.77%
5	Makadia et al. (R1)	43.77%
6	Daras et al. (R1)	42.74%
7	Papadakis et al.(R1)	41.85%
8	Shilane et al. (R3)	40.86%
9	GEDT	34.06%
10	LFD	24.51%

(c) Mean First Tier (Highly relevant).

	Participant	Value
1	Boosted-LFD	64.89%
2	Boosted-LFD-GEDT	64.46%
3	Boosted-GEDT	56.02%
4	Makadia et al. (R2)	40.55%
5	Makadia et al. (R1)	38.78%
6	Shilane et al. (R3)	37.40%
7	Papadakis et al.(R1)	37.40%
8	Shilane et al. (R2)	37.30%
9	GEDT	27.68%
10	LFD	21.63%

(d) Mean First Tier (Relevant)

	Participant	Value
1	Boosted-LFD-GEDT	57.84%
2	Boosted-GEDT	54.75%
3	Boosted-LFD	52.39%
4	Makadia et al. (R2)	27.86%
5	Makadia et al. (R1)	26.62%
6	Daras et al. (R1)	25.66%
7	Shilane et al. (R3)	25.63%
8	Papadakis et al.(R1)	25.61%
9	GEDT	20.57%
10	LFD	15.80%

(e) Mean Second Tier (Highly Relevant)

	Participant	Value
1	Boosted-LFD	64.50%
2	Boosted-LFD-GEDT	61.58%
3	Boosted-GEDT	55.70%
4	Shilane et al. (R2)	26.58%
5	Shilane et al. (R3)	26.26%
6	Makadia et al. (R2)	25.22%
7	Zaharia et al. (R1)	24.63%
8	Papadakis et al.(R1)	24.24%
9	GEDT	18.21%
10	LFD	14.32%

(f) Mean Second Tier (Relevant)

Table 1. Mean Average precision, Mean First Tier and Second Tier performance.

As future work, there are many avenues for improvements. First, most of existing 3D model repositories are often provided with a hierarchical classification. We plan to extend our framework to handle such structure of the data as well as fuzzy classification, since in nature a same model may belong to several categories simultaneously. Also we plan to investigate on the meaning of the selected feature space for each shape class and extend the framework to the problem of building creative prototypes of 3D models. The prototype should capture the high-level semantic features of the class.

8. Acknowledgement

The implementation of the GEDT descriptor has been kindly provided by Philip Shilane. Models that appear in this chapter are from the Princeton Shape Benchmark, courtesy of the Shape Analysis Group of the Princeton University. The query models and the evaluation tools used in this chapter are part of SHREC2006: the first Shape Retrieval Evaluation Contest (http://www.aimatshape.net/event/SHREC/shrec06).
This research is supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) program *Promotion of Environmental Improvement for Independence of Young Researchers* under the Special Coordination Funds for Promoting Science and Technology.

9. References

Ankerst, M., Kastenmoller, G., Kriegel, H.-P. and Seidl, T. (1999), Nearest neighbor classification in 3D protein databases, *in* 'the Seventh International Conference on Intelligent Systems for Molecular Biology', AAAI Press, pp. 34–43.

Biasotti, S., Marini, S., Spagnuolo, M. and Falcidieno, B. (2006), "Sub-part correspondence by structural descriptors of 3D shapes.", *Computer-Aided Design* , Vol. 38, pp. 1002–1019.

Chen, D.-Y., Tian, X.-P., Shen, Y.-T. and Ouhyoung, M. (2003), "On visual similarity based 3D model retrieval.", *Computer Graphics Forum* , Vol. 22, pp. 223–232.

Funkhouser, T. A., Min, P., Kazhdan, M. M., Chen, J., Halderman, J. A., Dobkin, D. P. and Jacobs, D. P. (2003), "A search engine for 3D models.", *ACM Transactions on Graphics* , Vol. 22, pp. 83–105.

Funkhouser, T. and Shilane, P. (2006), Partial matching of 3D shapes with prioritydriven search, *in* 'SGP '06: Proceedings of the fourth Eurographics Symposium on Geometry Processing', Eurographics Association, pp. 131–14.

Hao, W. and Luo, J. (2006), Generalized multiclass adaboost and its applications to multimedia classification, *in* 'CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop', IEEE Computer Society, p. 113.

Hilaga, M., Shinagawa, Y., Kohmura, T. and Kunii, T. L. (2001), Topology matching for fully automatic similarity estimation of 3D shapes, *in* 'Proceedings of the 28th annual conference on Computer graphics and interactive techniques', ACM Press, pp. 203–212.

Hou, S., Lou, K. and Ramani, K. (2005), "SVM-based semantic clustering and retrieval of a 3D model database", *Journal of Computer Aided Design and Application* , Vol. 2, pp. 155–164.

Hou, S. and Ramani, K. (2006), A probability-based unified 3D shape search, *in* 'European Commission International Conference on Semantic and Digital Media Technologies, Lecture notes in computer science', Vol. 4306, pp. 124–137.

Ip, C. Y., Regli, W. C., Sieger, L. and Shokoufandeh, A. (2003), Automated learning of model classifications, *in* 'SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications', ACM Press, pp. 322–327.

Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y. and Ramani, K. (2005), "Threedimensional shape searching: state-of-the-art review and future trends.", *Computer-Aided Design*, Vol. 37, pp. 509–530.

Johnson, A. (1997), Spin-Images: A Representation for 3-D Surface Matching, PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Kazhdan, M., Funkhouser, T. and Rusinkiewicz, S. (2003), Rotation invariant spherical harmonic representation of 3D shape descriptors, *in* 'SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing', pp. 156–164.

Laga, H., Takahashi, H. and Nakajima, M. (2006), Spherical wavelet descriptors for content-based 3D model retrieval, *in* 'SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)', pp. 75–85.

Lee, C. H., Varshney, A. and Jacobs, D.W. (2005), Mesh saliency, *in* 'SIGGRAPH '05: ACM SIGGRAPH 2005 Papers', ACM Press, New York, NY, USA, pp. 659–666.

Lew, M. S., Sebe, N., Djeraba, C. and Jain, R. (2006), "Content-based multimedia information retrieval: State of the art and challenges", *ACM Trans. Multimedia Comput. Commun. Appl.* , Vol. 2, ACM Press, New York, NY, USA, pp. 1–19.

M.Kortgen, G-J.Patrick, M.Novotni and R.Klein (2003), 3D shape matching with 3D shape contexts, *in* 'the 7th Central European Seminar on Computer Graphics'.

Novotni, M. and Klein, R. (2003), 3D Zernike descriptors for content based shape retrieval, *in* 'SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications', ACM Press, New York, NY, USA, pp. 216–225.

Ohbuchi, R. and Kobayashi, J. (2006), Unsupervised learning from a corpus for shapebased 3D model retrieval, *in* 'MIR '06: Proceedings of the 8th ACM international workshop on Multimedia information retrieval', ACM Press, pp. 163–172.

Ohbuchi, R., Kobayashi, J., Yamamoto, A. and Shimizu, T. (2007), Comparison of dimension reduction method for database-adaptive 3D model retrieval, *in* 'Fifth International Workshop on Adaptive Multimedia Retrieval (AMR 2007)'.

Reuter, M., Wolter, F.-E. and Peinecke, N. (2006), "Laplace-Beltrami spectra as "shape-DNA" of surfaces and solids", *Computer-Aided Design* , Vol. 38, pp. 342– 366.

Schapire, R. E. (2003), The boosting approach to machine learning: An overview., *in* 'In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu, editors, Nonlinear Estimation and Classification', Springer.

Shilane, P. and Funkhouser, T. (2006), "Selecting Distinctive 3D Shape Descriptors for Similarity Retrieval", *IEEE International Conference on Shape Modeling and Applications (SMI2006)* , Vol. 0, IEEE Computer Society, Los Alamitos, CA, USA, p. 18.

Shilane, P., Min, P., Kazhdan, M. and Funkhouser, T. (2004), The princeton shape benchmark, *in* 'SMI'04: Proceedings of the Shape Modeling International 2004 (SMI'04)', pp. 167–178.

Tangelder, J. W. and Veltkamp, R. C. (2004), A survey of content based 3D shape retrieval, *in* 'Shape Modeling International 2004, Genova, Italy', pp. 145–156.

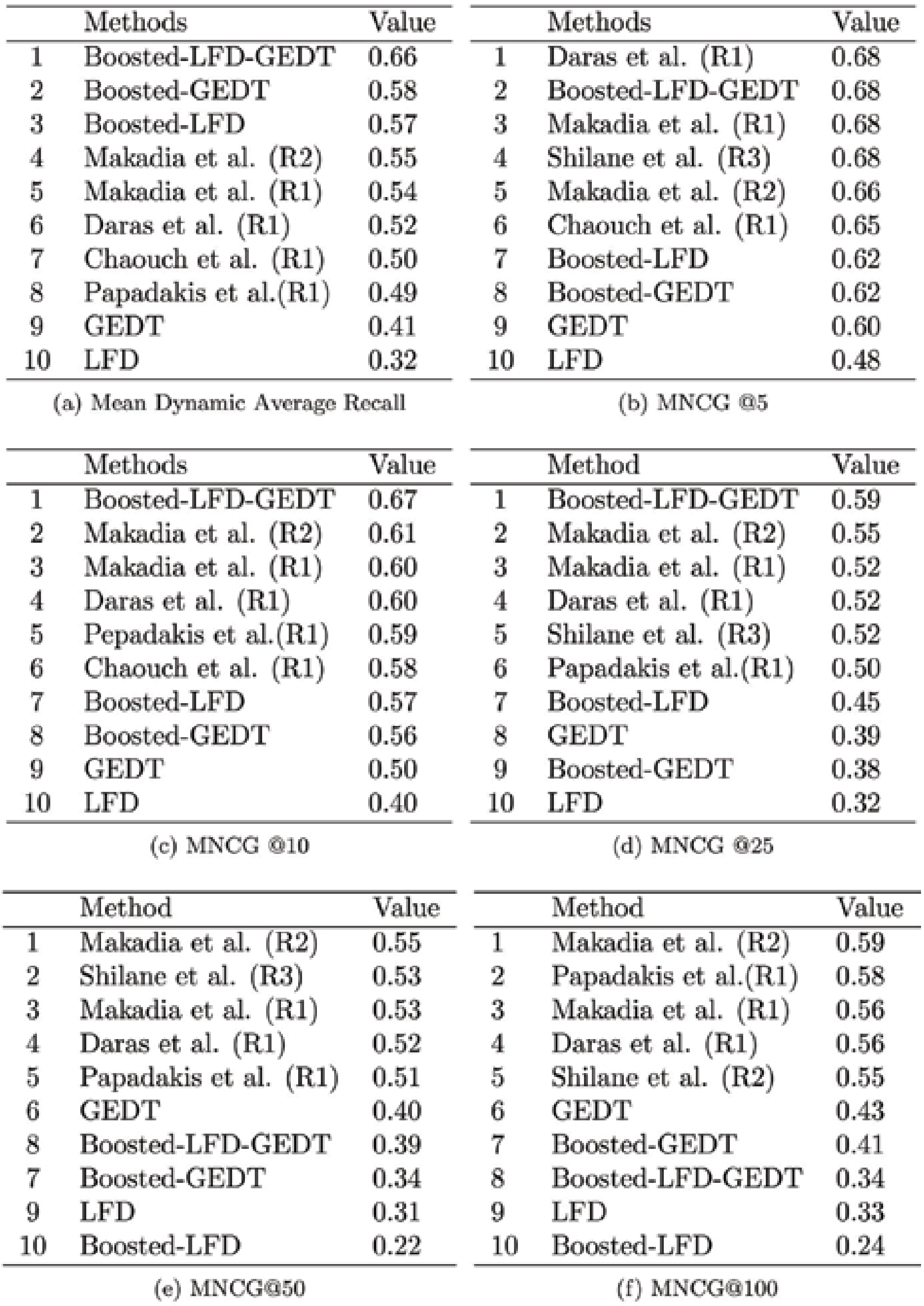
Tieu, K. and Viola, P. (2004), "Boosting image retrieval", *International Journal of Computer Vision* , Vol. 56, Kluwer Academic Publishers, Hingham, MA, USA, pp. 17–36.

T.Tung and F.Schmitt (2005), "The augmented multiresolution reeb graph approach for content-based retrieval of 3D shapes", *International Journal of Shape Modeling (IJSM)* , Vol. 11, pp. 91–120.

Veltkamp, R. C., Ruijsenaars, R., Spagnuolo, M., van Zwol, R. and ter Haar, F. (2006), SHREC2006: 3D Shape Retrieval Contest, Technical Report UU-CS- 2006-030, Department of Information and Computing Sciences, Utrecht University.

Xu, C., Tan, T., Li, S. Z., Wang, Y. and Zhong, C. (2006), Learning effective intrinsic features to boost 3D-based face recognition, *in* 'ECCV 2006, 9th European Conference on Computer Vision', Springer, pp. 416–427.

Yamauchi, H., Saleem, W., Yoshizawa, S., Karni, Z., Belyaev, A. and Seidel, H.- P. (2006), Towards stable and salient multi-view representation of 3D shapes, *in* 'Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)', p. 40.



	Methods	Value
1	Boosted-LFD-GEDT	0.66
2	Boosted-GEDT	0.58
3	Boosted-LFD	0.57
4	Makadia et al. (R2)	0.55
5	Makadia et al. (R1)	0.54
6	Daras et al. (R1)	0.52
7	Chaouch et al. (R1)	0.50
8	Papadakis et al.(R1)	0.49
9	GEDT	0.41
10	LFD	0.32

(a) Mean Dynamic Average Recall

	Methods	Value
1	Daras et al. (R1)	0.68
2	Boosted-LFD-GEDT	0.68
3	Makadia et al. (R1)	0.68
4	Shilane et al. (R3)	0.68
5	Makadia et al. (R2)	0.66
6	Chaouch et al. (R1)	0.65
7	Boosted-LFD	0.62
8	Boosted-GEDT	0.62
9	GEDT	0.60
10	LFD	0.48

(b) MNCG @5

	Methods	Value
1	Boosted-LFD-GEDT	0.67
2	Makadia et al. (R2)	0.61
3	Makadia et al. (R1)	0.60
4	Daras et al. (R1)	0.60
5	Pepadakis et al.(R1)	0.59
6	Chaouch et al. (R1)	0.58
7	Boosted-LFD	0.57
8	Boosted-GEDT	0.56
9	GEDT	0.50
10	LFD	0.40

(c) MNCG @10

	Method	Value
1	Boosted-LFD-GEDT	0.59
2	Makadia et al. (R2)	0.55
3	Makadia et al. (R1)	0.52
4	Daras et al. (R1)	0.52
5	Shilane et al. (R3)	0.52
6	Papadakis et al.(R1)	0.50
7	Boosted-LFD	0.45
8	GEDT	0.39
9	Boosted-GEDT	0.38
10	LFD	0.32

(d) MNCG @25

	Method	Value
1	Makadia et al. (R2)	0.55
2	Shilane et al. (R3)	0.53
3	Makadia et al. (R1)	0.53
4	Daras et al. (R1)	0.52
5	Papadakis et al. (R1)	0.51
6	GEDT	0.40
8	Boosted-LFD-GEDT	0.39
7	Boosted-GEDT	0.34
9	LFD	0.31
10	Boosted-LFD	0.22

(e) MNCG@50

	Method	Value
1	Makadia et al. (R2)	0.59
2	Papadakis et al.(R1)	0.58
3	Makadia et al. (R1)	0.56
4	Daras et al. (R1)	0.56
5	Shilane et al. (R2)	0.55
6	GEDT	0.43
7	Boosted-GEDT	0.41
8	Boosted-LFD-GEDT	0.34
9	LFD	0.33
10	Boosted-LFD	0.24

(f) MNCG@100

Table 2. Dynamic Average Recall and, Mean Normalized Cumulated Gain (MNCG) performance.

	Methods	Value
1	Daras et al. (R1)	0.70
2	Makadia et al. (R1)	0.69
3	Shilane et al. (R3)	0.69
4	Boosted-LFD-GEDT	0.68
5	Chaouch et al. (R1)	0.68
6	Makadia et al. (R2)	0.68
7	Boosted-LFD	0.63
8	Boosted-GEDT	0.63
9	GEDT	0.61
10	LFD	0.52

(a) MNDCG@5

	Methods	Value
1	Boosted-LFD-GEDT	0.68
2	Daras et al. (R1)	0.64
3	Makadia et al. (R1)	0.64
4	Makadia et al. (R2)	0.64
5	Chaouch et al. (R1)	0.62
6	Pepadakis et al.(R1)	0.62
7	Boosted-LFD	0.59
8	Boosted-GEDT	0.58
9	GEDT	0.54
10	LFD	0.45

(b) MNDCG@10

	Method	Value
1	Boosted-LFD-GEDT	0.60
2	Makadia et al. (R2)	0.59
3	Daras et al. (R1)	0.58
4	Makadia et al. (R1)	0.57
5	Shilane et al. (R3)	0.56
6	Papadakis et al.(R1)	0.55
7	Boosted-LFD	0.46
8	GEDT	0.45
9	Boosted-GEDT	0.41
10	LFD	0.38

(c) MNDCG@25

	Method	Value
1	Makadia et al. (R2)	0.58
2	Makadia et al. (R1)	0.57
3	Daras et al. (R1)	0.56
4	Shilane et al. (R3)	0.56
5	Papadakis et al. (R1)	0.54
6	GEDT	0.45
7	Boosted-LFD-GEDT	0.41
8	LFD	0.37
9	Boosted-GEDT	0.33
10	Boosted-LFD	0.21

(d) MNDCG@50

	Method	Value
1	Makadia et al. (R2)	0.59
2	Makadia et al. (R1)	0.57
3	Daras et al. (R1)	0.57
4	Papadakis et al.(R1)	0.56
5	Shilane et al. (R3)	0.55
6	GEDT	0.45
7	LFD	0.37
8	Boosted-GEDT	0.36
9	Boosted-LFD-GEDT	0.36
10	Boosted-LFD	0.22

(e) MNDCG@100

Table 3. Mean Normalized Discounted Cumulated Gain (MNDCG) performance.

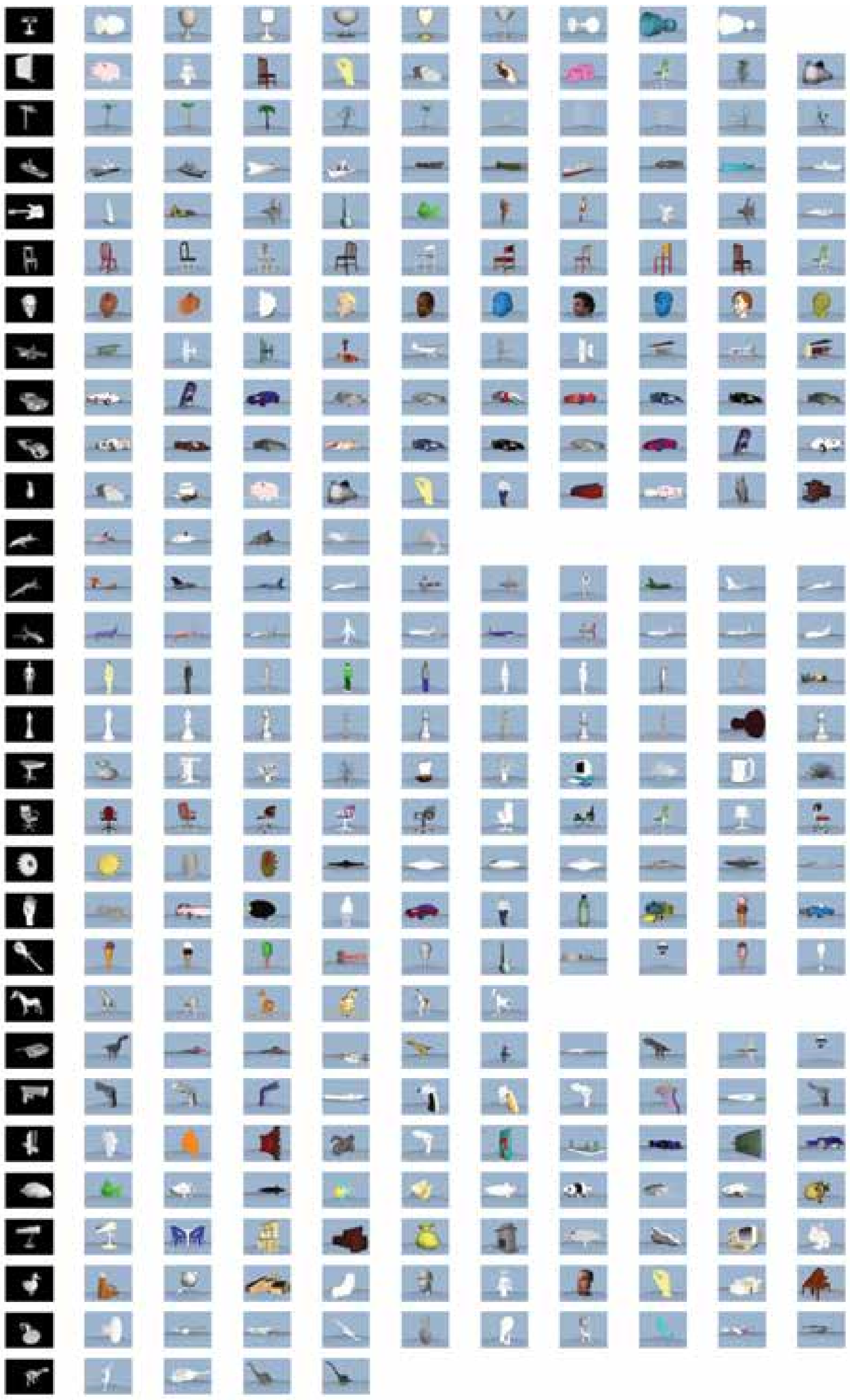


Fig. 4. Some retrieval results using the Boosted-LFD Descriptors. The models in the first column are used as query models2.2The 10-best matches are displayed.

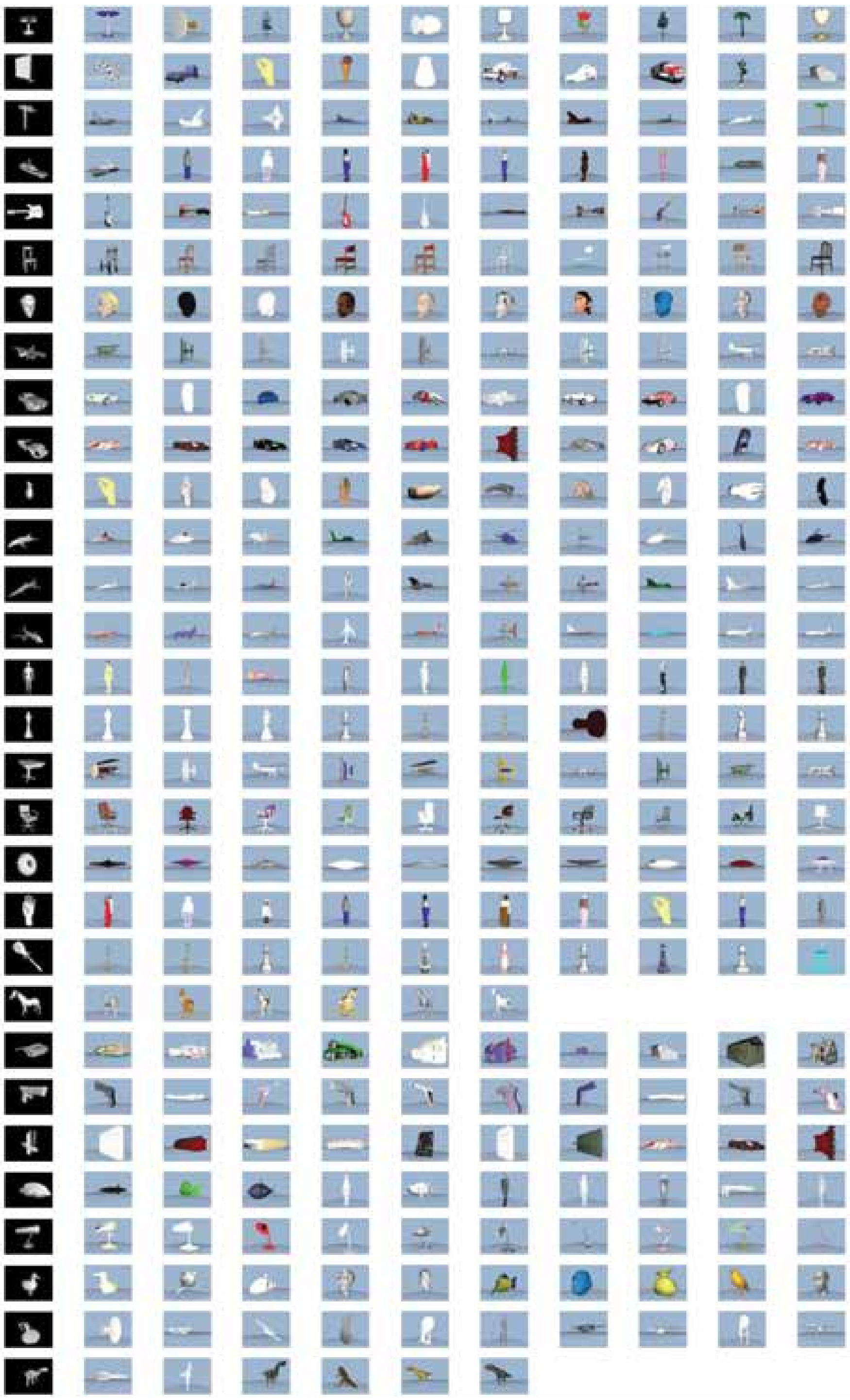


Fig. 5. Retrieval results using the Boosted-GEDT Descriptors. The models in the first column are used as query models. T2h3e 10-best matches are displayed.

Performance Analysis of Hybrid Non-Supervised & Supervised Learning Techniques Applied to the Classification of Faults in Energy Transport Systems

Jhon Albeiro Calderón[1], Germán Zapata Madrigal[2] and Demetrio A. Ovalle Carranza[3]
[1]Interconexión Eléctrica S.A. E.S.P.,
[2]Electrical and Mechanics Engineering Department,
[3]Computer Science Department, National University of Colombia – Campus Medellin
Colombia

1. Introduction

Most of the power systems protection techniques are related to the definition of system states by means of the identification of patterns from waveform of voltage and associated current. This means that the development of an adaptive protection can essentially be treated as a problem about classification/recognition of patterns (Song et al., 1997). Nevertheless, because the main causes of faults and the operation of nonlinear devices under certain conditions of fault, the methods of recognition of conventional patterns are unsatisfactory in some applications, particularly, in the case of high complexity electrical systems. In this sense, neural networks play an important role due to their unique ability of mapping nonlinear relations.

Some successful applications of neural networks in the area of electrical engineering (Song et al., 1996), (Dillon & Niebur, 1996) have demonstrated that they can be used like an alternative method to solve certain problems of great complexity where the conventional techniques have experienced difficulties. Nevertheless, when giving a glance to the different applications of neural networks to electric power systems, it is clear that almost all the developments that have been carried out are based on the multi-layers perceptron with retro-propagation learning algorithms (BP). Although, BP can provide very compact distributed representations of complex data sets, it has some disadvantages such as the following: it exhibits slow learning, it requires great sets of training, they easily fall in local minimums, and in general it shows little robustness (Song et al., 1997).

Another type of learning is the non-supervised one that surrounds the learning of patterns without a target. A typical non-supervised learning network is the Self-Organized Mapping (SOMs) developed by Teuvo Kohonen. A SOM network has the advantage of fast learning and small sets of training. Nevertheless, due to the absence of an output "truth" layer in the SOM, its use is not recommendable for the classification of patterns. Instead, it is used as an

initial procedure ("front-end") to an output layer with a supervised training, that is, combined non-supervised/supervised learning.
The networks that combine non-supervised and supervised learning have the powerful ability to organize any complexity of highly nonlinear patterns recognition problem. This type of neural network is insensitive to noise due to the low internal dimensional representation (Song et al., 1996). Based on this kind of characteristics the present research developed a hybrid model entitled Artificial Intelligence Adaptive Model (AIAM) (Calderón, 2007).
Next, it will be initially described the basic functionality of the several models analyzed in the research, the respective main results, and finally the AIAM model and conclusions.

2. Neural networks models

With the purpose of selecting the most appropriate neural network model to be used for the classification of faults in an Electrical Power System (EPS) an exploration of alternatives on models of neural networks was carried out based on the state-of-the-art of the subject (El-Sharkawi & Niebur, 1996), (Aggarwal & Song, 1997), (Aggarwal & Song, 1998a), (Aggarwal & Song, 1998b), (Kezunovic, 1997), (Dalstein & Kulicke, 1995), (Keerthipala et al, 1997), (Sidhu & Mitai, 2000), (Fernandez & Ghonaim, 2002), (Dalstein et al, 1996), (Zahra et al, 2000), (Ranaweera, 1994), (Oleskovicz et al., 2001), (Song et al, 1997), (Song et al, 1996), (Dillon & Niebur, 1996), (Dillon & Niebur, 1999),(Badrul et al., 1996).
Next, four important classifiers, based on neural networks, will be briefly described. Special emphasis was placed on the basic principles and differences, instead of a detailed description itself.

2.1 Back-Propagation classifier (BP)

BP classifiers are the most popular and widely applied neural networks. They train with supervision using the descending gradient algorithm to diminish the error between the real exits and the wished exits of the network.
In Fig. 1. the general architecture of this type of network is illustrated.

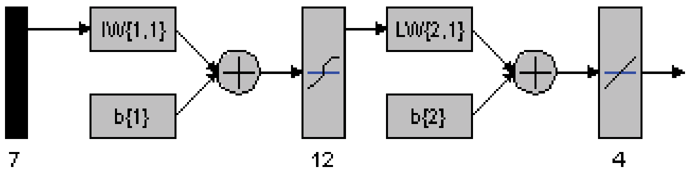


Fig. 1. General architecture used by the model of retro-propagation training. (Matlab educational license).

Many articles provide good introductions to the methods and successful applications of this type of neural networks applied to the power systems. Nevertheless, in general, most of the BP classifiers are (1) of prolonged training time; (2) of difficult selection for the optimal size, and (3) potentially with tendency to be caught in a local minimum (Song et al., 1996).
For this reason, improvements have been developed in recent years, particularly in the aspect concerning the learning process. In this sense, it is valuable to mention the fuzzy algorithms of controlled learning and the training based on genetic algorithms.

2.2 Feature Mapping classifier (FM)

One of the most important algorithms of non-supervised learning is the Self-Organized Feature Mapping (SOFM) proposed by Kohonen shown in Fig. 2. The SOFM is used to map non-supervised input vectors in a bi-dimensional space where the vectors are self-organized in groups that represent the different types.

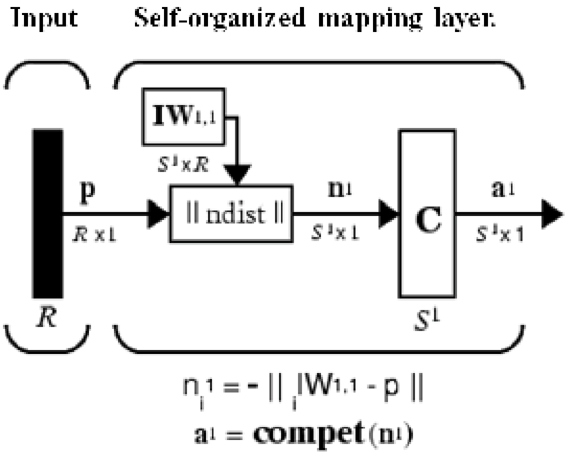


Fig. 2. General architecture used by the Kohonen model of SOMF. (Matlab educational license).

The SOFM learns to classify input vectors according to the form that they are grouped in the input space. This method differs from the competitive method of layers in which the neighboring neurons in the SOFM learn to recognize also adjacent sections in the input space. Thus, the self-organized maps learn so much the distribution (as the competitive method of layers makes it) as well as the topology of the input vectors that train. Neurons in the layer of a SOFM are originally organized in physical positions according to a topological function. The distances between neurons are calculated from their positions with a distance function.

In these networks there is no target for the error evaluation. That is, the learning of the synaptic weights is non-supervised, which means that, under the presentation of new input vectors, the network dynamically determines these weights, in such a way that, input vectors that are closely related will excite neurons that are closely grouped (Badrul et al., 1996). It is able to separate data in a specified number of categories and therefore able to act like a classifier. In the Kohonen network there are only two layers: an input layer where the patterns of the variables are placed and an output layer that has a neuron for each possible category or type.

2.3 Radial Base Function classifier (RBF)

The construction of a RBF in its most basic form considers three layers entirely different, as in Fig. 3. The first layer consists of the input nodes. The second layer is composed by the denominated Kernel nodes (base radial layer) which functions are different from those of a BP network. The Kernel nodes based on the radial base functions calculate symmetrical functions which are a maximum when the input is near the centroid of a node. The output nodes are simple sums.

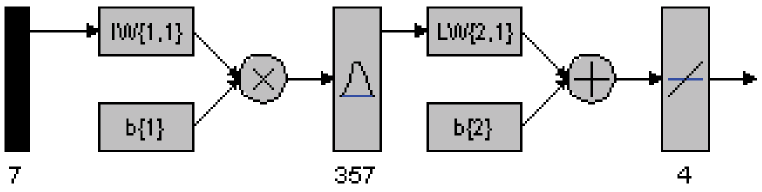


Fig. 3. General architecture used by the RBF model. (Matlab educational license).

This particular architecture of RBF has been proven to improve the training time but at the expense of considering many nodes in the radial base layer and connections of weights (in critical cases the number of neurons of this layer could get to be equal to the number of training samples, that is to say, a neuron per input pattern).

2.4 Vector Quantification Learning classifier (LVQ)

The Vector Quantification Learning network (LVQ) is a form of adaptive classifier which structure is shown in Fig. 4. This classifier requires a final stage of supervised training to improve its performance. LVQ contains an input layer, a Kohonen layer and the output layer. The number of nodes of the entrance layer is equal to the number of entrance parameters. The number of nodes of the Kohonen layer is based on the number of input vectors in the training data. The output layer contains a node for each type.

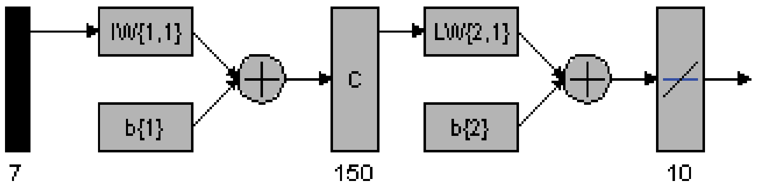


Fig. 4. General architecture used by the LVQ model. (Matlab educational license).

Based on the analysis of the previous neural networks models the research was oriented into two ways:

- To complement the neural model BP with a learning method that allowed improving the generalization and the resulting classification error. In order to do this the Bayesian Regularization methodology (BR) described in (Foresee and Hagan, 1997), (Hagan et al, 2002), (MacKay, 1998), was used.
- The search of an Adaptive model that would take advantage of the kindnesses of the combination of the non-supervised learning with the supervised learning but, as well as looking for to fix the weaknesses found in LVQ and RBF methods. To get this, the doctorate thesis (Vasilic, 2004), consisting in the Adaptive Resonance Theory (ART), was used as the starting point.

2.5 BP neural network model with Bayesian regularization

Taking into account the considerations of the previous concept, it was implemented the performance evaluation of the neural network BP incorporating additional training techniques to improve its performance. With the purpose of obtaining a high capacity of generalization of the network, it was considered (Foresee and Hagan, 1997), (Hagan et al, 2002), (MacKay, 1998), the approach known as the Bayesian regularization in which the weights of the network are assumed as random variables with specific distributions so that the parameters of stabilization get associated to the unknown variances connected to these distributions. In this way, it is possible to consider the parameters using technical statistics. A more a detailed description of this approach and its combination with other techniques can be found in (Foresee & Hagan, 1997).

In MATLAB it is possible to use this methodology by means of the trainbr algorithm that can be established as argument at the time of defining the network by means of the function newff.

For the detection and classification of the fault, a feed-forward network was used with a single hidden layer of s neurons (Foresee & Hagan, 1997), (Hagan et al, 2002), (MacKay, 1998). 7 neurons were considered for the input layer that corresponds to the rms values of the voltages and currents of the 3 phases, plus the sequence zero current. For the output layer, 4 neurons were considered corresponding to the binary values that indicate the failed phase (the 3 first bits) and if it is or not grounded (last bit). In this case, the used value of s was of 12 (value that was obtained after doing different tests of verification trying to diminish the resulting error, but at the same time guaranteeing a suitable level of generalization).

The general model of this network it is shown in Fig. 5. The functions of activation of MATLAB Tansig were used in the hidden layer and in the output layer the linear transference Purelin.

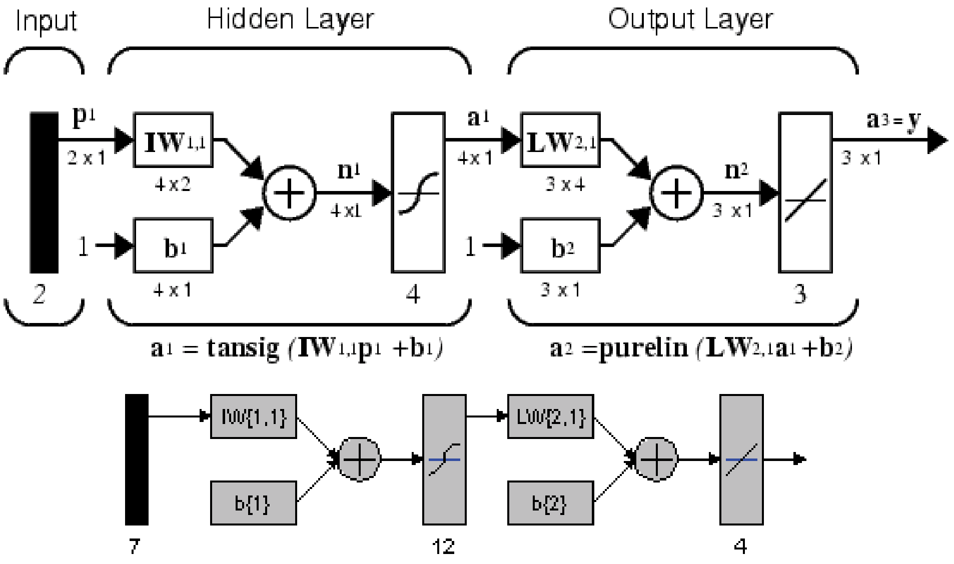


Fig. 5. Diagram of the classifier algorithm and the used neural network architecture BR. (Matlab educational license).

2.6 ART model (adaptive resonance theory)

ART Model does not have a defined typical structure with a specified number of neurons. Instead, it is made up of an adaptive structure with auto-evolving neurons. The structure solely depends on the characteristics and order of presentation of the patterns in the input data set. In Fig. 6. the diagram of the complete procedure used for the training of the neuronal network type ART is explained in (Vasilic, 2004).

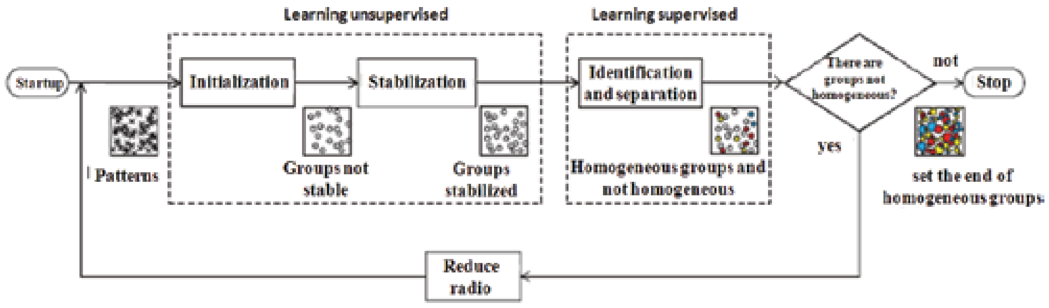


Fig. 6. Combined learning of Supervised and Non- supervised Neural Networks (Vasilic, 2004).

The training consists in numerous iterations in the stages of supervised and non-supervised learning, suitably combined to obtain a maximum efficiency. Groups of similar patterns lay in groups, defined as hyper-spheres in a multidimensional space, where the dimension of the space is determined by means of the length of the input patterns. The neural network initially uses non-supervised learning with input patterns not tagged in order to form unstable fugitive groups. This is an attempt to discover the patterns density by means of getting them in groups to consider prototypes of groups that can serve as prototypes of the typical input patterns. The category tags are assigned later on to the groups during the stage of supervised learning. The tuning parameter called "threshold of monitoring" or "radio", controls the size of the group and therefore the number of generated groups, and it is consecutively reduced during the iterations. If the monitoring threshold is high, many different patterns within a group can then be incorporated, and this generates a small number of heavy groups. If the monitoring threshold is low, they only activate the same group patterns that are very similar, and this generates a great number of fine groups. Subsequent to the training, the centers of the groups serve as typical neurons of the neural network. The structure of prototypes only depends on the density of the input patterns. Each training pattern has been located in only one group, at the same time as each group contains one or more similar input patterns. A prototype is centrally located in the respective group, and it is either identical to one of the real patterns or identical to a synthesized prototype of the found patterns. A category tag is assigned to each group symbolizing a type of groups with a symbolic characteristic, meaning that each group belongs to one of the existing categories. The number of categories corresponds to the desired number of outputs of the neural network. Finally, during the implementation phase of the trained network, the distance between each new pattern and the established prototypes is calculated, and using a fuzzy classifier of the nearest neighbors, it is assigned the most representative category to the pattern in evaluation.

In Fig. 7 it is shown the steps carried out for the mapping of the input space in categories decision regions using algorithm ART2 proposed by (Vasilic, 2004). Initially, using non-supervised/supervised learning, the space of the training patterns is transferred within a level of initial abstraction that contains a set of groups with the corresponding prototypes, size and category. Later, the groups are fuzzyficated and transformed into an intermediate level of abstraction. Finally, by means of the defuzzyfication, regions of refined decision are established, and a level of final abstraction is obtained.

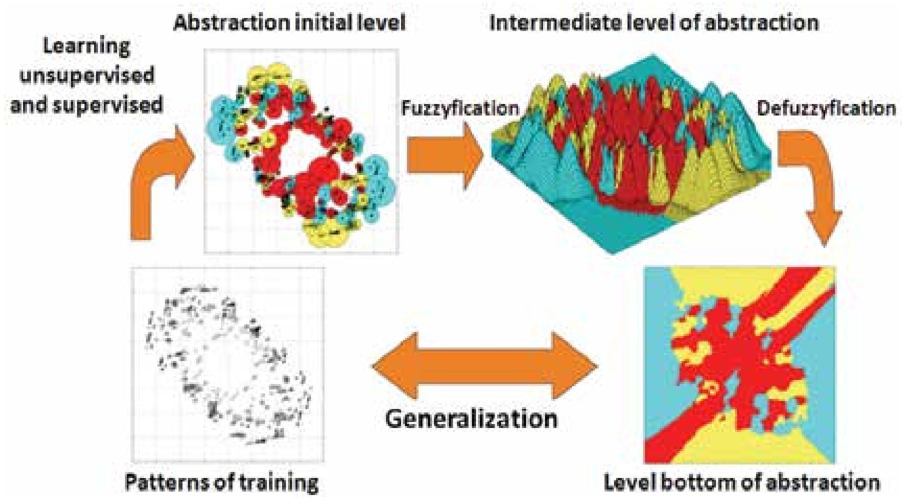

Abstraction initial level
Intermediate level of abstraction
Learning unsupervised and supervised
Fuzzyfication
Defuzzyfication
Generalization
Patterns of training
Level bottom of abstraction

Fig. 7. Mapping of the patterns space using supervise/non-supervised training through the ART2 algorithm. (Vasilic, 2004).

It is observed in Fig. 8, the classification obtained from homogenous groups of the same category using the ART1 methodology (which allows the overlapping of groups and the presence of elements in this zone) and which it is obtained by means of the ART2 methodology (which reduces the radios until no longer elements in the zones of overlaps are present). By means of this modification, the model ART2 tries to improve its performance in relation to the classification error, since it reduces the ambiguity that appears when there are elements in the zones of overlaps that could produce erroneous classification of some pattern. However, it is important to outline that as the radios get more restricted, the network loses capacity of generalization. The final ideal model is a commitment between the needs of precision in the classification with the generalization capacity of the model.

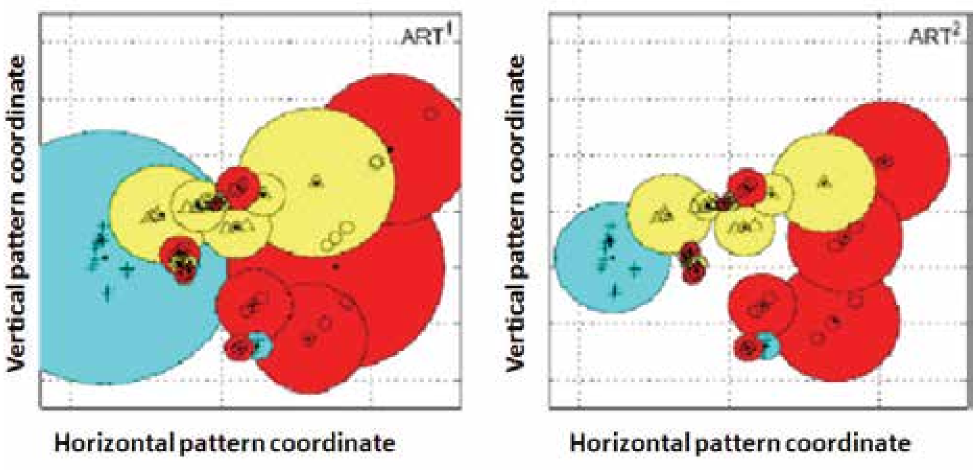


Fig. 8. Comparison between ART1 and ART2 models. (Vasilic, 2004).

3. Art2 model improved

As a contribution of the current research, the model ART2 of (Vasilic, 2004), was improved by introducing a formal methodology for the "reduction of the radios" and by introducing a novel concept denominated "learning on line".

3.1 Formal methodology for reduction of radios

With the purpose of trying to solve the ambiguity that appears when clusters belonging to different categories are present, and with a region of non-zero intersection among them (that is to say, that there are a certain number of training patterns in that region), (Vasilic, 2004), proposes a solution to this problem (ART2) consisting of introducing some rule during the phase of supervised training to construct homogenous clusters that covers solely patterns of exactly one category (valid rule of homogenous intersected clusters), allowing regions of intersection between clusters of different categories as long as patterns do not exist in those regions.

It is outlined in (Vasilic, 2004), the ART2 methodology expressed in natural language, but it is not formally described the algorithm, nor details on its implementation provided. In the current research work, a formal proposal was developed to carry out the implementation of the model classification ART2 and be able to go from the obtained clusters with ART1 to the obtained clusters with ART2, as seen in Fig. 8.

Next, it is presented the procedure used and the formal description of the implemented algorithm. Initially, homogenous intersected valid cluster rule is defined and then the rules to make the reduction of the radios.

3.1.1 Homogenous intersected valid clusters rule

Let to ch be a homogenous cluster of the form [r, P, C, CP], where:

r: is the radius of cluster. r belongs to the real numbers.

P: a vector of dimension n, is the cluster prototype found with the training patterns; each input of this vector belongs to the real numbers set.

C: is the type of cluster, C pertaining to the integer numbers.

CP: is a set of vectors of dimension n where each input of each vector belongs to the real numbers, (training patterns which conform the cluster).

Let to @: [V1 x V2→R] be a function that delivers the Euclidian distance between two vectors, where V1 and V2 are vectors of dimension n.

Let to A be a finite set of homogenous clusters without training patterns in their intersection regions, A = {ch1, ch2, ch3, ..., chn}.

Let to ch1[r1,P1,C1,CP1] and ch2[r2,P2,C2,CP2] be a pair of homogenous clusters of A.

Let to M be the number of patterns in CP1, let to K be the number of patterns in CP2 .

Then:

ch1, ch2 belong to A
IF, AND ONLY IF:
(P1@P2 < r1 + r2) and (C1 ≠ C2) and
FOR EVERYTHING (cpm ϵ CP1) { P2@cpm > r2 } and
FOR EVERYTHING (cpk ϵ CP2) { P1@cpk > r1 } ; k = 1, 2, …, K ; m = 1, 2,.., M

The rule of homogenous intersected valid clusters is fulfilled if the clusters belong to A.
During the supervised training (phase of stabilization), once the homogenous clusters are obtained, it must be verified that no patterns exist inside some region of intersection among the clusters found, that is to say, that the homogenous intersected valid clusters rule is fulfilled; if this rule is not accomplished, the reduction of radios rule should be applied, analyzing the possibility of reducing the radius of any of both clusters in conflict, or of both, if it is necessary, assuring not to exclude any pattern, and having this way the rule fulfilled, to later on add these new optimized clusters to the final set of clusters.

3.1.2 Reduction of radios rule

Let to min (): [Rn → R] be a function that receives a set of real numbers and delivers the minor.
Le to max() : [Rn → R] be a function that receives a set of real numbers and delivers the major.
If ch1, ch2 does not belong to A
It is verified if patterns of ch1 in ch2 exist, and if it is possible the reduction of its radius is done.

```
1   If
2   EXIST (cpm ϵ CP1) { P2@cpm < r2 } ; m = 1, 2, …, M
3   then
4           r2max = min(P2@cpm=1, P2@cpm=2, ..., P2@cpm=M)
5                r2min = max(P2@cpk=1, P2@cpk=2, ..., P2@cpk=K)
6           si r2min > r2max
7           then
8                   r2 = r2max - (r2max - r2min)/L
```

It is verified if patterns of ch1 in ch2 exist, and if it is possible, the reduction of its radius is done.

```
1   If
2   EXIST (cpk ϵ CP2) { P1@cpk < r1 } ; k = 1, 2, …, K
3   then
4                r1max = min(P1@cpk=1, P1@cpk=2, ..., P1@cpk=k)
5         r1min = max(P1@cpm=1, P1@cpm=2, ..., P1@cpm=M)
6         si r1min < r1max
7           then
8                   r1 = r1max - (r1max – r1min)/L
```

Where L is an arbitrary constant inverse to the magnitude in which the radius is reduced. If the given restriction in line 6 is fulfilled the radius can be reduced, and add the cluster to the final set of homogenous clusters. This operation is done for all the homogenous clusters found after the stabilization, and also done against the homogenous clusters that previously have been added in set A.
In Fig. 9 to Fig. 11 it is graphically illustrated what can happen in the intersection of the clusters.
Notice that in Fig. 10 cluster 2 (yellow) cannot reduce its radius since it would exclude the most distant pattern, for this reason these patterns must be part of the following iteration in

the non-supervised & supervised training of ART. Cluster 1 can reduce its radius, and it is included in the set of the final cluster, this is illustrated in Fig. 11.

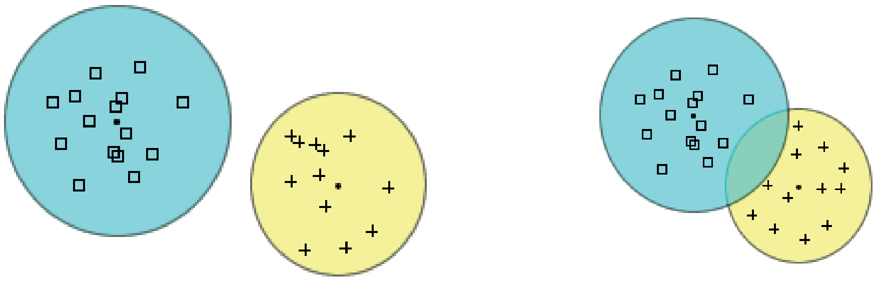


Fig. 9. To the left two homogeneous clusters without intersection. To the right two homogeneous clusters with intersection and without patterns in this region.

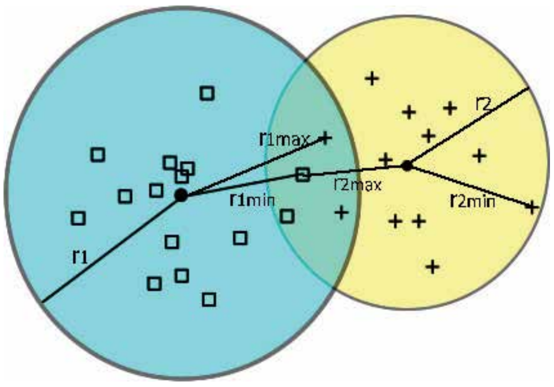


Fig. 10. Two homogeneus clusters with intersection and patterns inside this region.

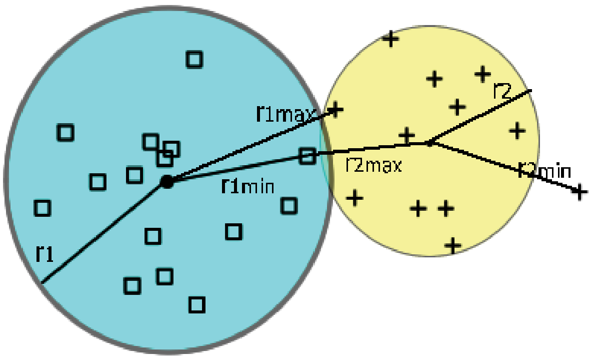


Fig. 11. Reduction of radios is applied to the left cluster and the right cluster is disregarded because it is not possible to do it.

3.2 On-line learning methodology

By means of this technique an adaptive model is obtained, that gradually accommodate its structure to the changes that provide the actual enviroment where this adaptive model develops. That is to say, whenever the algorithm makes an erroneous classification, it will have the opportunity of re-training it and learning on line with the appropriate type provided by the expert. In this way, the algorithm will be learning out more and more from the experiences to improve along its life time. In Fig. 12 to Fig. 15 the used procedure is shown.

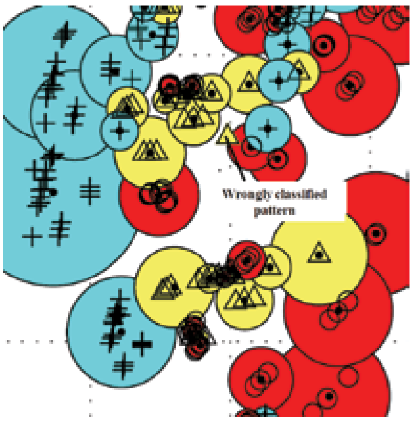


Fig. 12. Erroneous Classification. (Calderón, 2007).

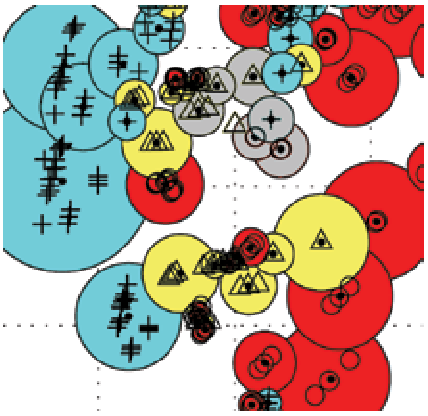


Fig. 13. Selection of the nearest neighboring K-clusters. (Calderón, 2007).

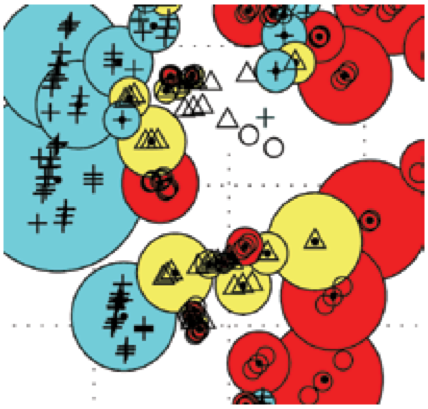


Fig. 14. New set of patterns for re-training. (Calderón, 2007).

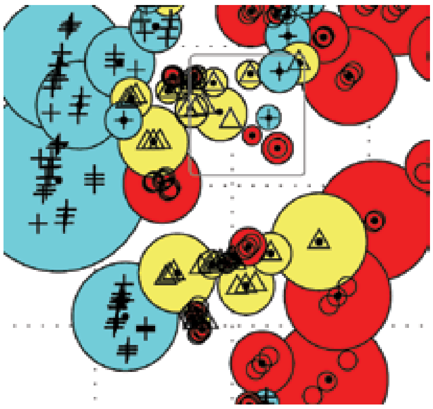


Fig. 15. Resulting set of clusters after on-line re-training. (Calderón, 2007).

Initially, the patterns for the re-training are prepared, extracting the cluster that made the erroneous classification (see Fig. 12) and the nearest neighboring k clusters (see Fig. 13), to form a reduced set of patterns for re-training (Fig. 14). With this set of patterns, the phases of non-supervised learning and supervised learning are implemented again illustrated in Fig. 6. This time the re-training is very efficient now that from the beginning, a subgroup of reduced clusters it is taking into account, all of them homogenous (the time of re-training takes a few seconds).

4. Design of faults classifiers based on neural networks

4.1 Generation of training and validation data

With the purpose of obtaining training samples of the signals of currents of phase and of zero-sequence the tool of simulation ATP was used (Alternative Transient Program) which has been validated at world-wide level as one of the most adapted to analyze electrical power systems (Electric Power Research Institute, 1989), (CanAm EMTP User Group, 1992).
In Fig. 16 the electrical system used for systematic exploration of the considered cases is illustrated.
With the purpose of automatically generating the data file with the ATP cases of variability of conditions of the SEP was developed a module in MATLAB that constructs the ATP format for the sensitivity analysis. Then, this file is run by means of the ATP program to generate the samples of Training, Validation and Checking of the studied models. In Fig. 17 all the flow of information from simulations with ATP and MATLAB until the model of neural network is schematically shown.
Initially, by means of the interface MATLAB-ATP, 508 patterns for training and 246 patterns for validation and checking were simulated. These cases of validation and checking were simulated as intermediate conditions of the training patterns with the purpose of verifying that over-training (validation stage) and the capacity of generalization of the model (checking stage) do not happen. Sensitivity was made on several parameters such as the impedances of source, chargeability of the transmission line, location of the fault, impedance of fault, and the type of fault: mono-phase (A, B, C), two-phase isolated (AB, BC and CA), two-phase to earth (AB-g, BC-g, CA-g) and three-phase (ABC).

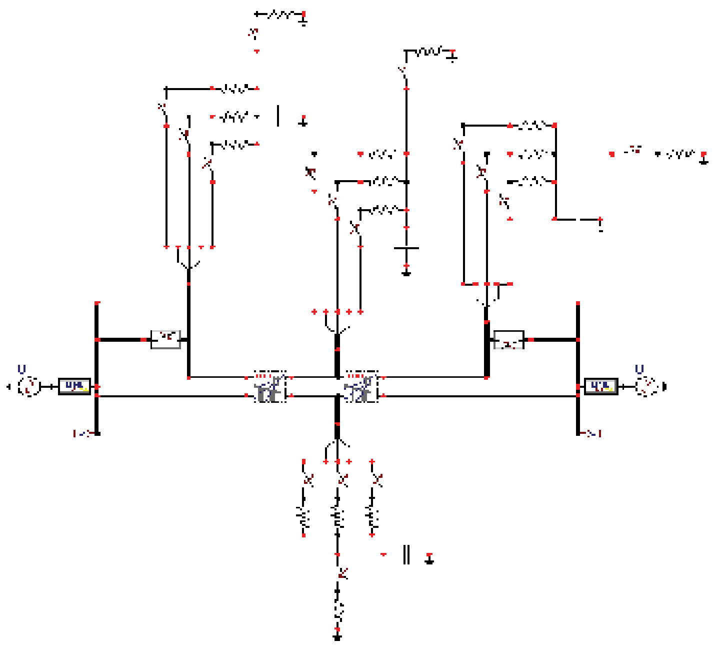


Fig. 16. Typical electrical circuit for analyzing conditions of faults in a SEP. (Alternative Transient Program-ATP).

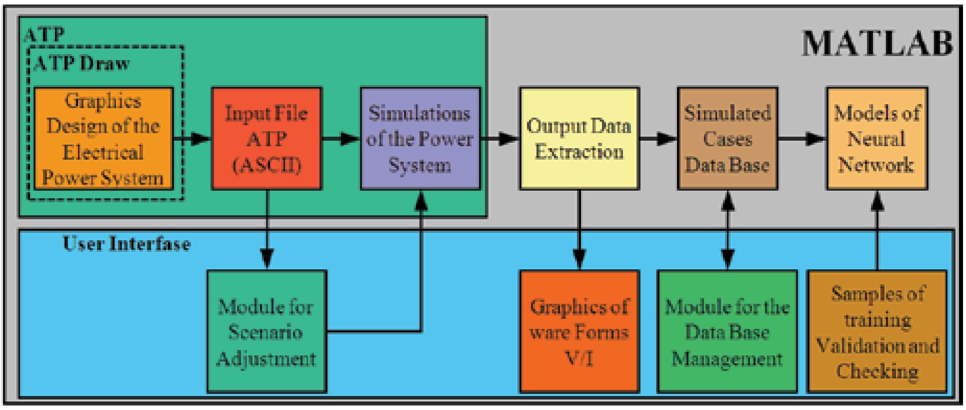

ATP
ATP Draw
Graphics Design of the Electrical Power System
Input File ATP (ASCII)
Simulations of the Power System
MATLAB
Output Data Extraction
Simulated Cases Data Base
Models of Neural Network
User Interfase
Module for Scenario Adjustment
Graphics of ware Forms V/I
Module for the Data Base Management
Samples of training Validation and Checking

Fig. 17. Integrated software tools for the simulation of electrical power system by means of ATP and MATLAB.

After that, this interface was used to generate 46996 simulated ATP cases of which 36500 were used for training, 5248 for validation, and 5248 for checking. Such the BR model as the ART 2 improved model were verified with these cases.

4.2 Inputs and outputs of the neural networks

The application of a patterns classifier requires first of all the selection of characteristics that contain information necessary to distinguish between the classes, it must be insensitive to the input variability, it must be limited in number to allow efficient calculation, and to limit the amount of required data of training.

As signals of input to the neural network can be selected different parameters from the system. The outputs of the neural network must indicate the type of fault. In general, two types of definition of outputs can be adopted. The first format is that the outputs are compound of A, B, C and G, which indicates that the fault was in the phase a, b or c and there is a connection with earth (G).

A	B	C	G	
0	0	0	0 -	normal condition
1	0	0	1 -	phase to earth fault
1	1	0	0 -	phases a and b without earth fault
1	1	1	0 -	phases a, b and c without earth fault.

The second type has 11 outputs, the first that represents the normal condition and each one of the remain ten is responsible for a type of fault, for example:

1000000000	normal condition.
0100000000	fault phase A.

4.3 Comparison of performance of the classifiers

4.3.1 Size of the neural network

The number of inputs such as the first four neural networks of study as to BR and ART 2 improved model were chosen equal to 7 consisting of three RMS (Root Mean Square) voltages, the three RMS phase currents and RMS zero sequence current. For the number of outputs BP, BR, ART 2 improved and RBF the first type of output was used, for the LVQ the second type and for network MF the output chose like a bi-dimensional Kohonen matrix of dimension 8x8.

As it is well-known, the selection of an optimal number of hidden layers and nodes for a continue BP network being point of research although numerous articles in these areas have been published. For the present study it was only used a hidden layer that turned out to be adapted for this individual application.

For the BP model a hidden layer was used that was to be adapted for this individual application. The number of nodes analyzed was considered from 10 to 16. Finally, with a selection of 12 neurons for the hidden layer a good performance was obtained. This size was used to BR model getting excellent performance too. The size of the matrix for the Kohonen model depends to a great extent on the kind of problem and the availability of training vectors. In this study, a matrix of 8x8 was selected after running a series of simulations and comparing the obtained results.

In order to determine the optimal structure of a RBF, a set of RBF models were trained and validated. In these simulations was carried out an analysis of sensitivity of the number of Kernel nodes varying from 300 to 508 based on the global performance of the network and it was found that with 357 neurons in the hidden layer a suitable performance is obtained.

In the adjustment of LVQ structure, the critical part is the selection of the number of neurons of the layer of Kohonen (competitive layer). In this analysis the total number of training vectors is to be kept in mind and select the number of neurons of the Kohonen layer like a multiple of the number of output nodes. In this study the number of nodes of Kohonen was selected based on the total of training vectors and the number of the eleven outputs. After several simulations were carried out it was found that an optimal number of neurons for the hidden layer of Kohonen for this application are 150.

Finally, the structure of ART 2 model is so different than the previous models and in this case the final number of clusters is automatically determined according to algorithm and is dependent of the threshold assigned by the user.

4.3.2 Learning process

In the study of BP network and BR model were used the functions of MATLAB tansig transference for the neurons of the hidden layer and the function purelin for the output linear layer. The learning factor that controls the rate of convergence and stability was chosen equal to 0.05. Initially, all the weights and bias were put in small random values. The input values were presented to the network and the output variables were specified. The training process took place until the value of error RMS (Root Mean Square) between the real output and the wished output reached an inferior acceptable value of 0.1. For the Kohonen layer was chosen equal to 0,9 for the phase of initial ordering and 0,02 for the final phase of tuning. 1000 steps were simulated in the ordering phase and it was considered a neighboring radius of 1 for tuning phase. A grille form was used for the space distribution of the neurons in the matrix of two dimensions.
The parameters of the units of RBF network were determined by means of the function newrb of MATLAB. First a subtractive grouping of the data by means of the function subclust of MATLAB with the purpose of estimating an average spread that complied most of the considered data was carried out. From this analysis was obtained a spread value of 0.19. Later, the centers of the radial units were determined by means of an algorithm of adaptive group that uses the function dist of MATLAB combined with a parameter of bias=0.833/spread. Once the centers are determined the algorithm newrb of MATLAB makes an iterative procedure of addition of new neurons until obtaining an error adapted between the real outputs of the model and targets of training assigned.
In LVQ network the function newlvq of MATLAB was used. The algorithm newlvq constructs an LVQ neural network like the one presented in Fig. 4. It is used as input criterion to consider the percentage of samples that each class has. For example, in this study 10 classes corresponding to 10 conditions of fault were considered. For the 508 samples each fault condition has associated 50 samples (0,1 p.u). All together, the sum must be equal to 1 p.u (100% of the 508 samples). For the learning process the function learnlv1 of MATLAB was used considering a learning rate of 0.01.
The ART 2 model training was explained in detail above and is depicted in Fig.6.

4.3.3 Training and validation error

The error often is used like a criterion to finish the learning process. It has been possible to find that, for a given set of training data and structures of network, the error of minimum learning that can be reached is similar for all the networks. Nevertheless, the time to reach the value of the error (speed of learning) is entirely different (Song et al., 1997). It is important to notice that obtaining the smaller error during the learning does not necessarily imply the best performance of the network. That is, there must be commitment between learning error and error during the validation phase.

4.3.4 Precision of classification

Initially, the BP, FM, RBF and LVQ neural networks trained were validated with 246 cases generated by means of ATP program under several conditions of the system and intermediate conditions of fault to the 508 cases considered in the training.

From the results obtained from the research for networks analyzed it was possible to observe that the rates of error vary with respect to the type of fault. As it was expected, the classification error is greater for the faults phase-phase without earth, which is the type of fault more difficult to detect.
The network that had the smaller error of classification was the RBF (only a 7% of the validation cases did not have suitable classification). In Fig. 18 the results of the training of RBF network are shown, where it can be noticed that the error between what is wished and what is real is very low.

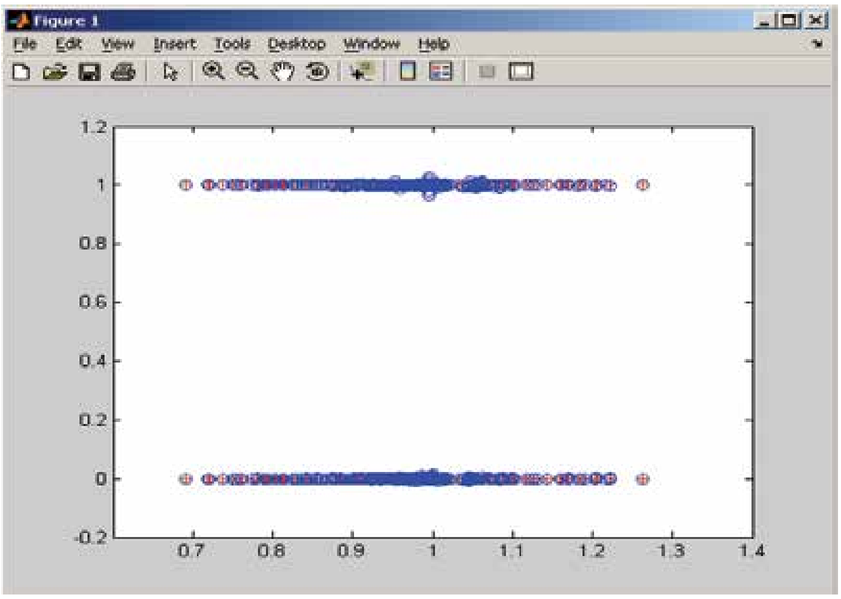


Fig. 18. Vector target (o) vs. Network output (+) during the training of RBF network. (Matlab educational license).

Although the previous models are, in general, good for classification purposes, these had some difficulties when certain conditions of the electrical system were considered (for example, high impedances faults). In some of these cases, the classification error was not suitable.
Due to this, based in the previous results, the research was oriented in the search of a hybrid model that was able to adapt itself to many expected conditions from the electrical power system and at the same time had a low classifcation error, and high level of generalization. The BR and ART 2 improved models were developed and then trained and validated using the methology described and ilustrated in Fig. 6.
By using the BR model the training error was 0% for the 36500 cases considered, 0.74% for the 5248 validation cases and 1,39% for the 5248 checking cases.
By using the ART 2 model the training error was 0.1% for the 36500 cases considered and 3.7% for the 10496 validation and checking cases.

4.3.5 Robustness

For many reasons, it is not possible to assume that the cases presented to the classifier during the phase of application are complete and precisely represented by the training set. It is particularly certain in the classifiers of fault of transmission lines. The patterns of training are normally limited, and in most cases are generated by means of simulation in computer that does not exactly match the real data of field. In general the data that enter the algorithms will be affected by the transducers and the noise from the atmosphere. Also, the parameters of the power system and the conditions change continuously. Thus, then actually a good robustness of the trained classifier is required. It includes deviations in the measurements and superposed white noise. The rates of undesired classification are considerably greater in BP network for the different cases considered, whereas the error rates for FM, RBF, LVQ and ART 2 improved neural networks increased moderately or very little. This is due to the purely supervised nature of BP network. The surfaces of decision of BP networks can take non intuitive forms because the space regions that are not occupied by the training data are classified arbitrarily. One way to improve this problem could be to combine BP with BR method (Bayesian regularization) in order to reduce the classification errors. Instead, the other networks analyzed are governed by non-supervised learning in which the regions of the input space occupied by the training data are not classified according to the proximity that commonly exists among the training data.

In summary, it is important to underline that a classifier has to be evaluated by its time of training, error rate, calculation, adaptation, and its real time implementation requirements. At the time of making a decision related to the selection of a network in particular it must be taken into account the combination of all these aspects and the possibility of considering new changes in the algorithms that allow improvement of the performance for the specific applications that are considered.

5. Conclusions and future work

The design of power systems protections can be essentially treated like a problem of pattern classification/recognition. The neural networks can be used like an attractive alternative for the development of new protection relays as much as the complexity of the electrical power systems grows. Different strategies of learning have to be explored before adopting a particular structure to a specific application, and establishing a commitment between the off-line training and the real time implementation.

In general, the combined non-supervised/supervised learning techniques offers better performance than the purely supervised training. In the present study it was possible to verify that FM, RBF and LVQ networks have a greater speed of training, similar error rate, better robustness to consider variations of both the system and the environment, and require much less amount of training data compared with BP network (Song et al., 1997). . On the other hand, the BP network is more compact and it is hoped to be faster when it is placed in operation under the real time performance.

This study, additionally showed, that in spite of those models have good performance to classify faults in electrical power systems in some special cases (for example, high impedances faults) the resultant error is not suitable. In order to take this fact into account, it is necessary to consider BP with BR or ART 2 improved models which resolve this kind of conflict.

It is important noticing that the present study focused in the performance of different models of neural networks applied to the classification of faults in electrical power systems.

Nevertheless, for the effects of being considered as protection alternatives of electrical power systems the techniques presented have to be integrally evaluated, considering in addition several practical issues. For example, it has to be combined with real field tests and the implementation of corresponding hardware.

6. Acknowledements

This work presents the results of the researches carried out by the National University of Colombia and the company Interconexión Eléctrica S.A. E.S.P (ISA), as partial advance of the research project co-financed by COLCIENCIAS entitled: "Computer Science Tools for the Automatic Diagnosis of Events in Transmission Lines of Electrical Power".

7. References

Aggarwal, R. and Song, Y. (1997). Artificial neural networks in power systems: Part I - General introduction into neural computing, *Power Engineering Jour.*, vol.112 11, no. 3, pp. June 1997,129–134.

Aggarwal, R. and Song, Y. (1998a). Artificial neural networks in power systems: Part II - Types of artificial neural networks, *Power Engineering Jour.*, vol. 12, no.1, Feb. 1998, pp. 41–47.

Aggarwal, R. and Song, Y. (1998b). Artificial neural networks in power systems: Part III - Examples of applications in power systems, *Power Engineering Jour.*, vol. 12, no. 6, Dec. 1998, pp. 279–287.

Badrul, H. ; Choudhury and Kunyu, Wang. (1996). Fault Classification Using Kohonen Feauture Mapping, *IEEE, Electrical Engineering Department*, University of Wyoming.

Calderón, J. A. (2007). Artificial Intelligence Adaptive Model for the Automatic Faults Diagnosis using oscillate-graphics records. *Thesis - Magister on Computer Science*, Computer Science Department, National University of Colombia – Campus Medellin.

Calderón, J. A. Ovalle ,D and Zapata, G. (2007). Comparative Analysis between Models of Neuronal Networks for the classification of faults in Electrical Systems. *IEEE CERMA México.*

CanAm EMTP User Group. (1992). Alternative Transients Program (ATP) - *Rule Book*, Portland, OR, Sep. 1992.

Dalstein, T. and Kulicke, B. (1995). Neural network approach to fault type classification for high speed protective relaying, IEEE Trans. Power Delivery, vol. 10, no. 2, Apr. 1995, pp. 1002–1011.

Dalstein, T.; Friedrich, T. ; Kulicke, B. and Sobajic, D. (1996). Multi neural network based fault area estimation for high speed protective relaying, *IEEE Trans. Power Delivery*, vol. 11, no. 2, Apr. 1996, pp. 740–747.

Dillon, T.S. & Niebur, D. (1996). Artificial Neural networks Applications in Power Systems. CRN Press.

Dillon. T. and Niebur. D. (1999). *Tutorial on artificial neural networks for power systems*. Engineering intelligent Systems for Electrical Engineering and Communications, v 7, n 1, pp. 3-17.

Electric Power Research Institute. (1989). Electromagnetic Transient Program (EMTP) - Rule Book, EPRI EL 6421-1, CA, June 1989, Palo Alto.

El-Sharkawi, M.A. & Niebur, D. (1996). Application of artificial neural networks to power systems. IEEE Tutorial Course Text 96TP112-0, Piscataway, NJ.

Fernandez, A. L. O. and Ghonaim, N. K. I. (2002). A novel approach using a FIRANN for fault detection and direction estimation for high-voltage transmission lines, *IEEE Trans. Power Delivery*, vol. 17, no. 4, Oct. 2002, pp. 894–900.

Foresee, F.D. and Hagan, M.T. (1997). Gauss-Newton approximation to Bayesian regularization, *Proceedings of the 1997 International Joint Conference on Neural Networks*, pp. 1930-1935.

Hagan. M.T. ; Demuth. H.B.; De Jesus. O. (2002). An introduction to the use of neural networks in control systems. International Journal of Robust and Nonlinear Control, v 12, n 11. pp. 959-85

Keerthipala, W.W. L. ; Wai, C. T. and Huisheng, W. (1997). Neural network based classifier for power system protection, *Electric Power Systems Research*, vol. 42, no. 2, Aug. 1997, pp. 109–114.

Kezunovic, M. (1997). A survey of neural net applications to protective relaying and fault analysis, *Engineering Intelligent Systems for El. Engr. and Comm.*, vol. 5, no. 4, Dec. 1997, pp. 185–192.

MacKav. D.J.C.; Takeuchi. R. (1998). (Cavendish Lab., Cambridge Univ., UK). *Statistics and Computing*, v 8, n 1, .pp. 15-23.

Oleskovicz, M.; Coury, D.V.; Aggarwal, R.K. (2001). A complete scheme for fault detection, classification and locationin transmission lines using neural networks. Developments in Power System Protection, 2001, Seventh International Conference on (IEE). vol , Issue , pp.335 – 338.

Ranaweera, D. K. (1994). Comparison of neural network models for fault diagnosis of power systems, Electric Power Systems Research, vol. 29, no. 2, Mar. 1994, pp. 99–104.

Sidhu. T.S. and Mitai. L. (2000). Rule extraction from an artificial neural network based fault direction discriminator. *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings*. Navigating to a New Era (Cat. No.00TH8492), pp. 692-6 vol.2

Song, Y. H. ; Xuan, Q. X. ; Johns, A. T. (1997). Comparison studies of five neural network based fault classifiers for complex transmission lines. *Electric Power Systems Research*, vol. 43, no. 2, (Nov. 1997), pp. 125–132.

Song, Y. H. ; Xuan, Q. X. and Johns, A. T. (1997). Comparison studies of five neural network based fault classifiers for complex transmission lines, *Electric Power Systems Research*, vol. 43, no. 2, Nov. 1997, pp. 125–132.

Song, Y.H., Johns, A.T. and Aggarwal, R.K. (1996). Applications in Power Systems, Science Press and Kluwer Academic, *Computational Intelligence*.

Vasilic, S. (2004). Fuzzy neural network pattern recognition algorithm for classification of the events in power system networks. *Doctoral dissertation*, Texas A&M University. Available electronically from http : / /handle .tamu .edu /1969 .1 /436.

Zahra, F. ; Jeyasurya, B. and Quaicoe, J. E. (2000). High-speed transmission line relaying using artificial neural networks, *Electric Power Systems Research*, vol. 53, no. 3, Mar. 2000, pp. 173–179.

Genetic Network Programming with Reinforcement Learning and Its Application to Creating Stock Trading Rules

Yan Chen, Shingo Mabu and Kotaro Hirasawa
Waseda University
Japan

1. Introduction

Evolutionary Computation is well-known for producing the solutions in optimization problems based on change, composition and selection. We have proposed Genetic Network Programming (GNP) [1, 2] as an extended method of Genetic Algorithm (GA) [3, 4] and Genetic Programming (GP) [5, 6]. It has been clarified that GNP is an effective method mainly for dynamic problems since GNP represents its solutions using graph structures, which contributes to creating quite compact programs and implicitly memorizing past action sequences in the network flows. Moreover, we proposed an extended algorithm of GNP which combines evolution and reinforcement learning [7] (GNP-RL). GNP-RL has two advantages, and one of them is online learning. Since original GNP is based on evolution only, the programs are evolved mainly after task execution or enough trial, i.e., offline learning. On the other hand, the programs in GNP-RL can be changed incrementally based on rewards obtained during task execution, i.e., online learning. Concretely speaking, when an agent takes a good action with a positive reward at a certain state, the action is reinforced and when visiting the state again, the same action will be adopted with higher probability. Another advantage of GNP-RL is the combination of a diversified search of GNP and an intensified search of RL. The role of evolution is to make rough structures through selection, crossover and mutation, while the role of RL is to determine one appropriate path in a structure made by evolution. Diversified search of evolution could change programs largely with which the programs could escape from local minima. RL is executed based on immediate rewards obtained after taking actions, therefore intensified search can be executed efficiently.
Research on stock price prediction and trading model using evolutionary computation and neural networks has been done [8–10] in recent years. Generally speaking, there are two kinds of methods for predicting stock prices and determining the timing of buying or selling stocks: one is fundamental analysis which analyzes stock prices using the financial statement of each company, the economic trend and movements of the exchange rate; the other is technical analysis which analyzes numerically the past movement of stock prices. The proposed method belongs to technical analysis since it determines the timing of buying and selling stocks based on the technical indices such as Relative Strength Index, MACD, Golden/Dead Cross and so on.

There are three important points in this paper. First, we combine GNP and Sarsa Learning [11] which is one of the reinforcement learning methods, while Importance Index (IMX) and Candlestick Charts [12–15] are introduced for efficient stock trading decision making. Concretely speaking, Sarsa is used to select appropriate actions (buying/selling), stock price information obtained from IMX and candlestick charts through the experiences during the trading. IMX and candlestick charts tell GNP whether or not the buying or selling signals are likely to appear at the current day. Second, although there are so many technical indices in the technical analysis, GNP with Sarsa can select appropriate indices and also select candlestick charts to judge the buying and selling timing of stocks. In other words, GNP with Sarsa could optimize the combinations of the information obtained by technical indices and candlestick charts. The third important point is that sub-nodes are introduced in each node to determine appropriate actions (buying/selling) and to select appropriate stock price information depending on the situation.

This paper is organized as follows: In Section 2, the related works are described. In Section 3, the algorithm of the proposed method is described. Section 4 shows simulation environments, conditions and results. Section 5 is devoted to conclusions.

2. Related works

Prediction in financial domains, especially in stock market is quite difficult for a number of reasons. First, the ultimate goal of our research is not to minimize the prediction error, but to maximize the profits. It forces us to consider a large number of independent variables, thereby increasing the dimensionality of the search space. Second, the weak relationships among variables tend to be nonlinear, and may hold only in limited areas of the search space. Especially, the data in stock markets are highly time-variant and changing every minute. Third, the stock market data are given in an event-driven way. They are highly influenced by the indeterminate dealing. In financial practice, the key is to find the hidden interactions among variables [16].

Stock market analysis has been one of the most actively pursued avenues of Machine Learning (ML) research and applications. The most recent literature in the related fields exposed Portfolio Optimization, Investment Strategy Determination, and Market Risk Analysis as three major trends in the utilization of Machine Learning approaches. Portfolio Optimization focuses on the correlative properties of stock market data in order to extract mutual dependency (or independency) information [17–19]. Investment Strategy Determination addresses financial prediction based on financial index analysis for the purposes of investment decision-making. Various Neural Network approaches are by far the most commonly taken route in the related works. However, other alternative methods exist, such as Support Vector Machines [20], Genetic Algorithms [21] and statistical analysis [22]. The Market Risk Analysis concentrates on the evaluation of the risk factors involved in various investment options, such as expected return and volatility. An example of an overall market risk evaluation system is described in [23]. Our research focuses on the problem of Investment Strategy Determination through the use of GNP with reinforcement learning technique.

In recent years, evolutionary algorithms have been applied to several financial problems.

There have been several applications of Genetic Algorithms (GA) to the financial problems, such as portfolio optimization, bankruptcy prediction, financial forecasting, fraud detection and scheduling [24]. Genetic Programming (GP) has also been applied to many problems in the time-series prediction.

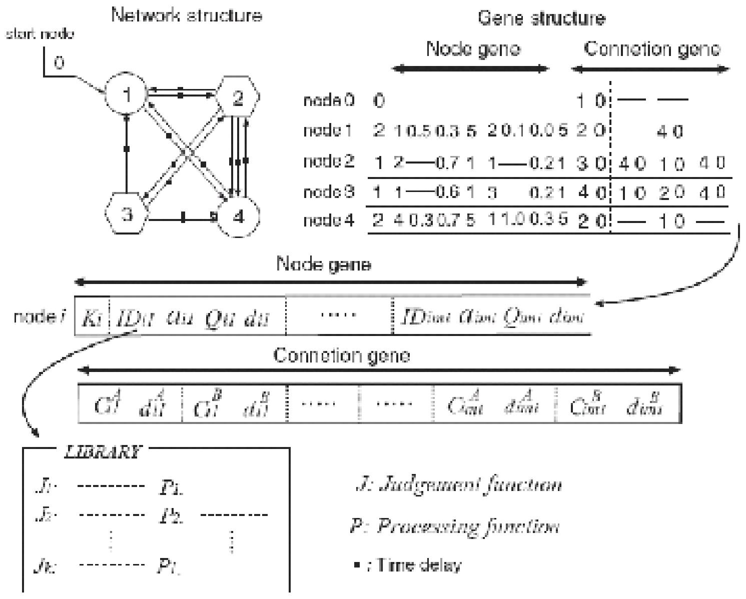


Fig. 1. Basic structure of GNP with Sarsa

In our research, we propose Genetic Network Programming with Sarsa Learning for creating trading rules on stock markets. GNP has the following advantages in the financial prediction field. First, GNP has a memory function because of its graph structure, i.e., judgment nodes and processing node are connected to each other in a network. As stock markets are highly influenced by the time, we can consider the information in the past well by the memory function of GNP for creating the effective programs. Second, GNP works extremely well for dealing with the stock market problems. That is because GNP has quite compact structure and it can reuse the nodes for many times. By using GNP we can create effective trading rules in the stock market, and we can also save the calculation time and memory consumption because of the compact structures of GNP. By combining GNP with Sarsa Learning in this paper, we get more advantages such as the combination of online learning and offline learning, diversified search and intensified search.

3. GNP with Sarsa (GNP-Sarsa) and its trading algorithm

3.1 Basic structure of GNP-Sarsa

Figure 1 shows a basic structure of GNP-Sarsa an**d Fig. 2** shows judgment node and processing node structures. GNP-Sarsa consists of judgment nodes and processing nodes, which are connected to each other. Judgment nodes have if-then type branch decision functions. They return judgment results for assigned inputs and determine the next node.
Processing nodes take actions (buying or selling stocks). While judgment nodes have conditional branches, processing nodes have no conditional branches. The role of a start node is to determine the first node to be executed. The graph structure of GNP has some inherent characteristics such as compact structures and an implicit memory function that contributes to creating effective action rules as described in section 2. GNP-Sarsa has two kinds of time delays: time delays GNP-Sarsa spend on judgment or processing, and the ones

it spends on node transitions. In this paper, the role of time delays is to determine the maximum number of technical indices and candlestick information to be considered when GNP-Sarsa determines buying or selling at a certain day.

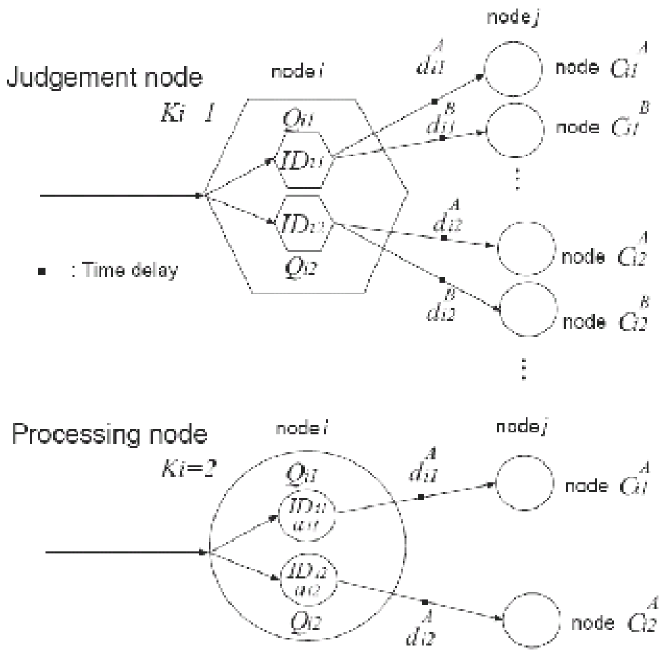


Fig. 2. Node Structure

In the table of node gene, K_i represents the node type, K_i = 0 means start node, K_i = 1 means judgment node and K_i = 2 means processing node. ID_i represents an identification number of the node function, e.g., K_i = 1 and ID_i = 2 mean the node is J_2. a_{ip} is a parameter which represents the threshold for determining buying or selling stocks in a processing node. Q_{ip} means Q value which is assigned to each state and action pair. In this method, "state" means a current node, and "action" means a selection of a sub-node (node function). In general reinforcement learning framework, the current state is determined by the combination of the current information, and action is an actual action an agent takes, e.g., buying or selling stocks. However, in GNP-Sarsa, the current node is defined as the current state, and a selection of a sub-node is defined as an action. d_{ip} ($1 \leq p \leq m_i$, m_i is the number of subnodes in judgment and processing nodes) is the time delay spent on the judgment or processing at node i, while d_{ip}^A, d_{ip}^B, … are time delays spent on the node transition from node i to the next node. In this paper, d_{ip}^A, d_{ip}^B, … are set at zero time unit, d_{ip} of each judgment node is set at one time unit, d_{ip} of each processing node is set at five time units. We suppose that the trade in one day ends when GNP uses five or more time units, which means the trade in one day ends when GNP executes fewer than five judgment nodes and one processing node, or five judgment nodes. C_{ip}^A, C_{ip}^B, … show the node number of the next node. Judgment node determines the upper suffix of the connection genes to refer to depending on the judgment result. If the judgment result is "B," GNP-Sarsa refers to C_{ip}^B and d_{ip}^B. Processing nodes always refer to C_{ip}^A and d_{ip}^A because processing nodes have no conditional branch.

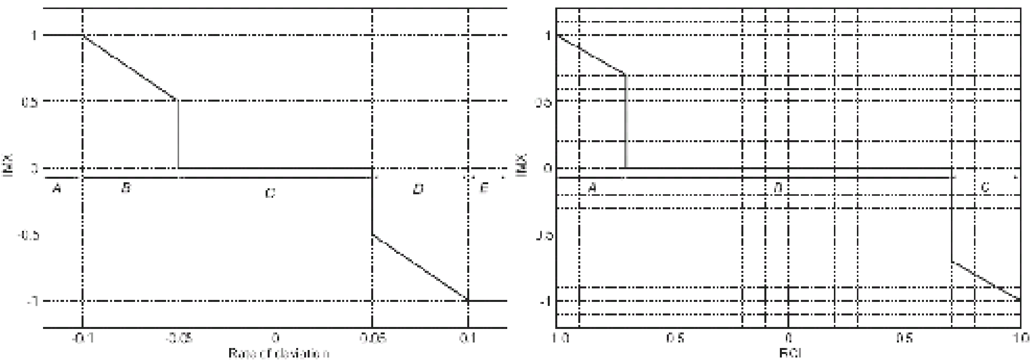


Fig. 3. IMX functions in judgment nodes (in case of ROD and RCI)

3.2 Judgment and processing functions of GNP-Sarsa

The node transition of GNP-Sarsa starts from a start node and continues depending on the node connections and judgment results. Fig. 2 shows node structures of a judgment node and a processing node.

(1) *Judgment node*: When a current node i is a judgment node, first, one Q value is selected from Q_{i1}, …, Q_{imi} based on ε-greedy policy. That is, a maximum Q value among Q_{i1}, …, Q_{imi} is selected with the probability of 1-ε , or a random one is selected with the probability of ε. Then corresponding function (ID_{ip}) is selected. The gene ID_{ip} shows a technical index or a candlestick GNP judges at node i. Each technical index has its own IMX function shown in Fig. 3. x axis shows the value of each technical index, and the sections A, B, C, ... correspond to judgment results. Suppose Q_{i1} and the corresponding ID_{i1} = 1 (judgment of rate of deviation) are selected, and if the rate is more than 0.1, the judgment result becomes E, and the next node number becomes C_{i1}^{E}. y axis shows the output of the IMX function and it is used at a processing node. However, the IMX output of golden cross, dead cross and MACD could be 1, 0 or -1 based on the cross of the lines, and the values correspond to judgment results A, B and C, respectively. Concretely speaking, for three days after a golden cross appears, the IMX output becomes 1, and for three days after a dead cross appears, it becomes -1, otherwise 0. Furthermore, for three days After MACD passes through the signal from the lower side to the upper side, the IMX output becomes 1, and for three days after it does from the upper to the lower, the IMX output becomes -1, otherwise it becomes 0. Generally, golden cross indicates buying signals and dead cross indicates selling signals, therefore, buying signals become stronger as the IMX output is close to 1, and selling signals become stronger as it is close to -1.

In this paper, candlestick chart is used as one of judgment functions. As we know, candlestick chart has been winning international recognition for its good indication of stock prices, and it has been widely used as the means of indicating the fluctuations of the stocks.

The proposed method has judgment nodes which check candlestick chart patterns. The judgment function of candlestick chart is executed as follows. When the selected sub-node has a judgment function of candlestick chart, GNP judges yesterday's candlestick and the candlestick of the day before yesterday. There are eight patterns of candlestick charts as shown in Fig. 4 according to two kinds of rules: (A) Judge whether there is a gap or not between yesterday's lowest price and the highest price of the day before yesterday, or between yesterday's highest price and the lowest price of the day before yesterday. (B) Judge whether or not yesterday's closing price is higher than the opening price of the day before yesterday. Especially, when the opening price equals to the closing price, the case is treated as black body candlestick. As an example, when the candlestick pattern is "3", GNP-Sarsa selects third branch to transfer to the next node. However, judgment nodes of candlestick chart do not have IMX function.

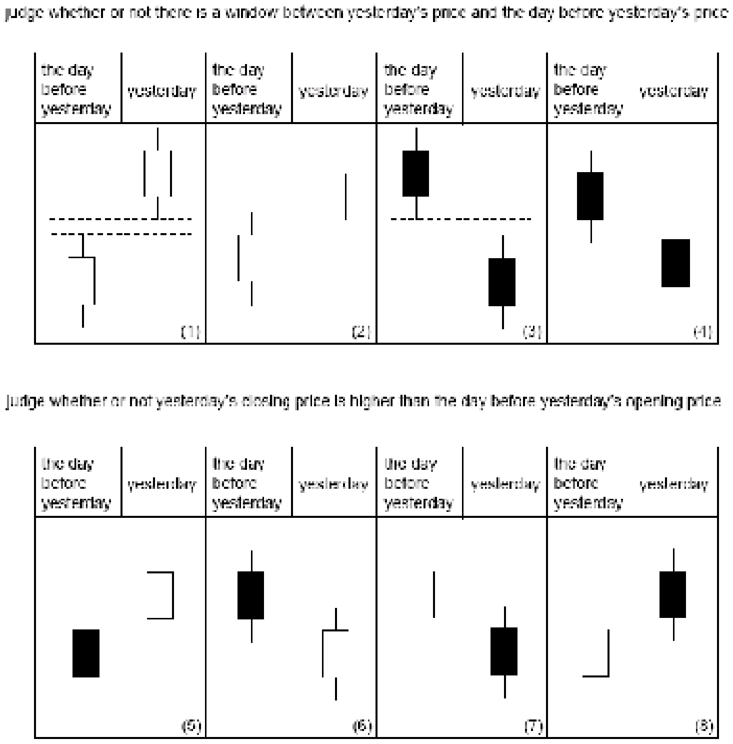

judge whether or not there is a window between yesterday's price and the day before yesterday's price
the day before yesterday
yesterday
(1)
(2)
(3)
(4)
judge whether or not yesterday's closing price is higher than the day before yesterday's opening price
(5)
(6)
(7)
(8)

Fig. 4. Candlestick chart patterns

(2) *Processing node*: When a current node is a processing node, Q_{ip}, the corresponding ID_{ip} and a_{ip} are selected based on ε -greedy policy. The selected a_{ip} is a threshold for determining buying or selling stocks. We explain the procedure of buying and selling stocks using Fig. 5, where the current node at time t is a processing node.

1. First, one Q value is selected from Q_{i1}, … Q_{imi} based on ε-greedy policy. That is, a maximum Q value among Q_{i1}, … Q_{imi} is selected with the probability of 1-ε , or a random one is selected with the probability of ε . Then the corresponding a_{ip} is selected.
2. Calculate an average of the IMXs obtained at the judgment nodes executed in the node transition from the previous processing node to the current processing node.

$$A_t = \frac{1}{|I'|}\sum_{i' \in I'} IMX(i')$$

where, I' shows a set of suffixes of the judgment node numbers executed in the node transition from the previous processing node to the current processing node. IMX(i')

shows an IMX output at node $i' \in I'$. However, when a judgment node of the candlestick chart was executed or an IMX output is zero at a judgment node of golden cross, dead cross and MACD, the node number is excluded from I' for calculating A_t .

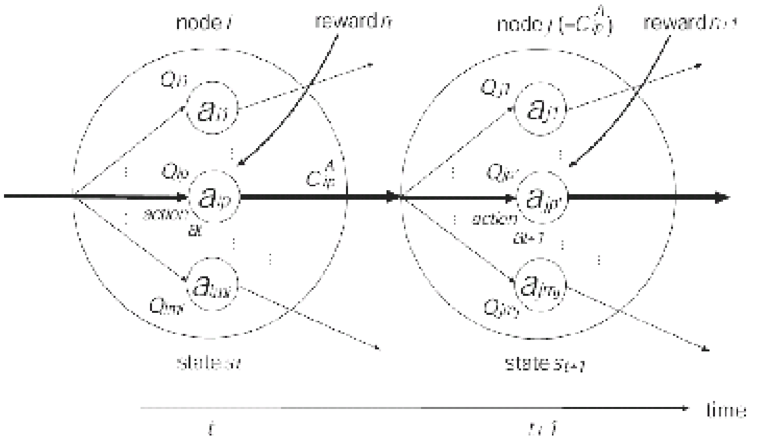


Fig. 5. An example of node transition

3. determine buying or selling:
   In the case of ID_{ip} = 0 (buy): if $A_t \geq a_{ip}$ and we do not have any stocks, GNP buys as much stocks as possible. Otherwise, GNP takes no action.
   In the case of ID_{ip} = 1 (sell): if $A_t < a_{ip}$ and we have stocks, GNP sells all the stocks. Otherwise, GNP takes no action.
4. The current node is transferred to the next node. If a_{ip} is selected, the next node number becomes C_{ip}^A.

The above procedure puts the information of the technical indices together into A_t, and GNP-Sarsa determines buying or selling stocks by comparing At with a_{ip}. Therefore, the points of this paper are 1) to find appropriate a_{ip} in the processing nodes by evolution and Sarsa, and 2) to determine I' by evolution, in other words, what kinds of judgments (technical indices and candlestick charts) should be considered is determined automatically.

3.3 Learning phase

First we explain Sarsa algorithm briefly. Sarsa can obtain Q values which estimate the sum of the discounted rewards obtained in the future. Suppose an agent selects an action a_t at state s_t at time t, a reward r_t is obtained and an action a_{t+1} is taken at the next state s_{t+1}. Then Q (s_t, a_t) is updated as follows.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

a is a step size parameter, and γ is a discount rate which determines the present value of future rewards: a reward received k time steps later is worth only γ^{k-1} times of the reward supposed to receive at the current step.

As described before, a state means the current node and an action means the selection of a sub-node. Here, we explain the procedure for updating Q value in this paper.

1. At time t, GNP refers to Q_{i1}, …, Q_{imi} and selects one of them based on ε -greedy. Suppose that GNP selects Q_{ip} and the corresponding function ID_{ip}.

2. GNP executes the function ID_{ip}, gets the reward r_t and suppose the next node j becomes C_{ip}^A.
3. At time t+1, GNP selects one Q value in the same way as step1. Suppose that Q_{jp}' is selected.
4. Q value is updated as follows.

$$Q_{ip} \leftarrow Q_{ip} + \alpha[r_t + \gamma Q_{jp'} - Q_{ip}]$$

5. $t \leftarrow t + 1, i \leftarrow j, p \leftarrow p'$ then return step 2.

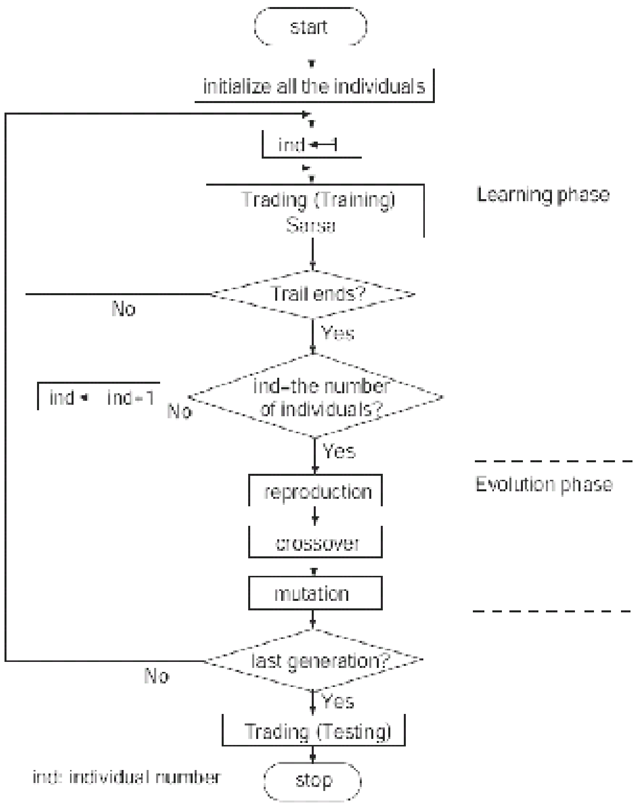


Fig. 6. Flowchart of GNP-Sarsa

3.4 Evolution phase
Figure 6 shows the whole flowchart of GNP-Sarsa. In this sub-section, the genetic operators in the evolution phase are introduced. The role of evolution is to change graph structures and randomly change node parameters a_{ip}.

3.4.1 Crossover
Crossover is executed between two parents and generates two offspring [Fig. 7]. The procedure of crossover is as follows.

1. Select two individuals using tournament selection twice and reproduce them as parents.

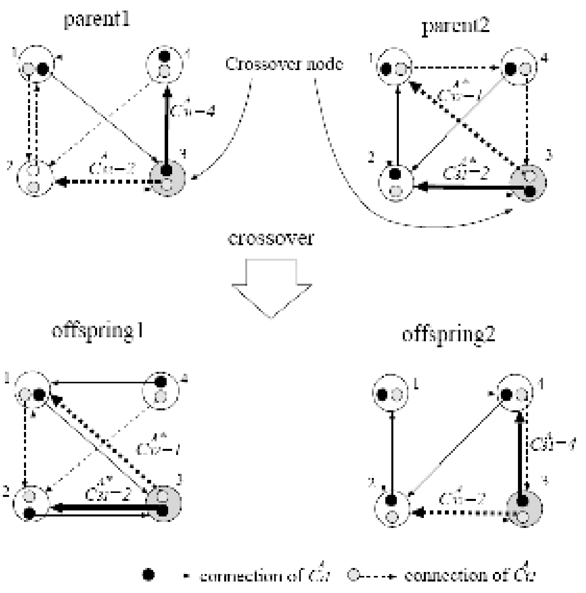


Fig. 7. Crossover

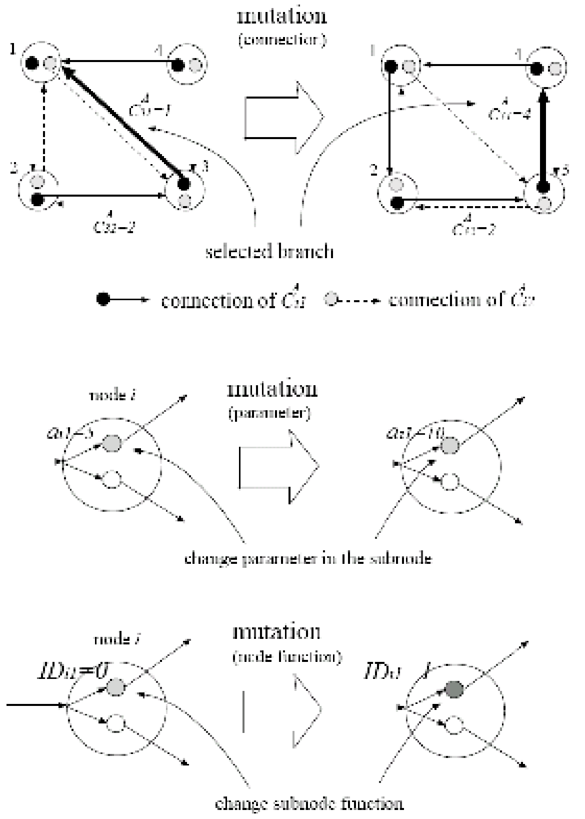


Fig. 8. Mutation

2. Each node is selected as a crossover node with the probability of P_c.
3. Two parents exchange the genes of the corresponding crossover nodes, i.e., the nodes with the same node number.
4. Generated new individuals become the new ones of the next generation.

Figure 7 shows a crossover example of the graph structure with three processing nodes for simplicity. If GNP exchanges the genes of judgment nodes, it must exchange all the genes with suffix *A*, *B*, *C*, … simultaneously.

3.4.2 Mutation

Mutation is executed in one individual and a new one is generated [Fig. 8]. The procedure of mutation is as follows.

1. Select one individual using tournament selection and reproduce it as a parent.
2. Mutation operation
   a. change connection: Each node branch (C_{ip}^A, C_{ip}^B, …) is selected with the probability of Pm, and the selected branch is reconnected to another node.
   b. change parameters (a_{ip}): Each a_{ip} is changed to other value with the probability of P_m.
   c. change node function: Each node function (ID_{ip}) is selected with the probability of P_m, and the selected function is changed to another one.
3. Generated new individual becomes the new one of the next generation.

4. Simulation

To confirm the effectiveness of GNP-Sarsa, we carried out the trading simulations using 16 brands selected from the companies listed in the first section of Tokyo stock market in Japan (see Table 3). The simulation period is divided into two periods; one is used for training and the other is used for testing simulation.

Training: January 4, 2001–December 30, 2003 (737 days)

Testing: January 5, 2004–December 30, 2004 (246 days)

We suppose that the initial funds is 5,000,000 Japanese yen in both periods, and the order of buying or selling is executed at the opening of the trading day, i.e., we can buy and sell stocks with the opening price.

4.1 Fitness and reward

Reward shows a capital gain of one trade (one set of buying and selling) and is used for learning. Fitness is the sum of the rewards obtained in the trading period.

Reward=selling price - purchase price

Fitness=Σ Reward

4.2 Conditions of GNP-Sarsa

GNP-Sarsa uses judgment nodes which judge the technical indices shown in Table 1 and candlestick charts. The technical indices are calculated using three kinds of calculation periods except Golden/Dead cross and MACD. Therefore, the number of kinds of judgment nodes is 21 (including one candlestick judgment). The number of processing functions is two: buying and selling. Table 2 shows simulation conditions. The total number of nodes in each individual is 31 including 20 judgment nodes, 10 processing nodes and one start node. However, the functions ID_{ip} in sub-nodes are determined randomly at the beginning of the first generation, and changed appropriately by evolution.

Technical index	period1	period2	period3
Rate of deviation	5	13	26
RSI	5	13	26
ROC	5	13	26
Volume ratio	5	13	26
RCI	9	18	27
Stochastics	12	20	30
Golden/Dead cross	5 (short term), 26 (long term)		
MACD	5 (short term), 26 (long term), 9 (signal)		

Table 1. Calculation periods of the technical indices [day]

Number of individuals = 300 (mutation: 179, crossover:120, elite:1)
Number of nodes = 31 (Judgment node:20, Processing node:10, start node:1) Number of sub-node in each node = 2
Pc=0.1, Pm=0.03, α=0.1, γ=0.4, ε=0.1

Table 2. Simulation conditions

The initial connections between nodes are also determined randomly at the first generation. At the end of each generation, 179 new individuals are produced by mutation, 120 new individuals are produced by crossover, and the best individual is preserved. The other parameters are the ones showing good results in the simulations. The initial *Q* values are set at zero.

4.3 Simulation results

First, 300 individuals are evolved for 300 generations using the training data. Fig. 9 shows the fitness curve of the best individual at each generation in the training term using the data of Toyota motor, and the line is the average over 30 independent simulations. From the Figure, we can see that GNP-Sarsa can obtain larger profits for the training data as the generation goes on. The fitness curves of the other companies have almost the same tendency as that of Toyota Motor.
Next, the test simulation is carried out using the best individual at the last generation in the training term. Table 3 shows the profits and losses in the testing term. The values in Table 3 are the average of the 30 independent simulations with different random seeds. For the comparison, the table also shows the results of Buy&Hold which is often considered to be a benchmark in trading stocks simulations. Buy&Hold buys as much stocks as possible at the opening of the market on the first day in the simulations, and sells all the stocks at the opening on the last day. From the table, the proposed method can obtain larger profits than Buy&Hold in the trade of 12 brands out of 16. By comparing with original GNP, the proposed method can get larger profits than traditional GNP in the trade of 13 brands out of 16. Especially, the stock prices of NEC, Fuji Heavy Ind., KDDI, Nomura Holdings, Shin-Etsu Chemical Co., Ltd. are down trend, so Buy&Hold always makes a loss, however the proposed method can obtain profits in five all brands.

	Profit[yen](profit rate[%])		
Brand	GNP-Sarsa	GNP	Buy&Hold
Toyota Motor	522,333(10.4)	480,500(9.6)	520,000 (10.4)
Mitsubishi Estate	444,733(8.9)	405,700(8.1)	664,000(13.3)
Showa Shell Sekiyu	263,100(5.3)	294,755(5.9)	319,200(6.4)
East Japan Railway	413,833(8.3)	491,500(9.8)	477,000(9.5)
NEC Corporation	36,600(0.7)	-126,150(-2.5)	-1,026,000(-20.5)
Fuji Heavy Ind.	217,133(4.3)	97,700(2.0)	-189,000(-3.8)
Sekisui House, Ltd.	582,466(11.6)	54,600(1.1)	264,000(5.3)
Mitsu & Co.	473,033(9.5)	118,450(2.4)	240,000(4.8)
Sony	148,733(3.0)	280,500(5.6)	150,000(3.0)
Tokyo Gas	669,733(13.4)	382,000(7.6)	372,000(7.4)
KDDI	199,400(4.0)	-76,600(-1.5)	-576,000(-11.5)
Tokyo Electric Power	570,266(11.4)	210,000(4.2)	262,500(5.3)
Daiwa House	612,633(12.3)	235,400(4.7)	32,000(0.6)
Nomura Holdings	366,033(7.3)	-293,785(5.9)	-985,500(-19.7)
Shin-Etsu Chemical	562,700(11.3)	7,250(0.1)	-264,000(-5.3)
Nippon Steel	469,866(9.4)	-27,350(0.5)	399,000(8.0)
Average	409,537(8.2)	158,404(3.2)	41,200(0.8)

Table 3. Profits in the test simulations

Figure 10 shows the change of the price of Toyota motor in the testing term and also shows typical buying and selling points by the proposed method. Fig. 11 shows the change of the funds as a result of the trading. From these figures, we can see that GNP-Sarsa can buy stocks at the lower points and sell at the higher points.

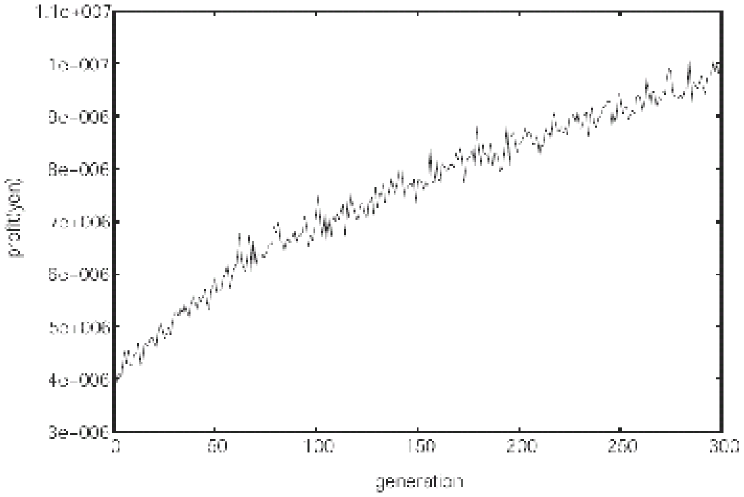


Fig. 9. Fitness curve in the training period (Toyota Motor)

Figure 12 shows the average ratio of the nodes used in the test period over 30 independent simulations in order to see which nodes are used and which are most efficient for stock trading model. The total number of node function is 23, while each processing node has a node number (0–1), and each judgment node has a node number (2–22). The x-axis shows the kinds of the nodes while the y-axis shows the average ratio of the used nodes. From the figure, we can see that the processing nodes are used to determine buying and selling stocks, and the judgment nodes of "Rate of deviation1" corresponding to period1 and "Volume ratio3" corresponding to period3 are frequently used.

Thus it can be said that GNP-Sarsa judges that these nodes are important to determine stock trading. GNP-Sarsa can automatically determine which nodes should be used in the current situation by evolving node functions and connections between nodes, in other words, GNP-Sarsa can optimize the combination of technical indices and candlestick charts used for stock trading model.

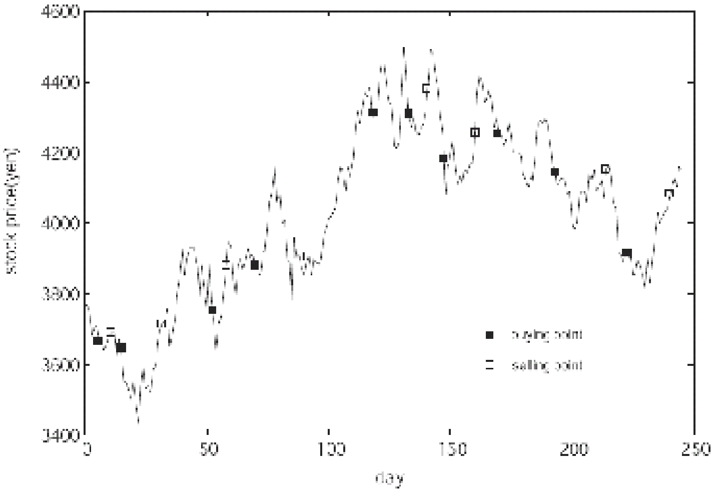


Fig. 10. Stock price of Toyota Motor and typical buying/selling points in 2004 (test period)

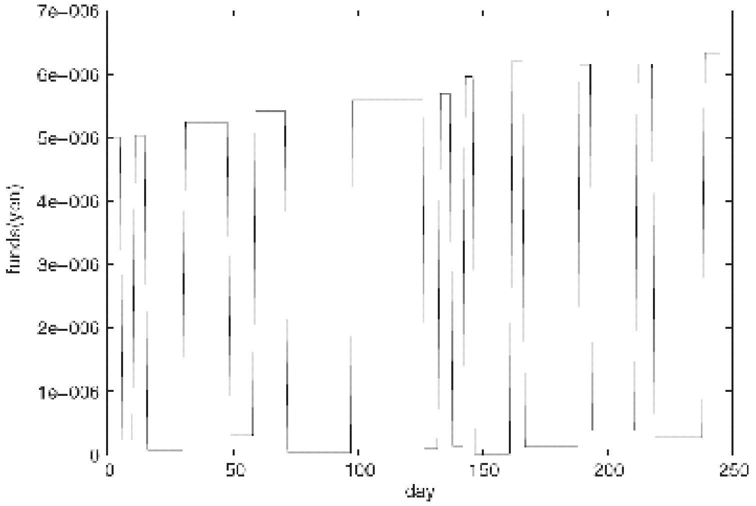


Fig. 11. Change of funds in the test simulation (Toyota Motor)

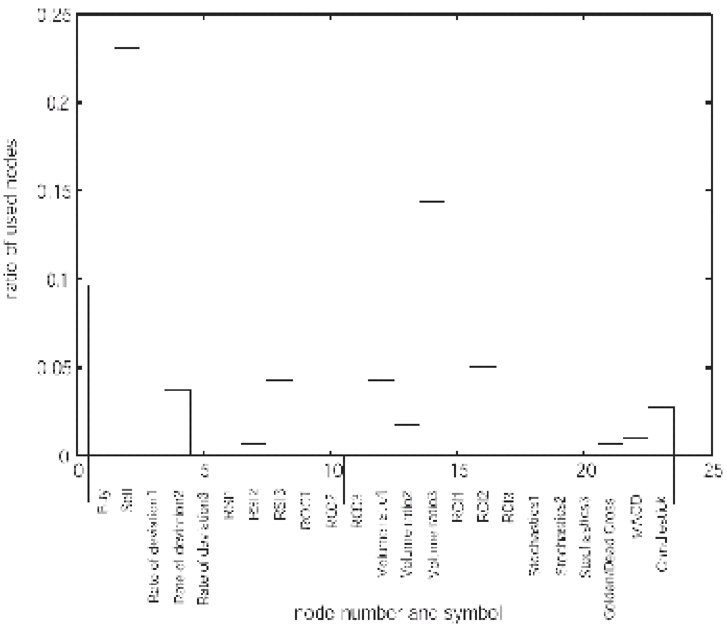


Fig. 12. Ratio of nodes used by GNP-Sarsa in the test period (Toyota Motor)

5. Conclusions

In this paper, a stock trading model using GNP-Sarsa with important index and candlestick charts is proposed. First, a newly defined IMX function is assigned to each technical index to tell GNP-Sarsa whether buying or selling stocks is recommended or not. Second, Sarsa learns Q values to select appropriate sub-nodes/functions used to judge the current stock price information and determine buying and selling timing. We carried out simulations using stock price data of 16 brands for four years. From the simulation results, it is clarified that the fitness becomes larger as the generation goes on and the profits obtained in the testing term are better than Buy&Hold in the simulations of 12 brands out of 16. By comparing with original GNP, the proposed method can get larger profits than traditional GNP in the trade of 13 brands out of 16. When there is downtrend, Buy&Hold makes a loss in five brands, but the proposed method can obtain profits in five all brands.

There remain some problems to be solved. First, in this paper, the calculation period of each technical index is fixed in advance. However, to improve the performance of the proposed method, we should develop a new method that can learn appropriate calculation periods. Next, it is necessary to consider the way of classifying the candlestick chart body type, and create more efficient judgment functions to judge current stock price appropriately. Also, we will evaluate the proposed method comparing with other methods using many data of other brands.

6. References

[1] Mabu, S., Hirasawa, K. & Hu, J. (2007), A graph-based evolutionary algorithm: Genetic network programming and its extension using reinforcement learning, *Evolutionary Computation*, MIT Press, Vol.15, No.3, pp. 369-398.

[2] Eguchi, T., Hirasawa, K., Hu, J. & Ota N. (2006), Study of evolutionary multiagent models based on symbiosis, *IEEE Trans. Syst., Man and Cybern. B*, Vol.36, No.1, pp. 179-193.

[3] Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press.

[4] Goldberg, D. E. (1989), *Genetic Algorithm in search, optimization and machine learning*, Addison-Wesley.

[5] Koza, J. R. (1992), *Genetic Programming, on the programming of computers by means of natural selection*, Cambridge, Mass., MIT Press.

[6] Koza, J. R. (1994), *Genetic Programming II, Automatic Discovery of Reusable Programs*, Cambridge, Mass., MIT Press.

[7] Sutton, R. S. & Barto, A. G. (1998), *Reinforcement Learning -An Introduction*, Cambridge, Massachusetts, London, England, MIT Press.

[8] Baba, N., Inoue, N. & Yanjun, Y. (2002), Utilization of soft computing techniques for constructing reliable decision support systems for dealing stocks, *Proceedings of Int. Joint Conf. on Neural Networks*.

[9] Potvin, J. -Y., Soriano, P. & Vallee, M. (2004), Generating trading rules on the stock markets with genetic programming, *Computers & Operations Research*, Vol.31, pp. 1033-1047.

[10] Oh, K. J., Kim, T. Y., Min, S. -H. & Lee, H. Y. (2006), Portfolio algorithm based on portfolio beta using genetic algorithm, *Expert Systems with Application*, Vol.30, pp. 527-534.

[11] Mabu, S., Hatakeyama, H., Thu, M. T., Hirasawa, K. & Hu, J. (2006), Genetic Network Programming with Reinforcement Learning and Its Application to Making Mobile Robot Behavior, *IEEJ Trans. EIS*, Vol.126, No.8, pp. 1009-1015.

[12] Lee, K. H. & Jo, G.S. (1999), Expert system for predicting stock market timing using a candlestick chart, *Expert Systems with Applications*, Vol.16, pp. 357-364.

[13] Izumi, Y., Yamaguchi, T., Mabu, S., Hirasawa, K. & Hu, J. (2006), Trading Rules on the Stock Market using Genetic Network Programming with Candlestick Chart, *Proceedings of 2006 IEEE Congress on Evolutionary Computation*, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp. 8531-8536, July 16-21.

[14] Mabu, S., Izumi, Y., Hirasawa, K. & Furuzuki, T. (2007), Trading Rules on Stock Markets Using Genetic Network Progamming with Candle Chart, *T. SICE*, Vol.43, No.4, pp. 317-322, (in Japanese).

[15] Izumi, Y., Hirasawa, K. & Furuzuki, T. (2006), Trading Rules on the Stock Markets Using Genetic Network Progamming with Importance Index, *T. SICE*, Vol.42, No.5, pp. 559-566, (in Japanese).

[16] Dhar, V. (2001), A Comparison of GLOWER and Other Machine Learning Methods for Investment Decision Making, *Springer Berlin Press*, pp.208-220.

[17] Duerson, S., Khan, F. S., Kovalev, V. & Malik, A. H. (2005), Reinforcement Learning in Online Stock Trading Systems.
http://www.cc.gatech.edu/grads/h/hisham/projects/ml7641/RLStockTrading. pdf

[18] Pafka, S., Potters, M. & Kondor, I. (2004), Exponential Weighting and Random-Matrix-Theory-Based Filtering of Financial Covariance Matrices for Portfolio Optimization, arXiv:cond-mat/0402573v1, 2004. *Quantitative Finance*, (to be appeared).

[19] Basalto, N., Bellotti, R., De Carlo, F., Facchi, P. & Pascazio, S. (2005), Clustering stock market companies via chaotic map synchronization, Physica A, 345, p. 196, arXiv:cond-mat/0404497v1.

[20] Huang, W., Nakamori, Y. & Wang, S. Y. (2005), Forecasting stock market movement direction with support vector machine Source, *Computers and Operations Research*, Vol.32, Issue 10, pp. 2513-2522.

[21] Porecha, M. B., Panigrahi, P. K., Parikh, J. C., Kishtawal, C. M. & Basu, S. (2005), Forecasting non-stationary financial time series through genetic algorithm, arXiv:nlin/0507037v1.

[22] Jensen, M. H., Johansen, A., Petroni, F. & Simonsen, I. (2004), Inverse Statistics in the Foreign Exchange Market, Physica A, 340, p. 678, arXiv:cond-mat/0402591v2.

[23] Mikosch, T. & Starica, C. (2004), Stock Market Risk-Return Inference. An Unconditional Non-parametric Approach, *SSRN Working Paper Series*.

[24] Iba, H. & Sasaki, T. (2001), Using Genetic Programming to Predict Financial Data, *Proceedings of the Congress of Evolutionary Computation*, pp. 244-251.

Heuristic Dynamic Programming Nonlinear Optimal Controller

Asma Al-tamimi, Murad Abu-Khalaf and Frank Lewis
The Hashemite University, Math work, The University of Texas at Arlington
Jordan, USA

1. Introduction

This chapter is concerned with the application of approximate dynamic programming techniques (ADP) to solve for the value function, and hence the optimal control policy, in discrete-time nonlinear optimal control problems having continuous state and action spaces. ADP is a reinforcement learning approach (Sutton & Barto, 1998) based on adaptive critics (Barto et al., 1983), (Widrow et al., 1973) to solve dynamic programming problems utilizing function approximation for the value function. ADP techniques can be based on value iterations or policy iterations. In contrast with value iterations, policy iterations require an initial stabilizing control action, (Sutton & Barto, 1998). (Howard, 1960) proved convergence of policy iteration for Markov Decision Processes with discrete state and action spaces. Lookup tables are used to store the value function iterations at each state. (Watkins, 1989) developed Q-learning for discrete state and action MDPs, where a 'Q function' is stored for each state/action pair, and model dynamics are not needed to compute the control action.
ADP was proposed by (Werbos, 1990,1991,1992) for discrete-time dynamical systems having continuous state and action spaces as a way to solve optimal control problems, (Lewis & Syrmos, 1995), forward in time. (Bertsekas & Tsitsiklis, 1996) provide a treatment of Neurodynamic programming, where neural networks (NN) are used to approximate the value function. (Cao, 2002) presents a general theory for learning and optimization.
(Werbos, 1992) classified approximate dynamic programming approaches into four main schemes: Heuristic Dynamic Programming (HDP), Dual Heuristic Dynamic Programming (DHP), Action Dependent Heuristic Dynamic Programming (ADHDP), (a continuous-state-space generalization of Q-learning (Watkins, 1989)), and Action Dependent Dual Heuristic Dynamic Programming (ADDHP). Neural networks are used to approximate the value function (the critic NN) and the control (the action NN), and backpropagation is used to tune the weights until convergence at each iteration of the ADP algorithm. An overview of ADP is given in (Si et al., 2004) (e.g. (Ferrari & Stengel, 2004), and also (Prokhorov & Wunsch, 1997), who deployed new ADP schemes known as Globalized-DHP (GDHP) and ADGDHP.
ADP for linear systems has received ample attention. An off-line policy iteration scheme for discrete-time systems with known dynamics was given in (Hewer, 1971) to solve the discrete-time Riccati equation. In (Bradtke et al, 1994) implemented an (online) Q-learning policy iteration method for discrete-time linear quadratic regulator (LQR) optimal control

problems. A convergence proof was given. (Hagen, 1998) discussed, for the LQR case, the relation between the Q-learning method and model-based adaptive control with system identification. (Landelius, 1997) applied HDP, DHP, ADHDP and ADDHP value iteration techniques, called greedy policy iterations therein, to the discrete-time LQR problem and verified their convergence. It was shown that these iterations are in fact equivalent to iterative solution of an underlying algebraic Riccati equation, which is known to converge (Lancaster & Rodman, 1995). (Lu & Balakrishnan, 2000) showed convergence of DHP for the LQR case.

(Morimoto et al, 2003) developed differential dynamic programming, a Q-learning method, to solve optimal zero-sum game problems for nonlinear systems by taking the second-order approximation to the Q function. This effectively provides an exact Q-learning formulation for linear systems with minimax value functions. In our previous work (Al-tamimi et al, 2007), we studied ADP value iteration techniques to solve the zero-sum game problem for linear discrete-time dynamical systems using quadratic minimax cost. HDP, DHP, ADHDP and ADDHP formulations were developed for zero-sum games, and convergence was proven by showing the equivalence of these ADP methods to iterative solution of an underlying Game Algebraic Riccati Equation, which is known to converge. Applications were made to H-infinity control.

For nonlinear systems with continuous state and action spaces, solution methods for the dynamic programming problem are more sparse. Policy iteration methods for optimal control for continuous-time systems with continuous state space and action spaces were given in (Abu-khalaf & Lewis, 2005) (Abu-Khalaf at el, 2004), but complete knowledge of the plant dynamics is required. The discrete-time nonlinear optimal control solution relies on solving the discrete-time (DT) Hamilton-Jacobi-Bellman (HJB) equation (Lewis & Syrmos, 1995), exact solution of which is generally impossible for nonlinear systems. Solutions to the DT HJB equation with known dynamics and continuous state space and action space were given in (Huang, 1999), where the coefficients of the Taylor series expansion of the value function are systematically computed. In (Chen & Jagannathan, 2005), the authors show that under certain conditions a second-order approximation of the discrete-time (DT) Hamilton-Jacobi-Bellman (HJB) equation can be considered; under those conditions discussed in that paper, the authors solve for the value function that satisfies the second order expansion of the DT HJB instead of solving for the original DT HJB. The authors apply a policy iteration scheme on this second order DT HJB and require an initially stable policy to start the iterations scheme. The authors also used a single (critic) neural network to approximate the value function of the second order DT HJB. These are all off-line methods for solving the HJB equations that require full knowledge of the system dynamics.

Convergence proofs for the on-line value-iteration based ADP techniques for nonlinear discrete-time systems are even more limited. (Prokhorov & Wunsch, 1997) use NN to approximate both the value (e.g. a critic NN) and the control action. Least mean squares is used to tune the critic NN weights and the action NN weights. Stochastic approximation is used to show that, at each iteration of the ADP algorithm, the critic weights converge. Likewise, at each iteration the action NN weights converge, but overall convergence of the ADP algorithm to the optimal solution is not demonstrated. A similar approach was used in (Si et al., 2004).

In (He & Jagannathan, 2005), a generalized or asynchronous version of ADP (in the sense of (Sutton & Barto, 1998) was used whereby the updates of the critic NN and action NN are interleaved, each NN being updated at each time step. Tuning was performed online. A Lyapunov approach was used to show that the method yields uniform ultimate bounded stability and that the weight estimation errors are bounded, though convergence to the exact optimal value and control was not shown. The input coupling function must be positive definite.

In this chapter, we provide a full, rigorous proof of convergence of the online value-iteration based HDP algorithm, to solve the DT HJB equation of the optimal control problem for general nonlinear discrete-time systems. It is assumed that at each iteration, the value update and policy update equations can be exactly solved. Note that this is true in the specific case of the LQR, where the action is linear and the value quadratic in the states. For implementation, two NN are used- the critic NN to approximate the value and the action NN to approximate the control. Full knowledge of the system dynamics is not needed to implement the HDP algorithm; in fact, the internal dynamics information is not needed. As a value iteration based algorithm, of course, an initial stabilizing policy is not needed for HDP.

The point is stressed that these results also hold for the special LQR case of linear systems $\dot{x} = Ax + Bu$ and quadratic utility. In the general folklore of HDP for the LQR case, only a single NN is used, namely a critic NN, and the action is updated using a standard matrix equation derived from the stationarity condition (Lewis & Syrmos1995). In the DT case, this equation requires the use of both the plant matrix A, e.g. the internal dynamics, and the control input coupling matrix B. However, by using a second action NN, the knowledge of the A matrix is not needed. This important issue is clarified herein.

Section two of the chapter starts by introducing the nonlinear discrete-time optimal control problem. Section three demonstrates how to setup the HDP algorithm to solve for the nonlinear discrete-time optimal control problem. In Section four, we prove the convergence of HDP value iterations to the solution of the DT HJB equation. In Section five, we introduce two neural network parametric structures to approximate the optimal value function and policy. As is known, this provides a procedure for implementing the HDP algorithm. We also discuss in that section how we implement the algorithm without having to know the plant internal dynamics. Finally, Section six presents two examples that show the practical effectiveness of the ADP technique. The first example in fact is a LQR example which uses HDP with two NNs to solve the Riccati equation online without knowing the A matrix. The second example considers a nonlinear system and the results are compared to solutions based on State Dependent Riccati Equations (SDRE).

2. The discrete-time HJB equation

Consider an affine in input nonlinear dynamical-system of the form

$$x_{k+1} = f(x_k) + g(x_k)u(x_k). \tag{1}$$

where $x \in \mathbb{R}^n$, $f(x) \in \mathbb{R}^n$, $g(x) \in \mathbb{R}^{n\times m}$ and the input $u \in \mathbb{R}^m$. Suppose the system is drift-free and, without loss of generality, that $x = 0$ is an equilibrium state, e.g. $f(0) = 0$, $g(0) = 0$. Assume that the system (1) is stabilizable on a prescribed compact set $\Omega \in \mathbb{R}^n$.

Definition 1. Stabilizable system: A nonlinear dynamical system is defined to be stabilizable on a compact set $\Omega \in \mathbb{R}^n$ if there exists a control input $u \in \mathbb{R}^m$ such that, for all initial conditions $x_0 \in \Omega$ the state $x_k \to 0$ as $k \to \infty$.

It is desired to find the control action $u(x_k)$ which minimizes the infinite-horizon cost function given as

$$V(x_k)=\sum_{n=k}^{\infty} Q(x_n)+u^T(x_n)Ru(x_n) \tag{2}$$

for all x_k, where $Q(x)>0$ and $R>0\in\mathbb{R}^{m\times m}$. The class of controllers needs to be stable and also guarantee that (2) is finite, *i.e.* the control must be admissible (Abu-Khalaf & Lewis, 2005).

Definition 2. Admissible Control: A control $u(x_k)$ is defined to be admissible with respect to (2) on Ω if $u(x_k)$ is continuous on a compact set $\Omega\in\mathbb{R}^n$, $u(0)=0$, u stabilizes (1) on Ω, and $\forall x_0\in\Omega,\ V(x_0)$ is finite.

Equation (2) can be written as

$$\begin{aligned} V(x_k) &= x_k^TQx_k+u_k^TRu_k+\sum_{n=k+1}^{\infty} x_n^TQx_n+u_n^TRu_n \\ &= x_k^TQx_k+u_k^TRu_k+V(x_{k+1}) \end{aligned} \tag{3}$$

where we require the boundary condition $V(x=0)=0$ so that $V(x_k)$ serves as a Lyapunov function. From Bellman's optimality principle (Lewis & Syrmos, 1995), it is known that for the infinite-horizon optimization case, the value function $V^*(x_k)$ is time-invariant and satisfies the discrete-time Hamilton-Jacobi-Bellman (HJB) equation

$$V^*(x_k)=\min_{u_k}(x_k^TQx_k+u_k^TRu_k+V^*(x_{k+1})) \tag{4}$$

Note that the discrete-time HJB equation develops backward-in time.

The optimal control u^* satisfies the first order necessary condition, given by the gradient of the right hand side of (4) with respect to u as

$$\frac{\partial(x_k^TQx_k+u_k^T\ Ru_k)}{\partial u_k}+\frac{\partial x_{k+1}}{\partial u_k}^T\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}}=0 \tag{5}$$

and therefore

$$u^*(x_k)=\frac{1}{2}R^{-1}g(x_k)^T\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \tag{6}$$

Substituting (6) in (4), one may write the discrete-time HJB as

$$V^*(x_k)=x_k^TQx_k+\frac{1}{4}\frac{\partial V^{*T}(x_{k+1})}{\partial x_{k+1}}g(x_k)R^{-1}g(x_k)^T\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}}+V^*(x_{k+1}) \tag{7}$$

where $V^*(x_k)$ is the value function corresponding to the optimal control policy $u^*(x_k)$. This equation reduces to the Riccati equation in the linear quadratic regulator (LQR) case, which can be efficiently solved. In the general nonlinear case, the HJB cannot be solved exactly.

In the next sections we apply the HDP algorithm to solve for the value function V^* of the HJB equation (7) and present a convergence proof.

3. The HDP algorithm

The HDP value iteration algorithm (Werbos, 1990) is a method to solve the DT HJB online. In this section, a proof of convergence of the HDP algorithm in the general nonlinear discrete-time setting is presented.

3.1 The HDP algorithm

In the HDP algorithm, one starts with an initial value, e.g. $V_0(x)=0$ and then solves for u_0 as follows

$$u_o(x_k)=\arg\min_u(x_k^TQx_k+u^TRu+V_0(x_{k+1})) \tag{8}$$

Once the policy u_0 is determined, iteration on the value is performed by computing

$$\begin{aligned}V_1(x_k)&=x_k^TQx_k+u_0^T(x_k)Ru_0(x_k)+V_0(f(x_k)+g(x_k)u_0(x_k))\\&=x_k^TQx_k+u_0^T(x_k)Ru_0(x_k)+V_0(x_{k+1})\end{aligned} \tag{9}$$

The HDP value iteration scheme therefore is a form of incremental optimization that requires iterating between a sequence of action policies $u_i(x)$ determined by the greedy update

$$\begin{aligned}u_i(x_k)&=\arg\min_u(x_k^TQx_k+u^TRu+V_i(x_{k+1}))\\&=\arg\min_u(x_k^TQx_k+u^TRu+V_i(f(x_k)+g(x_k)u))\\&=\frac{1}{2}R^{-1}g(x_k)^T\frac{\partial V_i(x_{k+1})}{\partial x_{k+1}}\end{aligned} \tag{10}$$

and a sequence $V_i(x)\geq 0$ where

$$\begin{aligned}V_{i+1}(x_k)&=\min_u(x_k^TQx_k+u^TRu+V_i(x_{k+1}))\\&=x_k^TQx_k+u_i^T(x_k)Ru_i(x_k)+V_i(f(x_k)+g(x_k)u_i(x_k))\end{aligned} \tag{11}$$

with initial condition $V_0(x_k)=0$.

Note that, as a value-iteration algorithm, HDP does not require an initial stabilizing gain. This is important as stabilizing gains are difficult to find for general nonlinear systems.

Note that i is the value iterations index, while k is the time index. The HDP algorithm results in an incremental optimization that is implemented forward in time and online. Note that unlike the case for policy iterations in (Hewer, 1971), the sequence $V_i(x_k)$ is not a sequence of cost functions and are therefore not Lyapunov functions for the corresponding policies $u_i(x_k)$ which are in turn not necessarily stabilizing. In Section four it is shown that $V_i(x_k)$ and $u_i(x_k)$ converges to the value function of the optimal control problem and to the corresponding optimal control policy respectively.

3.2 The special case of linear systems

Note that for the special case of linear systems, it can be shown that the HDP algorithm is one way to solve the Discrete-Time Algebraic Riccati Equation (DARE) (Landelius, 1997)). Particularly, for the discrete-time linear system

$$x_{k+1} = Ax_k + Bu_k \tag{12}$$

the DT HJB equation (7) becomes the DARE

$$P = A^T PA + Q - A^T PB(R + B^T PB)^{-1} B^T PA \tag{13}$$

with $V^*(x_k) = x_k^T P x_k$.
In the linear case, the policy update (10) is

$$u_i(x_k) = -(R + B^T P_i B)^{-1} B^T P_i A x_k \tag{14}$$

Substituting this into (11), one sees that the HDP algorithm (10), (11) is equivalent to

$$\begin{aligned} P_{i+1} &= A^T P_i A + Q - A^T P_i B(R + B^T P_i B)^{-1} B^T P_i A \\ P_0 &= 0 \end{aligned} \tag{15}$$

It should be noted that the HDP algorithm (15) solves the DARE forward in time, whereas the dynamic programming recursion appearing in finite-horizon optimal control [21] develops backward in time

$$\begin{aligned} P_k &= A^T P_{k+1} A + Q - A^T P_{k+1} B(R + B^T P_{k+1} B)^{-1} B^T P_{k+1} A \\ P_N &= 0 \end{aligned} \tag{16}$$

where N represents the terminal time. Both equations (15) and (16) will produce the same sequence of P_i and P_k respectively. It has been shown in (Lewis & Syrmos, 1995) and (Lancaster, 1995) that this sequence converges to the solution of the DARE after enough iterations.
It is very important to point out the difference between equations (14) and (15) resulting from HDP value iterations with

$$u_i(x_k) = \underbrace{-(R + B^T P_i B)^{-1} B^T P_i A}_{K_i} x_k \tag{17}$$

$$\begin{aligned} &(A + BK_i)^T P_{i+1}(A + BK_i) - P_{i+1} = -Q - K_i^T R K_i \\ &(P_0, u_0): \text{ Initial stable control policy with corresponding Lyapunov function} \end{aligned} \tag{18}$$

resulting from policy iterations, those in(Hewer, 1971). Unlike P_i in (15), the sequence P_i in (18) is a sequence of Lyapunov functions. Similarly the sequence of control policies in (17) is stabilizing unlike the sequence in (14).

4. Convergence of the HDP algorithm

In this section, we present a proof of convergence for nonlinear HDP. That is, we prove convergence of the iteration (10) and (11) to the optimal value, *i.e.* $V_i \rightarrow V^*$ and $u_i \rightarrow u^*$ as $i \rightarrow \infty$. The linear quadratic case has been proven by (Lancaster, 1995) for the case of known system dynamics.
Lemma 1. Let μ_i be any arbitrary sequence of control policies and Λ_i be defined by

$$\Lambda_{i+1}(x_k) = Q(x_k) + \mu_i^T R \mu_i + \Lambda_i(\underbrace{f(x_k) + g(x_k)\mu_i(x_k)}_{x_{k+1}}) . \tag{19}$$

Let u_i and V_i be the sequences defined by (10) and (11). If $V_0(x_k) = \Lambda_0(x_k) = 0$, then $V_i(x_k) \le \Lambda_i(x_k) \;\; \forall i$.

Proof: Since $u_i(x_k)$ minimizes the right hand side of equation (11) with respect to the control u , and since $V_0(x_k) = \Lambda_0(x_k) = 0$, then by induction it follows that $V_i(x_k) \le \Lambda_i(x_k) \;\; \forall i$. ■

Lemma 2. Let the sequence V_i be defined as in (11). If the system is controllable, then: There exists an upper bound $Y(x_k)$ such that $0 \le V_i(x_k) \le Y(x_k) \;\; \forall i$.

If the optimal control problem (4) is solvable, there exists a least upper bound $V^*(x_k) \le Y(x_k)$ where $V^*(x_k)$ solves (7), and that $\forall i : 0 \le V_i(x_k) \le V^*(x_k) \le Y(x_k)$.

Proof: Let $\eta(x_k)$ be any stabilizing and admissible control policy, and Let $V_0(x_k) = Z_0(x_k) = 0$ where Z_i is updated as

$$\begin{aligned} Z_{i+1}(x_k) &= Q(x_k) + \eta^T(x_k) R \eta(x_k) + Z_i(x_{k+1}) \\ x_{k+1} &= f(x_k) + g(x_k)\eta(x_k) \end{aligned} . \tag{20}$$

It follows that the difference

$$\begin{aligned} Z_{i+1}(x_k) - Z_i(x_k) &= Z_i(x_{k+1}) - Z_{i-1}(x_{k+1}) \\ &= Z_{i-1}(x_{k+2}) - Z_{i-2}(x_{k+2}) \\ &= Z_{i-2}(x_{k+3}) - Z_{i-3}(x_{k+3}) \\ &\;\;. \\ &\;\;. \\ &\;\;. \\ &= Z_1(x_{k+i}) - Z_0(x_{k+i}) \end{aligned} \tag{21}$$

Since $Z_0(x_k) = 0$, it then follows that

$$\begin{aligned} Z_{i+1}(x_k) &= Z_1(x_{k+i}) + Z_i(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_{i-1}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-1}) + Z_{i-2}(x_k) \\ &= Z_1(x_{k+i}) + Z_1(x_{k+i-1}) + Z_1(x_{k+i-2}) + \ldots\ldots + Z_1(x_k) \end{aligned} \tag{22}$$

and equation (22) can be written as

$$\begin{aligned} Z_{i+1}(x_k) &= \sum_{n=0}^{i} Z_1(x_{k+n}) \\ &= \sum_{n=0}^{i} (Q(x_{k+n}) + \eta^T(x_{k+n}) R \eta(x_{k+n})) \\ &\le \sum_{n=0}^{\infty} (Q(x_{k+n}) + \eta^T(x_{k+n}) R \eta(x_{k+n})) \end{aligned} \tag{23}$$

Since $\eta(x_k)$ is an admissible stabilizing controller, $x_{k+n} \to 0$ as $n \to \infty$ and

$$\forall i : \;\; Z_{i+1}(x_k) \le \sum_{i=0}^{\infty} Z_1(x_{k+i}) = Y(x_k)$$

Using Lemma 1 with $\mu_i(x_k)=\eta(x_k)$ and $\Lambda_i(x_k)=Z_i(x_k)$, it follows that

$$\forall i:\ V_i(x_k)\le Z_i(x_k)\le Y(x_k)$$

which proves part a). Moreover if $\eta(x_k)=u^*(x_k)$, then

$$\underbrace{\sum_{n=0}^{\infty}(Q(x_{k+n})+u^{*T}(x_{k+n})Ru^*(x_{k+n}))}_{V^*(x_k)}\le\underbrace{\sum_{n=0}^{\infty}(Q(x_{k+n})+\eta^T(x_{k+n})R\eta(x_{k+n}))}_{Y(x_k)}$$

and hence $V^*(x_k)\le Y(x_k)$ which proves part b) and shows that $\forall i:0\le V_i(x_k)\le V^*(x_k)\le Y(x_k)$ for any $Y(x_k)$ determined by an admissible stabilizing policy $\eta(x_k)$. ■

Theorem 1. Consider the sequence V_i and u_i defined by (11) and (10) respectively. If $V_0(x_k)=0$, then it follows that V_i is a non-decreasing sequence

$$\forall i: V_{i+1}(x_k)\ge V_i(x_k)$$

and as $i\to\infty$

$$V_i\to V^*,\ u_i\to u^*$$

that is the sequence V_i converges to the solution of the DT HJB (7).

Proof: From Lemma 1, let μ_i be any arbitrary sequence of control policies and Λ_i be defined by

$$\Lambda_{i+1}(x_k)=Q(x_k)+\mu_i^T R\mu_i+\Lambda_i(\underbrace{f(x_k)+g(x_k)\mu_i(x_k)}_{x_{k+1}})$$

If $V_0(x_k)=\Lambda_0(x_k)=0$, it follows that $V_i(x_k)\le\Lambda_i(x_k)$ $\forall i$. Now assume that $\mu_i(x_k)=u_{i+1}(x_k)$ such that

$$\begin{aligned}\Lambda_{i+1}(x_k)&=Q(x_k)+\mu_i^T R\mu_i+\Lambda_i(f(x_k)+g(x_k)\mu_i(x_k))\\&=Q(x_k)+u_{i+1}^T Ru_{i+1}+\Lambda_i(f(x_k)+g(x_k)u_{i+1}(x_k))\end{aligned}\tag{24}$$

and consider

$$V_{i+1}(x_k)=Q(x_k)+u_i^T Ru_i+V_i(f(x_k)+g(x_k)u_i(x_k))\tag{25}$$

It will next be proven by induction that if $V_0(x_k)=\Lambda_0(x_k)=0$, then $\Lambda_i(x_k)\le V_{i+1}(x_k)$. Induction is initialized by letting $V_0(x_k)=\Lambda_0(x_k)=0$ and hence

$$\begin{aligned}V_1(x_k)-\Lambda_0(x_k)&=Q(x_k)\\&\ge 0\\V_1(x_k)&\ge\Lambda_0(x_k)\end{aligned}$$

Now assume that $V_i(x_k)\ge\Lambda_{i-1}(x_k)$, then subtracting (24) from (25) it follows that

$$V_{i+1}(x_k)-\Lambda_i(x_k)=V_i(x_{k+1})-\Lambda_{i-1}(x_{k+1})\ge 0$$

and this completes the proof that $\Lambda_i(x_k) \le V_{i+1}(x_k)$.
From $\Lambda_i(x_k) \le V_{i+1}(x_k)$ and $V_i(x_k) \le \Lambda_i(x_k)$, it then follows that

$$\forall i : V_i(x_k) \le V_{i+1}(x_k).$$

From part a) in Lemma 2 and the fact that V_i is a non-decreasing sequence, it follows that $V_i \to V_\infty$ as $i \to \infty$. From part b) of Lemma 2, it also follows that $V_\infty(x_k) \le V^*(x_k)$.
It now remains to show that in fact V_∞ is V^*. To see this, note that from (11) it follows that

$$V_\infty(x_k) = x_k^T Q x_k + u_\infty^T(x_k) R u_\infty(x_k) + V_\infty(f(x_k) + g(x_k) u_\infty(x_k))$$

and hence

$$V_\infty(f(x_k) + g(x_k) u_\infty(x_k)) - V_\infty(x_k) = -x_k^T Q x_k - u_\infty^T(x_k) R u_\infty(x_k)$$

and therefore $V_\infty(x_k)$ is a Lyapunov function for a stabilizing and admissible policy $u_\infty(x_k) = \eta(x_k)$. Using part b) of Lemma 2 it follows that $V_\infty(x_k) = Y(x_k) \ge V^*(x_k)$. This implies that $V^*(x_k) \le V_\infty(x_k) \le V^*(x_k)$ and hence $V_\infty(x_k) = V^*(x_k)$, $u_\infty(x_k) = u^*(x_k)$. ■

5. Neural network approximation for Value and Action

We have just proven that the nonlinear HDP algorithm converges to the value function of the DT HJB equation that appears in the nonlinear discrete-time optimal control.
It was assumed that the action and value update equations (10), (11) can be exactly solved at each iteration. In fact, these equations are difficult to solve for general nonlinear systems. Therefore, for implementation purposes, one needs to approximate u_i, V_i at each iteration. This allows approximate solution of (10), (11).
In this section, we review how to implement the HDP value iterations algorithm with two parametric structures such as neural networks (Werbos, 1990) and (Lewis & Jaganathan, 1999). The important point is stressed that the use of two NN, a critic for value function approximation and an action NN for the control, allows the implementation of HDP in the LQR case *without knowing* the system internal dynamics matrix A. This point is not generally appreciated in the folklore of ADP.

5.1 NN approximation for implementation of HDP algorithm for nonlinear systems

It is well known that neural networks can be used to approximate smooth functions on prescribed compact sets (Hornik & Stinchcombe, 1990). Therefore, to solve (11) and (10), $V_i(x)$ is approximated at each step by a critic NN

$$\hat{V}_i(x) = \sum_{j=1}^{L} w_{vi}^j \phi_j(x) = W_{Vi}^T \boldsymbol{\phi}(x) \tag{26}$$

and $u_i(x)$ by an action NN

$$\hat{u}_i(x) = \sum_{j=1}^{M} w_{ui}^j \sigma_j(x) = W_{ui}^T \boldsymbol{\sigma}(x) \tag{27}$$

where the activation functions are respectively $\phi_j(x), \sigma_j(x) \in C^1(\Omega)$. Since it is required that $V_i(x=0)=0$ and $u_i(x=0)=0$, we select activation functions with $\phi_j(0)=0, \sigma_j(0)=0$. Moreover, since it is known that V^* is a Lyapunov function, and Lyapunov proofs are convenient if the Lyapunov function is symmetric and positive definite, it is convenient to also require that the activation functions for the critic NN be symmetric, i.e. $\phi_j(x)=\phi_j(-x)$. The neural network weights in the critic NN (26) are w_{vi}^j. L is the number of hidden-layer neurons. The vector $\boldsymbol{\phi}(x) \equiv [\phi_1(x)\,\phi_2(x) \cdots \phi_L(x)]^T$ is the vector activation function and $W_{Vi} \equiv \left[w_{vi}^1\, w_{vi}^2 \cdots w_{vi}^L\right]^T$ is the weight vector at iteration i. Similarly, the weights of the neural network in (27) are w_{ui}^j. M is the number of hidden-layer neurons. $\boldsymbol{\sigma}(x) \equiv [\sigma_1(x)\,\sigma_2(x) \cdots \sigma_L(x)]^T$ is the vector activation function, and $W_{ui} \equiv \left[w_{ui}^1\, w_{ui}^2 \cdots w_{ui}^L\right]^T$ is the vector weight.

According to (11), the critic weights are tuned at each iteration of HDP to minimize the residual error between $\hat{V}_{i+1}(x_k)$ and the target function defined in equation (28) in a least-squares sense for a set of states x_k sampled from a compact set $\Omega \subset \mathbb{R}^n$.

$$\begin{aligned} d(x_k, x_{k+1}, W_{Vi}, W_{ui}) &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + \hat{V}_i(x_{k+1}) \\ &= x_k^T Q x_k + \hat{u}_i^T(x_k) R \hat{u}_i(x_k) + W_{Vi}^T \boldsymbol{\phi}(x_{k+1}) \end{aligned} \tag{28}$$

The residual error (c.f. temporal difference error) becomes

$$\left(W_{Vi+1}^T \boldsymbol{\phi}(x_k) - d(x_k, x_{k+1}, W_{Vi}, W_{ui})\right) = e_L(x). \tag{29}$$

Note that the residual error in (29) is explicit, in fact linear, in the tuning parameters W_{Vi+1}. Therefore, to find the least-squares solution, the method of weighted residuals may be used [11]. The weights W_{Vi+1} are determined by projecting the residual error onto $de_L(x)/dW_{Vi+1}$ and setting the result to zero $\forall x \in \Omega$ using the inner product, *i.e.*

$$\left\langle \frac{de_L(x)}{dW_{Vi+1}}, e_L(x) \right\rangle = 0, \tag{30}$$

where $\langle \mathrm{f,g} \rangle = \int_\Omega f g^T dx$ is a Lebesgue integral. One has

$$0 = \int_\Omega \phi(x_k)\left(\phi^T(x_k) W_{Vi+1} - d^T(x_k, x_{k+1}, W_{Vi}, W_{ui})\right) dx_k \tag{31}$$

Therefore a unique solution for W_{Vi+1} exists and is computed as

$$W_{Vi+1} = \left(\int_\Omega \phi(x_k) \phi(x_k)^T dx \right)^{-1} \int_\Omega \phi(x_k) d^T(\phi(x_k), W_{Vi}, W_{ui}) dx \tag{32}$$

To use this solution, it is required that the outer product integral be positive definite. This is known as a persistence of excitation condition in system theory. The next assumption is standard in selecting the NN activation functions as a basis set.

Assumption 1. The selected activation functions $\left\{\phi_j(x)\right\}^L$ are linearly independent on the compact set $\Omega \subset \mathbb{R}^n$.

Assumption 1 guarantees that excitation condition is satisfied and hence $\int_{\Omega} \phi(x_k)\phi(x_k)^T dx$ is of full rank and invertible and a unique solution for (32) exists.

The action NN weights are tuned to solve (10) at each iteration. The use of $\hat{u}_i(x_k, W_{ui})$ from (27) allows the rewriting of equation (10) as

$$W_{ui} = \arg\min_{w}\left(x_k^T Q x_k + \hat{u}_i^T(x_k, w) R \hat{u}_i(x_k, w) + \hat{V}_i(x_{k+1}^i) \right)\Big|_{\Omega} \tag{33}$$

where $x_{k+1}^i = f(x_k) + g(x_k)\hat{u}_i(x_k, w)$ and the notation means minimization for a set of points x_k selected from the compact set $\Omega \in \mathbb{R}^n$.

Note that the control weights W_{ui} appear in (33) in an implicit fashion, *i.e.* it is difficult to solve explicitly for the weights since the current control weights determine x_{k+1}. Therefore, one can use an LMS algorithm on a training set constructed from Ω. The weight update is therefore

$$\begin{aligned} W_{ui}\big|_{m+1} &= W_{ui}\big|_m - \alpha \left. \frac{\partial\left(x_k^T Q x_k + \hat{u}_i^T(x_k, W_{ui}|_m) R \hat{u}_i(x_k, W_{ui}|_m) + \hat{V}_i(x_{k+1})\right)}{\partial W_{ui}} \right|_{W_{ui}|_m} \\ W_{ui}\big|_{m+1} &= W_{ui}\big|_m - \alpha\sigma(x_k)\left(2R\hat{u}_i(x_k, W_{ui}|_m) + g(x_k)^T \frac{\partial\phi(x_{k+1})}{\partial x_{k+1}} W_{Vi} \right)^T \end{aligned} \tag{34}$$

where α is a positive step size and m is the iteration number for the LMS algorithm. By a stochastic approximation type argument, the weights $W_{ui}\big|_m \Rightarrow W_{ui}$ as $m \Rightarrow \infty$, and satisfy (33). Note that one can use alternative tuning methods such as Newton's method and Levenberg-Marquardt in order to solve (33).

In Figure 1, the flow chart of the HDP iteration is shown. Note that because of the neural network used to approximate the control policy the internal dynamics, *i.e.* $f(x_k)$ is not needed. That is, the internal dynamics can be unknown.

Remark. Neither $f(x)$ nor $g(x)$ is needed to update the critic neural network weights using (32). Only the input coupling term $g(x)$ is needed to update the action neural network weights using (34). Therefore the proposed algorithm works for system with partially unknown dynamics- no knowledge of the internal feedback structure $f(x)$ is needed.

5.2 HDP for Linear Systems Without Knowledge of Internal Dynamics

The general practice in the HDP folklore for linear quadratic systems is to use a critic NN to approximate the value, and update the critic weights using a method such as the batch update (32), or a recursive update method such as LMS. In fact, the critic weights are nothing but the elements of the Riccati matrix and the activation functions are quadratic polynomials in terms of the states. Then, the policy is updated using

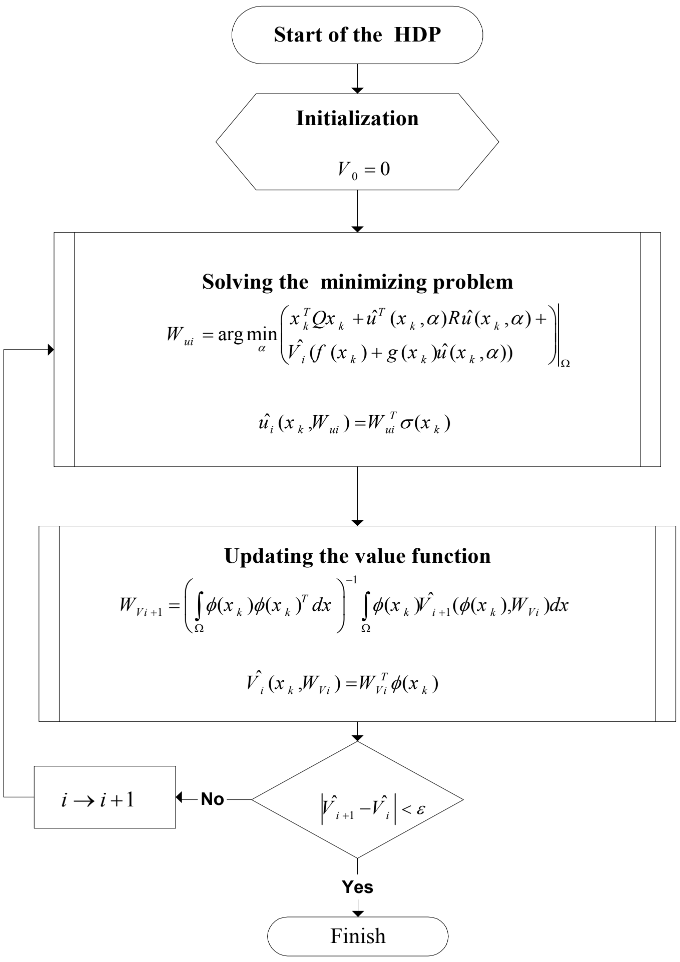

Start of the HDP
Initialization
$V_0 = 0$
Solving the minimizing problem
$W_{ui} = \arg\min_{\alpha} \begin{pmatrix} x_k^T Q x_k + \hat{u}^T(x_k,\alpha) R \hat{u}(x_k,\alpha) + \\ \hat{V}_i(f(x_k) + g(x_k)\hat{u}(x_k,\alpha)) \end{pmatrix}\Bigg|_{\Omega}$
$\hat{u}_i(x_k, W_{ui}) = W_{ui}^T \sigma(x_k)$
Updating the value function
$W_{Vi+1} = \left(\int_{\Omega} \phi(x_k)\phi(x_k)^T dx \right)^{-1} \int_{\Omega} \phi(x_k)\hat{V}_{i+1}(\phi(x_k), W_{Vi}) dx$
$\hat{V}_i(x_k, W_{Vi}) = W_{Vi}^T \phi(x_k)$
$\left|\hat{V}_{i+1} - \hat{V}_i\right| < \varepsilon$
No
$i \to i+1$
Yes
Finish

Fig. 1. Flow chart shows the proposed algorithm

$$u_i(x_k) = -(R + B^T P_i B)^{-1} B^T P_i A x_k \tag{35}$$

Note that this equation requires the full knowledge of both the internal dynamics matrix A and the control weighting matrix B . However, we have just seen (see remark above) that the knowledge of the A matrix can be avoided by using, instead of the action update(35), a second NN for the action

$$\hat{u}_i(x) = W_{ui}^T \boldsymbol{\sigma}(x)$$

In fact the action NN approximates the effects of A and B given in (35), and so effectively learns the A matrix.
That is, using two NN even in the LQR case avoids the need to know the internal dynamics A. In fact, in the next section we give a LQR example, and only the input coupling matrix B is needed for the HDP algorithm. Nevertheless, the HDP converges to the correct LQR Riccati solution matrix P.

6. Simulation examples

In this section, two examples are provided to demonstrate the solution of the DT HJB equation. The first example will be a linear quadratic regulator, which is a special case of the nonlinear system. It is shown that using two NN allows one to compute the optimal value and control (i.e. the Riccati equation solution) online *without knowing the system matrix A* . The second example is for a DT nonlinear system. MATLAB is used in the simulations to implement some of the functions discussed in the chapter.

6.1 Unstable multi-input linear system example

In this example we show the power of the proposed method by using an unstable multi-input linear system. We also emphasize that the method does not require knowledge of the system A matrix, since two neural networks are used, one to provide the action. = This is in contrast to normal methods of HDP for linear quadratic control used in the literature, where the A matrix is needed to update the control policy.
Consider the linear system

$$x_{k+1} = Ax_k + Bu_k \ . \tag{36}$$

It is known that the solution of the optimal control problem for the linear system is quadratic in the state and given as

$$V^*(x_k) = x_k^T P x_k$$

where P is the solution of the ARE. This example is taken from (Stevens & Lewis, 2003), a linearized model of the short-period dynamics of an advanced (CCV-type) fighter aircraft. The state vector is

$$x = [\alpha \quad q \quad \gamma \quad \delta_e \quad \delta_f]^T$$

where the state components are, respectively, angel of attack, pitch rate, flight-path, elevator deflection and flaperon deflection. The control input are the elevator and the flaperon and given as

$$u = [\delta_{ec} \quad \delta_{fc}]^T$$

The plant model is a discretized version of a continuous-time model given in (Bradtke & Ydestie, 1994)]

$$A = \begin{bmatrix} 1.0722 & 0.0954 & 0 & -0.0541 & -0.0153 \\ 4.1534 & 1.1175 & 0 & -0.8000 & -0.1010 \\ 0.1359 & 0.0071 & 1.0 & 0.0039 & 0.0097 \\ 0 & 0 & 0 & 0.1353 & 0 \\ 0 & 0 & 0 & 0 & 0.1353 \end{bmatrix}$$

$$B=\begin{bmatrix} -0.0453 & -0.0175 \\ -1.0042 & -0.1131 \\ 0.0075 & 0.0134 \\ 0.8647 & 0 \\ 0 & 0.8647 \end{bmatrix}$$

Note that system is not stable and with two control inputs. The proposed algorithm does not require a stable initial control policy. The ARE solution for the given linear system is

$$P=\begin{bmatrix} 55.8348 & 7.6670 & 16.0470 & -4.6754 & -0.7265 \\ 7.6670 & 2.3168 & 1.4987 & -0.8309 & -0.1215 \\ 16.0470 & 1.4987 & 25.3586 & -0.6709 & 0.0464 \\ -4.6754 & -0.8309 & -0.6709 & 1.5394 & 0.0782 \\ -0.7265 & -0.1215 & 0.0464 & 0.0782 & 1.0240 \end{bmatrix} \tag{37}$$

and the optimal control $u_k^* = Lx_k$, where L is

$$L=\begin{bmatrix} -4.1136 & -0.7170 & -0.3847 & 0.5277 & 0.0707 \\ -0.6315 & -0.1003 & 0.1236 & 0.0653 & 0.0798 \end{bmatrix} \tag{38}$$

For the LQR case the value is quadratic and the control is linear. Therefore, we select linear activation functions for the action NN and quadratic polynomial activations for the critic NN. The control is approximated as follows

$$\hat{u}_i = W_{ui}^T \sigma(x_k) \tag{39}$$

where W_u is the weight vector, and the $\sigma(x_k)$ is the vector activation function and is given by

$$\sigma^T(x) = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}$$

and the weights are

$$W_u^T = \begin{bmatrix} w_u^{1,1} & w_u^{1,2} & w_u^{1,3} & w_u^{1,4} & w_u^{1,5} \\ w_u^{2,1} & w_u^{2,2} & w_u^{2,3} & w_u^{2,4} & w_u^{2,5} \end{bmatrix}$$

The control weights should converge to

$$\begin{bmatrix} w_u^{1,1} & w_u^{1,2} & w_u^{1,3} & w_u^{1,4} & w_u^{1,5} \\ w_u^{2,1} & w_u^{2,2} & w_u^{2,3} & w_u^{2,4} & w_u^{2,5} \end{bmatrix} = -\begin{bmatrix} L_{11} & L_{12} & L_{13} & L_{14} & L_{15} \\ L_{21} & L_{22} & L_{23} & L_{24} & L_{25} \end{bmatrix}$$

The approximation of the value function is given as

$$\hat{V}_i(x_k, W_{Vi}) = W_{Vi}^T \phi(x_k)$$

where W_V is the weight vector of the neural network given by

$$W_V^T = \begin{bmatrix} w_v^1 & w_v^2 & w_v^3 & w_v^4 & w_v^5 & w_v^6 & w_v^7 & w_v^8 & w_v^9 & w_v^{10} & w_v^{11} & w_v^{12} & w_v^{13} & w_v^{14} & w_v^{15} \end{bmatrix}$$

and $\phi(x_k)$ is the vector activation function given by

$$\phi^T(x) =$$
$$\begin{bmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 & x_1x_5 & x_2^2 & x_2x_3 & x_4x_2 & x_2x_5 & x_3^2 & x_3x_4 & x_3x_5 & x_4^2 & x_4x_5 & x_5^2 \end{bmatrix}$$

In the simulation the weights of the value function are related to the P matrix given in (37) as follows

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} & P_{15} \\ P_{21} & P_{22} & P_{23} & P_{24} & P_{25} \\ P_{31} & P_{32} & P_{33} & P_{34} & P_{35} \\ P_{41} & P_{42} & P_{43} & P_{44} & P_{45} \\ P_{51} & P_{52} & P_{53} & P_{54} & P_{55} \end{bmatrix} = \begin{bmatrix} w_v^1 & 0.5w_v^2 & 0.5w_v^3 & 0.5w_v^4 & 0.5w_v^5 \\ 0.5w_v^2 & w_v^6 & 0.5w_v^7 & 0.5w_v^8 & 0.5w_v^9 \\ 0.5w_v^3 & 0.5w_v^7 & w_v^{10} & 0.5w_v^{11} & 0.5w_v^{12} \\ 0.5w_v^4 & 0.5w_v^8 & 0.5w_v^{11} & w_v^{13} & 0.5w_v^{14} \\ 0.5w_v^5 & 0.5w_v^9 & 0.5w_v^{12} & 0.5w_v^{14} & w_v^{15} \end{bmatrix}$$

The value function weights converge to

$$W_V^T = [55.5411 \quad 15.2789 \quad 31.3032 \quad -9.3255 \quad -1.4536 \quad 2.3142 \quad 2.9234 \quad -1.6594 \quad -0.2430$$

$$24.8262 \quad -1.3076 \quad 0.0920 \quad 1.5388 \quad 0.1564 \quad 1.0240]$$

The control weights converge to

$$W_u = \begin{bmatrix} 4.1068 & 0.7164 & 0.3756 & -0.5274 & -0.0707 \\ 0.6330 & 0.1005 & -0.1216 & -0.0653 & -0.0798 \end{bmatrix}$$

Note that the value function weights converge to the solution of the ARE (37), also the control weights converge to the optimal policy (38) as expected.

6.2 Nonlinear system example

Consider the following affine in input nonlinear system

$$x_{k+1} = f(x_k) + g(x_k)u_k \tag{40}$$

where

$$f(x_k) = \begin{bmatrix} 0.2x_k(1)\exp(x_k^2(2)) \\ .3x_k^3(2) \end{bmatrix} \qquad g(x_k) = \begin{bmatrix} 0 \\ -.2 \end{bmatrix}$$

The approximation of the value function is given as

$$\hat{V}_{i+1}(x_k, W_{Vi+1}) = W_{Vi+1}^T\phi(x_k)$$

The vector activation function is selected as

$$\phi(x) = [x_1^2 \quad x_1x_2 \quad x_2^2 \quad x_1^4 \quad x_1^3x_2$$
$$x_1^2x_2^2 \quad x_1x_2^3 \quad x_2^4 \quad x_1^6 \quad x_1^5x_2 \quad x_1^4x_2^2$$
$$x_1^3x_2^3 \quad x_1^2x_2^4 \quad x_1x_2^5 \quad x_2^6]$$

and the weight vector is

$$W_V^T = \begin{bmatrix} w_v^1 & w_v^2 & w_v^3 & w_v^4 & ..... & w_v^{15} \end{bmatrix}.$$

The control is approximated by

$$\hat{u}_i = W_{ui}^T \sigma(x_k)$$

where the vector activation function is

$$\sigma^T(x) = [x_1 \quad x_2 \quad x_1^3 \quad x_1^2 x_2 \quad x_1 x_2^2$$
$$x_2^3 \quad x_1^5 \quad x_1^4 x_2 \quad x_1^3 x_2^2 \quad x_1^2 x_2^3$$
$$x_1 x_2^4 \quad x_2^5]$$

and the weights are

$$W_u^T = \begin{bmatrix} w_u^1 & w_u^2 & w_u^3 & w_u^4 & ..... & w_u^{12} \end{bmatrix}.$$

The control NN activation functions are selected as the derivatives of the critic activation functions, since the gradient of the critic activation functions appears in (34). The critic activations are selected as polynomials to satisfy $\hat{V}_i(x = 0) = 0$ at each step. Note that then automatically one has $\hat{u}_i(x = 0) = 0$ as required for admissibility. We decided on 6th order polynomials for VFA after a few simulations, where it came clear that 4th order polynomials are not good enough, yet going to 8th order does not improve the results.
The result of the algorithm is compared to the discrete-time State Dependent Riccati Equation (SDRE) proposed in (Cloutier, 1997).
The training sets is $x_1 \in [-2,2], x_2 \in [-1,1]$. The value function weights converged to the following

$$W_V^T = [1.0382 \;\; 0 \;\; 1.0826 \;\; .0028 \;\; -0 \;\; -.053 \;\; 0 \; -.2792$$
$$-.0004 \;\; 0 \;\; -.0013 \;\; 0 \;\; .1549 \;\; 0 \;\; .3034]$$

and the control weights converged to

$$W_u^T = [\,0 \;\; -.0004 \;\; 0 \;\; 0 \;\; 0 \;\; .0651 \;\; 0 \;\; 0 \;\; 0 \;\; -.0003 \;\; 0 \;\; -.0046]$$

The result of the nonlinear optimal controller derived in this chapter is compared to the SDRE approach. Figure 2 and Figure 3 show the states trajectories for the system for both methods.
In Figure 4, the cost function of the SDRE solution and the cost function of the proposed algorithm in this chapter are compared. It is clear from the simulation that the cost function for the control policy derived from the HDP method is lower than that of the SDRE method. In Figure 5, the control signals for both methods are shown.

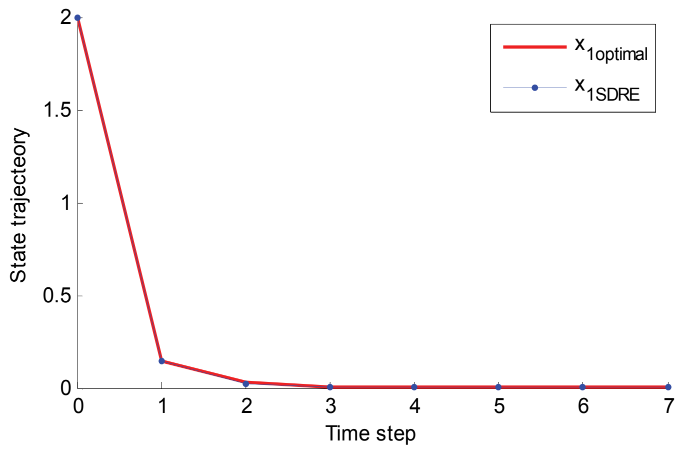


Fig. 2. The state trajectory for both methods

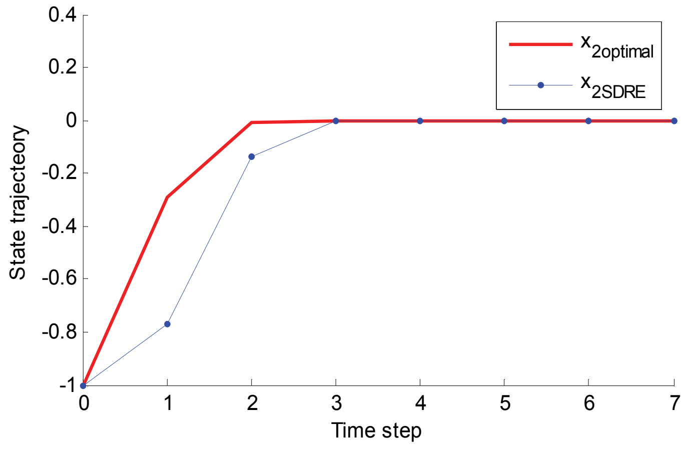


Fig. 3. The state trajectory for both methods

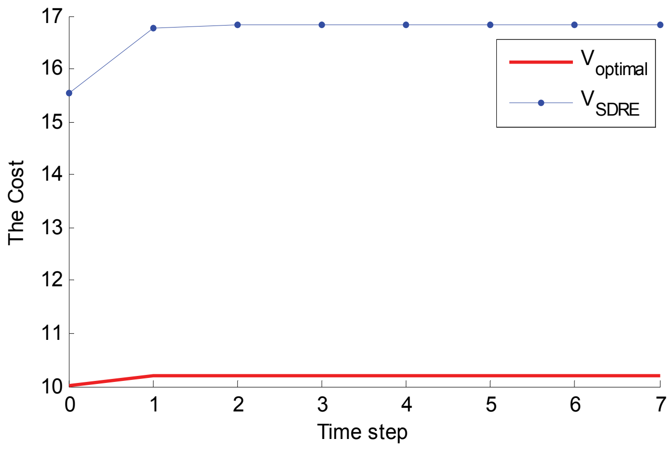


Fig. 4. The cost function for both methods

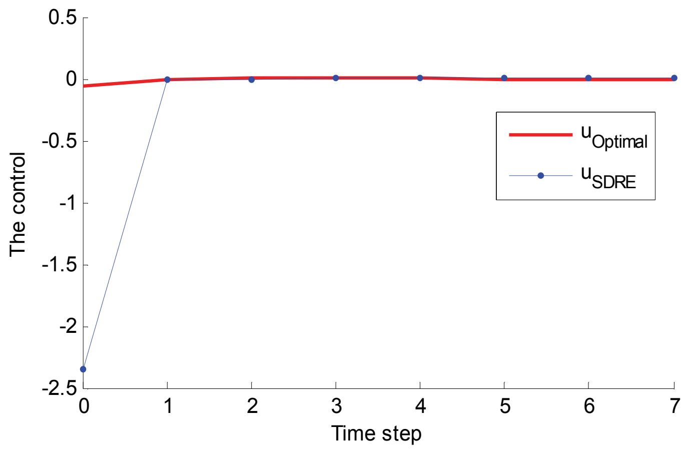


Fig. 5. The control signal input for both methods

7. Conclusion

We have proven convergence of the HDP algorithm to the value function solution of Hamilton-Jacobi-Bellman equation for nonlinear dynamical systems, assuming exact solution of value update and the action update at each iteration.
Neural networks are used as parametric structures to approximate at each iteration the value (i.e. critic NN), and the control action. It is stressed that the use of the second neural network to approximate the control policy, the internal dynamics, *i.e.* $f(x_k)$, is not needed to implement HDP. This holds as well for the special LQR case, where use of two NN avoids the need to know the system internal dynamics matrix *A*. This is not generally appreciated in the folkloric literature of ADP for the LQR. In the simulation examples, it is shown that the linear system critic network converges to the solution of the ARE, and the actor network converges to the optimal policy, without knowing the system matrix *A*. In the nonlinear example, it is shown that the optimal controller derived from the HDP based value iteration method outperforms suboptimal control methods like those found through the SDRE method.

8. References

Abu-Khalaf, M., F. L. Lewis. (2005). Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approach. *Automatica*, vol. 41, pp. 779 – 791.

Abu-Khalaf, M., F. L. Lewis, and J. Huang. (2004).Hamilton-Jacobi-Isaacs formulation for constrained input nonlinear systems. *43rd IEEE Conference on Decision and Control*, 2004, pp. 5034 - 5040 Vol.5.

Al-Tamimi, A. , F. L. Lewis, M. Abu-Khalaf (2007). Model-Free Q-Learning Designs for Discrete-Time Zero-Sum Games with Application to H-Infinity Control. *Automatica*, volume 43, no. 3. pp 473-481.

Al-Tamimi, A., M. Abu-Khalaf, F. L. Lewis. (2007). Adaptive Critic Designs for Discrete-Time Zero-Sum Games with Application to H-Infinity Control. *IEEE Transactions on Systems, Man, Cybernetics-Part B, Cybernaetics*, Vol 37, No 1, pp 240-24.

Barto, A. G., R. S. Sutton, and C. W. Anderson. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 835–846.

Bertsekas, D.P. and J. N. Tsitsiklis.(1996). Neuro-Dynamic Programming. *Athena Scientific*, MA.

Bradtke, S. J., B. E. Ydestie, A. G. Barto (1994). Adaptive linear quadratic control using policy iteration. *Proceedings of the American Control Conference* , pp. 3475-3476, Baltmore, Myrland.

Chen, Z., Jagannathan, S. (2005). Neural Network -based Nearly Optimal Hamilton-Jacobi-Bellman Solution for Affine Nonlinear Discrete-Time Systems. *IEEE CDC 05* ,pp 4123-4128.

Cloutier, J. R. (1997). State –Dependent Riccati equation Techniques: An overview. *Proceeding of the American control conference*, Albuquerque, NM, pp 932-936.

Ferrari, S., Stengel, R.(2004) Model-Based Adaptive Critic Designs. pp 64-94, Eds J. Si, A. Barto, W. Powell, D. Wunsch *Handbook of Learning and Approximate Dynamic Programming*, Wiley.

Finlayson, B. A. (1972). The *Method of Weighted Residuals and Variational Principles*. Academic Press, New York.

Hagen, S. B Krose. (1998). Linear quadratic Regulation using Reinforcement Learning. *Belgian_Dutch Conference on Mechanical Learning*, pp. 39-46.

He, P. and S. Jagannathan.(2005).Reinforcement learning-basedoutput feedback control of nonlinear systems with input constraints. *IEEE Trans. Systems, Man, and Cybernetics -Part B:Cybernetics*, vol. 35, no.1, pp. 150-154.

Hewer, G. A. (1971). An iterative technique for the computation of the steady state gains for the discrete optimal regulator. *IEEE Trans. Automatic Control*, pp. 382-384.

Hornik, K., M. Stinchcombe, H. White.(1990) .Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks. *Neural Networks*, vol. 3, pp. 551-560.

Howard, R. (1960). *Dynamic Programming and Markov Processes.*, MIT Press, Cambridge, MA.

Huang, J. (1999). An algorithm to solve the discrete HJI equation arising in the L_2-gain optimization problem. INT. J. Control, Vol 72, No 1, pp 49-57.

Kwon, W. H and S. Han. (2005). *Receding Horizon Control*, Springer-Verlag, London.

Lancaster, P. L. Rodman. (1995). *Algebraic Riccati Equations.* Oxford University Press, UK.

Landelius, T. (1997). *Reinforcement Learning and Distributed Local Model Synthesis.* PhD Dissertation, Linkoping University, Sweden.

Lewis, F. L., V. L. Syrmos. (1995) Optimal Control, 2nd ed., John Wiley.

Lewis, F. L., Jagannathan, S., & Yesildirek, A. (1999). *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Franci.

Lin W.,and C. I. Byrnes.(1996).H_∞Control of Discrete-Time Nonlinear System. *IEEE Trans. on Automat. Control* , vol 41, No 4, pp 494-510..

Lu, X., S.N. Balakrishnan. (2000). Convergence analysis of adaptive critic based optimal control. *Proc. Amer. Control Conf.*, pp. 1929-1933, Chicago.

Morimoto, J., G. Zeglin, and C.G. Atkeson. (2003). Minimax differential dynamic programming: application to a biped walking robot. *Proc. IEEE Int. Conf. Intel. Robots and Systems*, pp. 1927-1932, Las Vegas.

Murray J., C. J. Cox, G. G. Lendaris, and R. Saeks.(2002).Adaptive Dynamic Programming. *IEEE Trans. on Sys., Man. and Cyb.*, Vol. 32, No. 2, pp 140-153.

Narendra, K.S. and F.L. Lewis. (2001).Special Issue on Neural Network feedback Control. Automatica, vol. 37, no. 8.

Prokhorov, D., D. Wunsch. (1997). Adaptive critic designs. *IEEE Trans. on Neural Networks*, vol. 8, no. 5, pp 997-1007.

Prokhorov, D., D. Wunsch (1997). Convergence of Critic-Based Training. *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, vol. 4, pp. 3057—3060.

Si, J. and Wang. (2001). On-Line learning by association and reinforcement. *IEEE Trans. Neural Networks*, vol. 12, pp.264-276.

Si, Ji. A. Barto, W. Powell, D. Wunsch.(2004). *Handbook of Learning and Approximate Dynamic Programming.* John Wiley, New Jersey.

Stevens B., F. L. Lewis. (2003). Aircraft Control and Simulation, 2nd edition, John Wiley, New Jersey.

Sutton, R. S., A. G. Barto. (19998). *Reinforcement Learning, MIT Press.* Cambridge, MA .

Watkins, C.(1989). *Learning from Delayed Rewards.* Ph.D. Thesis, Cambridge University, Cambridge, England.

Werbos, P. J. (1991). A menu of designs for reinforcement learning over time. , *Neural Networks for Control*, pp. 67-95, ed. W.T. Miller, R.S. Sutton, P.J. Werbos, Cambridge: MIT Press.

Werbos, P. J. (1992). *Approximate dynamic programming for real-time control and neural modeling*. Handbook of Intelligent Control, ed. D.A. White and D.A. Sofge, New York: Van Nostrand Reinhold,.

Werbos, P. J. (1990). Neural networks for control and system identification. *Heuristics,* Vol. 3, No. 1, pp. 18-27.

Widrow, B., N. Gupta, and S. Maitra. (1973). Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 455–465.

Xi-Ren Cao. (2001). Learning and Optimization—From a Systems Theoretic Perspective. *Proc. of IEEE Conference on Decision and Control*, pp. 3367-3371.

Implicit Estimation of Another's Intention Based on Modular Reinforcement Learning

Tadahiro Taniguchi[1], Kenji Ogawa[2] and Tetsuo Sawaragi[3]
[1]*College of Information Science and Engineering,*
[2]*Matsushita Electoronic Industrial Co.*
[3]*Graduate School of Engineering and Science,*
Japan

1. Introduction

When we try to accomplish a collaborative task, e.g., playing football or carrying large tables, we have to share a goal and a way of achieving the goal. Although people accomplish such tasks, achieveing such cooperation is not so easy in the context of a computational multi-agent learning system because participating agents cannot observe another person's intention directly. We cannot know directly what other participants intend to do and how they intend to achieve that. Therefore, we have to notice another participant's intention by utilizing other hints or information. In other words, we have to estimate another's intention to accomplish collaborative tasks.
In particular, in multi-agent reinforcement learning tasks, when another's intention is unobservable the learning process is fatally harmed. When a participating agent of a collaborative task changes its intention and switches or modifies its controller, system dynamics for each agent will inevitably change. If other agents learn on the basis of simple reinforcement learning architecture, they cannot keep up with changes in the task environment because most reinforcement learning architectures assume that environmental dynamics are fixed. To overcome the problem, each agent must have a simple reinforcement learning architecture and some additional capability, which solves the problem. We take the capability of "estimation of another's intention" as an example of such a capability.
Human beings can perform several kinds of collaborative tasks. This means that we have some computational skills, which enable us to estimate another's intention to some extent even if we cannot observe another's intention directly.
The computational model for implicit communication is described in this chapter on the basis of a framework of modular reinforcement learning. The computational model is called situation-sensitive reinforcement learning (SSRL), which is a type of modular reinforcement learning architecture. We assumed that such a distributed learning architecture would be essential for an autonomous agent to cope with a physically dynamic environment and a socially dynamic environment that included changes in another agent's intentions. The skill, estimation of another's intention, seems to be a social skill. However, human adaptability, which we believe our selves to be equipped with to deal with a physically dynamic environment, enables an agent to deal with such a dynamic social environment, including

intentional changes of collaborators. Determining clearlify the computational relationship between the two skills is also a purpose of this study.
The mathematical basis for the implicit estimation of another's intention based on the framework of reinforcement learning is also provided. Furthermore, a simple truck-pushing task performed by a pair of agents is presented to evaluate the learning architecture.

2. Communication and estimation of another's intention

Communicating one's intention to another person enables the other person to estimate one's intention. Therefore, communication and estimation of another's intention are different aspects of the same phenomenon. Implicit estimation is a key idea to supplement the classical communication model, i.e., Shannon-Weaver communication model. Additionally, it is also important to understand a computational mechanism of emergence of communication.

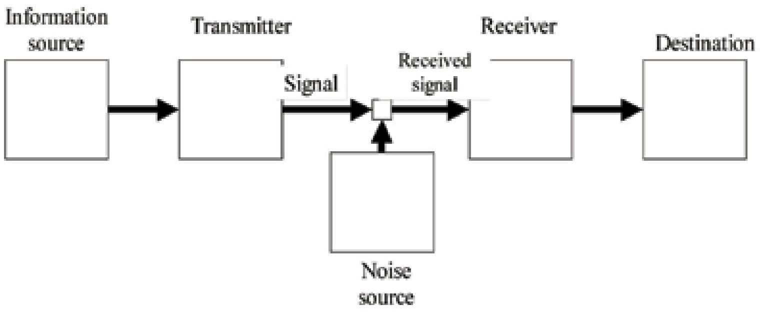


Fig. 1. Schematic diagram of general communication system

We describe the background in this section. In addition to that, an abstract mechanism of the implicit estimation is described on the basis of the notion of multiple internal models.

2.1 Communication models

Shannon formulated "communication" in mathematical terms [5]. In Shannon's communication model, a sender's messages encapsulated in signals or signs are carried through an information channel to a receiver. An encoder owned by the sender encodes the message to the signal by referring to its code table. When a receiver receives the signal, the receiver's decoder decodes the signal back to a message by referring to its code table. After that, the receiver understands the sender's intention and determines what to do. The general communication system described by Shannon is shown in Fig. 1 schematically.
In contrast to Shannon, Peirce, who started "semiotics," insisted that the basis of communication is symbols, and he defined a symbol as a triadic relationship among "sign," "object," and "interpretant"[2]. A "sign" is a signal that represents something to an interpreter. An "object" is something that is represented by the sign, and an "interpretant" is something that relates the sign to the object. In other words, an "interpretant" is a mediator between a "sign" and an "object." The words "sign" and "object" are easy for most people to understand. However, "interpretant" may be difficult to understand. An "interpretant" is sometimes a concept an interpreter comes up with, an action the interpreter takes, or culture in which people consider the sign and object to be related. The important point of Peirce's

semiotics is that the relationship between "sign" and "object" is not fixed. The relationship can be dynamically changing. The relationship simply depends on the "interpretant." The dynamic process by which a sign represents an object mediated by an interpretant is called "semiosis." Peirce's semiotics is thoroughly constructed from the viewpoint of an interpreter. In the framework of Peirce's semiotics, the third element, "interpretant," plays an essential role in communication. In Shannon's communication model, one premise is that a shared code table is required. However, an autonomous agent cannot observe other agents' internal goals or code table. In contrast, Peirce's semiosis does not require such a premise. Semiosis is a phenomenon that emerges inside of an autonomous agent. The participants in a communication must create meaning from incoming signs based on their physical and social experience. Such an individual learning process is considered to supplement symbolic communication. However, semiosis requires autonomous agents to have sufficient adaptability and capability to create meanings from superficial meaningless signs.

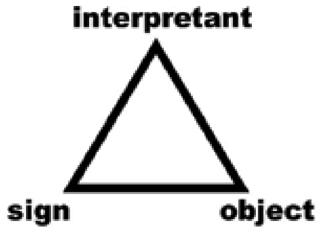


Fig. 2. Semiotic triad

In a human collaborative task, a human participant becomes able to distinguish several situations, which are modified by another's changing intentions. In such a case, the kind of policy the participant should follow in each situation is not clear beforehand. However, if the team continues to collaborate through trial and error, some kind of shared rules will be formed as a kind of habit of the team, and a follwer on the team becomes able to perform adequately by referring to the situation and the habit. This process corresponds to "semiosis" in Peirce's semiotics. Here, "sign," "object," and "interpretant" correspond to a "situation," "the leader's intention," and "acquired rule" or "the follower's action," respectively.

An important point in this scenario is that the "situation" has no meaning before the follower distinguishes situation, performs adequately, and a tacit rule is established between the two agents.

In this chapter, we describe candidates for computational communication models, which are based on Peirce's semiosis.

2.2 Estimation of another's intention

Roughly speaking, we assume there are two ways in which we estimate another's intention. Here, we explain the difference between the two ways of estimating another's intention.

For illustrative purposes, we assume that there is a leader in an organization who makes decisions. The leader makes decisions to direct the team, and followers play their roles based on the decision.

In such a case, the leader communicates his/her intention to the follwers, and followers in the organization have to estimate a leading agent's intention to cope with cooperative tasks.

The communication and the estimation of another's intention are different aspects of the same phenomenon, as we described above. How can followers members estimate the leader's intention? This is the problem.
Here, we take two kinds of estimation of another's intention into consideration. One is "explicit estimation," and the other is "implicit estimation."

2.2.1 Explicit estimation of another's intention

One solution for communicating one's intention to another person is to express one's intention directly with predefined signals, e.g., by pointing to the goal and by commanding the other person to act. The method of communication requires a shared symbolic system as a basic premise. The symbolic system is often called a code table. If the symbolic system used in this communication must be completely shared by the participants in the cooperativetask environment, a participant who receives a message understands exactly what the person transmitting the message wants to do. The receiver of the message can estimate the sender's intentions based on externalized signs. We call this process the "explicit estimation" because the intention of the leader is explicitly expressed as externalized signals. In this communication model, both agents have to share a predefined code table before the tasks. In the explicit estimation model, the accuracy of the communication is measured by the coincidence between the transmitted message and the receiver's interpretation of the sender's message, which is obtained by decoding the incoming signal utilizing the shared code table. The process of estimating another's intention in a collaborative task is shown in Figure 3 schematically. A leader and a follower carry a truck collaboratively. How can the follower estimate the leader's goal using explicit estimation when the leader changes his goal?

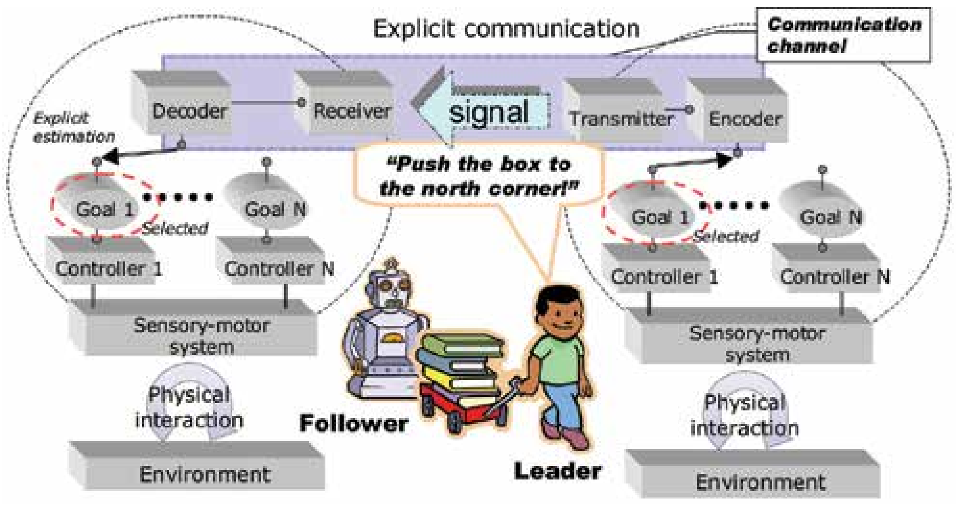


Fig. 3. explicit estimation

First, the leader agent changes his goal. In the explicit estimation scheme, this seems like a natural framework of communication. After Shannon formulated "communication" mathematically, many sociologists and computer scientists have described "communication" as above. However, the communication model based on explicit estimation of another's

intention has two shortcomings. One is that the method of sharing the code table between the two agetns is unknown. If we consider the two agents to be autonomous, neither agent can observe the other agent's internal goals and code table. Therefore, neither agent can utilize a "teacher signal" as feedback of its interpretation to upgrade its code table. The second shortcoming is that the leader agent has to display his intention whenever he changes his goal. These are two problems of explicit estimation of another's intention.
In contrast, when we review what we do in collaborative tasks, we find that we do not always send verbal messages representing our intention to our collaborators. We sometimes execute a collaborative task without saying anything. In this case, the leader's intention is not transmitted to the follower by sending the explicit linguistic sign but through the shared environmental dynamics implicitly. Explicit estimation of another's intention is not the only way of communication. To complement or to support the explicit estimation, implicit estimation is necessary.

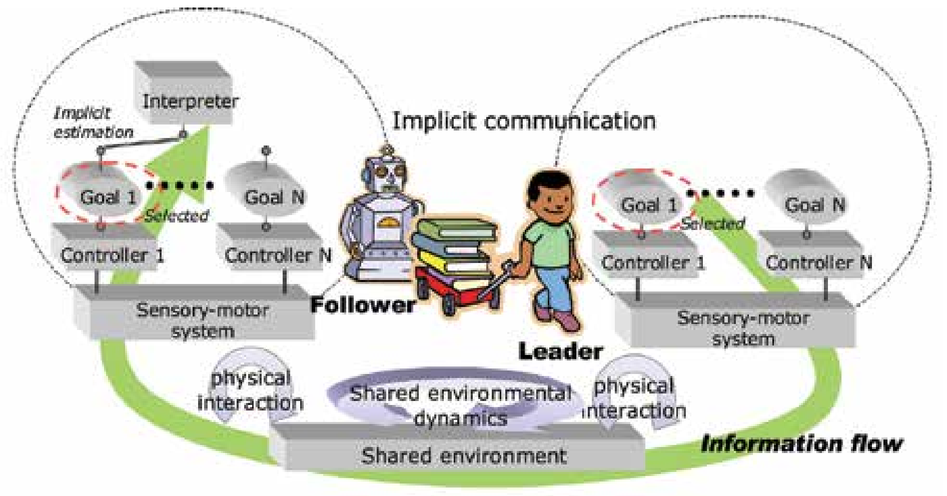


Fig. 4. Implicit communication

2.2.2 Implicit estimation of another's intention

People occasionally undertake collaborative tasks without saying anything. Even if a leader says nothing to members of his organization, they can often perform the task by estimating the leader's intentions on the basis of their observation. We call such an estimation process "implicit estimation" of another's intention. However, if there were no pathways through which information about the leader's intention goes to the followers, the followers could never estimate the leader's intention. One reason followers can estimate the leader's intention is that the action and sensation of the followers are causally related to the leader's intentions.
In other words, sensations a participating agent has after he/she performs actions are affected by the leader's way of acting and another agents' ways of acting. Therefore, subjective environmental dynamics for a participating agent are causally affected by the leader's intention because other agents are assumed to behave based on the leader's intention.

We assume none of the members can observe any information except for their own sensory-motor information. However, they can estimate the leader's intentions. We call this process "implicit estimation." Implicit estimation is achieved by watching how the agent's sensation changes. In control tasks, an agent usually observes state variables.
In what follows, we assume that an agent obtains state variables, e.g., position, velocity, and angle. State variables are usually considered to be objectives to be controlled in many control tasks. However, in implicit communication, state variables also become information media of another agent's intention. An participating agent can estimate another's intention by observing changes in state variables. The information goes through their shared dynamics.
The process of implicit estimation of another's intention is showen in Figure 4, schematically. First, the leader changes his goal. When the leader's goal has changed, his controller, which produces his behavior, is switched. That, of course, affects physical dynamics of the dynamical system shared between the leader and the follower. If a participating agent has a state predictor, he will become aware of the qualitative change in the shared dynamics because his prediction of the state value collapses If physical dynamics are stable, he can predict his state variables consistently. If the follower agent notices the change in subjective physical dynamics, the follower can notice the change in the leader's intention based on the causal relationship between the leader's intention and his facing dynamical system.
Therefore, the capability to predict state variables seems to be required for physical skills and social skills. This scenario suggests the process of learning physical skills to control the target system and the method to communicate with the partner agent might be quite similar in such cooperative tasks.

3. Multiple internal models

Our computational model of implicit estimation of another's intention is based on modular reinforcement learning architecture including multiple internal models. To achieve implicit estimation of another's intention described in the previous section, an agent must have a learning architecture that includes state predictors. We focus on multiple internal models as neural architectures that achieve such an adaptive capability.

3.1 Multiple internal models and social adaptability

Relationships between the human brain's social capability and physical capability are commanding interest. From the viewpoint of computational neuroscience, Wolpert et al. [17, 3] suggested that MOSAIC, which is a modular learning architecture representing a part of the human central nervous system (CNS), acquires multiple internal models that play an essential role in adapting to the physical dynamic environment as well as other roles. We regard this as a candidate for a brain function that connects human physical capability and social capability. An internal model is a learning architecture that predicts the state transition of the environment or other target system. This is a belief that a person can operate his/her body and his/her grasping tool by utilizing an obtained internal model[16]. The internal model is acquired in the cerebellum through interactions. The learning system of internal models is considered to be a kind of schema that assimilates exterior dynamics and accommodates the internal memory system, i.e., internal model. If a person encounters various kinds of environments and/or tools, which have different dynamical properties, the human brain needs to differentiate them and acquire several internal models. However, segmentation of dynamics is not given a priori. Therefore, a learning architecture representing multiple internal models should generate and learn internal models, and recognize changes in physical dynamics in its facing environment at the same time. To describe such a learning system, several computational models have been proposed, e.g., MPFIM [17], the mixture of RNNs[10], RNNPB[9], and the schema model [13]. Most of them are comprised of several learning predictors. The learning architecture switches the predictors and accommodates them through interactions with the environment. Such a learning architecture is often called a modular learning system. The RNNPB is not a modular learning system. Tani insisted internal models should be obtained in a single neural network in a distributed way[9]. In most modular learning architecture, a Bayesian rule is used to calculate the posterior probability in which a current predictor is selected. In contrast, the schema model [13] is a modular learning architecture that does not use a Bayesian rule but hypothesis-testing theory. At the moment, multiple internal models are usually considered to be a learning system for an autonomous system to cope with a physically dynamic environment. Meanwhile, Wolpert et al. addressed a hypothesis that a person utilizes multiple internal models to estimate another's intention from the observation of another's movement. Although these internal models described in the hypothesis seem to add a slightly different feature to the original definition of an internal model, interestingly, the hypothesis tries to connect neural architectures for physical adaptability and social adaptability. Doya et al. [1] proposed a modular learning architecture that enables robots to estimate another's intention and to communicate with each other in a reinforcement learning task.

In addition, when a person performs a collaborative task with others, one can notice changes in another agent's intention by recognizing the change in his/her facing dynamical system without any direct observation of the other agent's movement. This means multiple internal models enable an agent to notice changes in another agent's intention. This usage of multiple internal models does not require adding any features to the original definition of multiple internal models.

3.2 Implicit estimation of another's intention based on multiple internal models

"Intention" in everyday language denotes a number of meanings. Therefore, a perfect computational definition of "intention" is impossible. In this chapter, we simply consider an "intention" as a goal the agent is trying to achieve. In the framework of reinforcement learning, an agent's goal is represented by a reward function. Therefore, an agent who has several intentions has several internal goals, i.e., several internal reward functions, G^m. If an internal reward function, G^m, is selected, a policy, u^m, is selected and modified to maximize the cumulative future internal reward through interactions with the task environment.

In the following, we assume that the collaborative task involves two agents. The system is described as

$$y = f(x, u_1, u_2^m) + n, \quad (1)$$

$$= f(x, u_1, u_2^m(x)) + n, and \quad (2)$$

$$= F^m(x, u_1) + n. \quad (3)$$

Here, x is a state variable, u_i is the i-th agent's motor output, and n is a noise term. We assumed that an agent would not be able to observe another agent's motor output directly. In such cases, environmental dynamics seem to be Eq. 3 to the first agent. If the second agent changes its policy, environmental dynamics for the first agent change. Therefore, in a physically stationary environment, the first agent can establish that the second agent has changed its intention by noticing changes in environmental dynamics.

The discussion can be summarized as follows. If physical environmental dynamics, f, is fixed, agents who have multiple internal models can detect changes in another agent's intentions by detecting changes in subjective environmental dynamics, F. The computational process is equal to the process by which an agent detects changes in the original physical dynamics.

We define "situation" as "how state variable x and motor output u change observed output y." In this case, a change in an agent's intentions leads to a change in the subjective situation of another agent. By utilizing multiple internal models, an agent is expected to differentiate situations and execute adequate actions. In the next section, we describe a concrete modular reinforcement learning architecture named Situation-Sensitive Reinforcement Learning (SSRL).

4. Situation-sensitive reinforcement learning architecture

It is important for autonomous agents to accumulate the results of adaptation to various environments to cope with dynamically changing environments. Acquired concepts, models, and policies should be stored for similar situations that are expected to occur in the near future. Not only learning a certain behavior and/or a certain model, but also the obtained behaviors, policies, and models is essential to describe such a learning process. Many modular learning architectures [7, 4] and hierarchical learning architectures [10, 8] have been proposed to describe this kind of learning process. This section introduces such a modular-learning architecture called the situation-sensitive reinforcement learning architecture (SSRL). This enables an autonomous agent to distinguish changes the agent is facing in situations, and to infer the partner agent's intentions without any teacher signals from the partner.

4.1 Discrimination of intentions based on changes in dynamics

Fig. 5 is an overview of SSRL. SSRL has several state predictors, F^m, representing situations and internal goals, G^m, representing intentions. Each state predictor F^m corresponds to each situation.

$$e_t^j = ||y_t - F^j(x_t, u_t)||^2, \tag{4}$$

$$P(j|\bar{e}_t^j) = \exp(-\frac{\bar{e}_t^j}{2\sigma^2}) / \sum_{k=1}^{p} \exp(-\frac{\bar{e}_t^k}{2\sigma^2}), and \tag{5}$$

$$j^* = \arg\min_j P(j|\bar{e}_t^j), \tag{6}$$

where $\bar{e}_t^j$ is the temporal average of the prediction error, e_t^j, of the j-th state predictor, F_j. If averaged error $\bar{e}_t^j$ has a normal distribution and the system dynamics is F_j, the posterior probability, $P(j \mid \bar{e}_t^j)$, can be defined based on the Bayesian framework above under the condition that there is no other information. If there are no adequate state predictors in SSRL, the SSRL allocates one more state predictor based on hypothesis-testing theory [13].

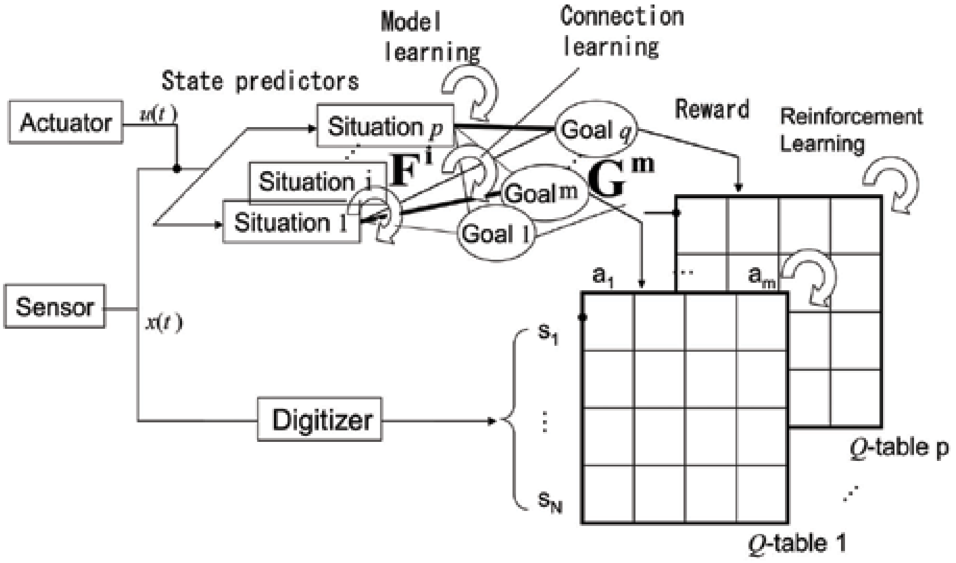


Fig. 5. Situation-Sensitive Reinforcement Learning architecture

We model the state predictors by using locally linear predictors, and we don't estimate the standard deviation σ. The updating rule are switched based on hypothesis testing.

Case 1: $\bar{e}_t^{j^*} < \delta_l$

In this case, the learning system considers that incoming sample data are normal samples for the existing predictors, decides the curret situation j^*, and update the corresponding function F^{j^*} by using assimilated samples.

Case 2: $\bar{e}_t^{j^*} > \delta_u$

In this case, the learning system considers that incoming sample data are outliers for the existing predictors, and prepare a new fnction F^{p+1}. It decides the curret situation j^{p+1}. However, the new predictor is considered as a exeptional state predictor until $\bar{e}_t^{p+1} < \delta_l$. If the predictor's averaged error reaches under δ_l, the function F^{p+1} is taken into a list of existing predictors, and $p \leftarrow p + 1$.

Case 3: $\delta_l < \bar{e}_t^{j^*} < \delta_u$

In this case, the system take no account of the incoming sample.

This is an intermediate method for the MOSAIC model [17, 15], which is based on the Basian framework, and the schema model [13], which is based on hypothesis-testing theory. SSRL detects the current situation based on Eq. 6. During this an adequate state predictor is selected and assimilates the incoming experiences; SSRL acquires the state predictors by ridge regression based on the assimilated experiences.

4.2 Reinforcement learning

Each policy corresponding to a goal is acquired by using reinforcement learning [6]. SSRL uses Q-Learning [14] in this paper. This method can be used to estimate the state-action value function, $Q(s, a)$, through interactions with the agent's environment. The optimal state-action value function directly gives the optimal policy. When we define $\mathcal{S}$ as a set of state variables and $\mathcal{A}$ as a set of motor outputs, and we assume the environment consists of a Markov decision process, the algorithm for Q-learning is described as

$$\begin{aligned} Q &\leftarrow Q(s,a) + \alpha(r + \gamma V(s') - Q(s,a)), \\ V(s') &= \max_{a' \in \mathcal{A}} Q(s', a'), and \end{aligned} \tag{7}$$

$$u(s) = \operatorname*{argmax}_{a' \in \mathcal{A}} Q(s,a), \tag{8}$$

where $s \in \mathcal{S}$ is a state variable, $a \in \mathcal{A}$ is a motor output, $r(s, a)$ is a reward, and s' is a state variable at the next time step. In these equations, α is the learning rate and γ is a discount factor. After an adequate Q is acquired,the agent can utilize an optimal policy, u, as in Eq. 8. Boltzmann selection is employed during the learning phase.

$$p(a|s) = exp(\beta Q(s,a)) / \sum_{a}^{\prime} (\beta Q(s,a')) \tag{9}$$

4.3 Switching architecture of internal goals

An agent can detect changes in the other agent's intentions by distinguishing between situations he/she faces. However, the goals themselves cannot be estimated even if switching between several goals can be detected. Here, we describe a learning method, which enables an agent to estimate the another's intentions implicitly. The method requires three assumptions to be made.

A1 Physical environmental dynamics f do not change.

A2 Every internal goal is equally difficult to achieve.

A3 The leader agent always selects each optimal policy for each intention.

The mathematical explanation for these assumptions will be described in the next section. We employes Boltzmann selection for internal goal switch. The rule to select the internal goals are described as

$$p(m|j) = \exp(Bw_{jm}) / \sum_{i=1}^{q} \exp(Bw_{ji}), \tag{10}$$

where $p(m|j)$ is the probability that G^m will be selected under situation, F^j, and B is the inverse temperature. The network connection, w_{jm}, between the current situation, F^j, and the current internal goal, G^m, is modified by the sum of the obtained reward, R_t^{jm}, during a certain period during the t-th trial, i.e.,

$$w_{jm} = \nu R_t^{jm} + (1-\nu) w_{jm} \tag{11}$$

Here, ν is the learning rate of the internal goal switching module. Eq. 11 shows that connection w_{jm} becomes strong if internal goal G^m is more easy to accomplish when the situation is F^j. Eq. 10 shows that an internal goal is more likely to be selected if its network connection is stronger than the other's. The abstract figure for the switching module is shown in Fig. 6. If the learning process for the switching architecture of internal goals is preceded and converged, a certain internal goal corresponding to a situation is selected.

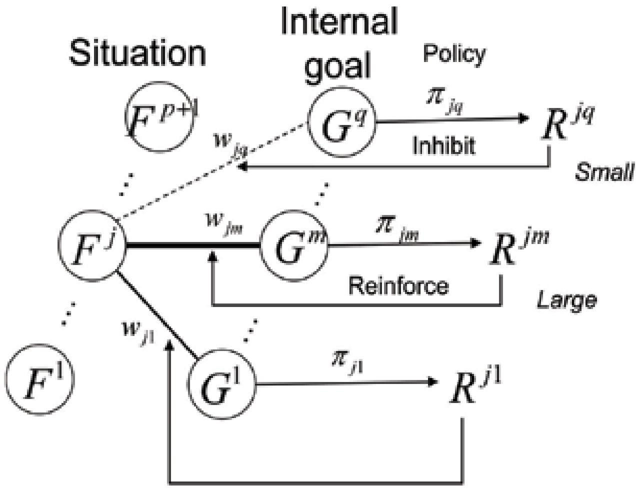


Fig. 6. Internal goal switching module

4.4 Mathematical basis for internal goal switching module

This section provides the mathematical basis for the learning rule ofr the implicit communication. First, the Bellman equation for the i-th (i = 1, 2) agent of a system involving two agents are described as[1].

$$V_i^\lambda(x)|_{u_j} = \max_{u_i \in U_i} \sum_{x'} P(x'|x, u_i, u_j) \times [G^\lambda(x,x') + \gamma V_i^\lambda(x')], \tag{12}$$

where G^λ is a reward function for the λ-th goal, u_i is the i-th agent's motor output, and x' is the x in the next step. G^λ in this framework is not assumed to have motor outputs as variables of the function. The optimal value function for the i-th agent depends on the other agent's policy, u_j . Here, we define u_i^λ as the i-th agent's policy that maximizes the j-th agent's maximized value function whose goal is G^λ.

$$u_i^\lambda = \operatorname*{argmax}_{u_i \in U_i} \max_{u_j \in U_j} \sum_{x'} P(x'|x, u_i, u_j) \times [G^\lambda(x,x') + \gamma V_i^\lambda(x')] and \tag{13}$$

$$V_i^{\lambda|\nu} \equiv V_i^\lambda|_{u_j^\nu}. \tag{14}$$

[1] In this section, we have assumed $i \neq j$ without making any remarks.

The assumptions, A2 and A3, we made in the previous section can be translated into the following,

A′2 : We assumed the *j*-th agent would use the controller, u_j^λ, and

A′3 : $V_i^{\lambda|\lambda}(x_0) = V_i^{\nu|\nu}(x_0)$,

where x_0 is the initial point of the task. The following relationship can easily be derived from the definition.

$$V_i^{\lambda|\lambda} = V_i^{\nu|\nu} \geq V_i^{\nu|\lambda}. \tag{15}$$

Therefore, the *i*-th agent's internal goal becomes the same as *j*-th agent's goal, if the *i*-th agent select a reward function that maximizes the value function under the condition that the *j*-th agent uses controller u_j^λ. When the initial point is not fixed, $V_i(x_0)$ is substituted by the averaged cumulative sum of rewards the $i-th$ agent obtains, who starts the task around the initial point, x_0. This leads us to the algorithm eq.11.

5. Experiment

We evaluate SSRL in this section. To fulfill all the assumptions made in Section 4 completely is difficult in a realistic task environment. The task described in this section roughly satisfies the assumptions, A′2 and A′3.

5.1 Conditions

We applied the proposed method to the truck-pushing task shown in Fig. 7. Two agents in the task environment, "Leader" and "Follower,"cooperatively push a truck to various locations. Both agents can adjust the truck's velocity and the angle of the handle. However, a single agent cannot achieve the task alone because its control force is limited. In addition, the Leader has all fixed policies for all sub-goals beforehand, and holds a stake in deciding the next goal. However, the agents cannot communicate with each other. Therefore, the agents cannot "explicitly" communicate their intentions. The Follower perceives situation F^j by using SSRL, changes its internal goal G^m based on the situation, and learns how to achieve the collaborative task. The two agents output the angle of the handle, θ_L, θ_F, and the wheel's rotating speed, ω_L, ω_F. Here，the final motor output to the truck, θ, ω, is defined as

$$\theta = K_\theta(\theta_L + \theta_F) and \tag{16}$$

$$\omega = K_\omega(\omega_L + \omega_F), \tag{17}$$

where K_θ and K_ω are the gain parameters of the truck. K_θ and K_ω were set to 0.5 in this experiment. The Leader's controller was designed to approximately satisfy the assumptions in Section 3. The controller in this experiment was a simple PD controller. The Follower's state, s, was defined as $s = [\rho, \alpha]$. The state space was digitized into 10 × 8 parts. The action space was defined as $\theta_F = \{-\pi/4, -\pi/8, 0, \pi/8, \pi/4\}$ and $\omega_F = \{0.0, 3.0\}$. As a result of the two agents' actions, the truck's angular velocity, Ω, was observed by the Follower agent. Ω, θ, and ω have a relationship of

$$\Omega \propto \omega \tan\theta. \tag{18}$$

The agents can carry the truck to a certain goal by cooperatively controlling Ω. The main state variables are shown in Fig. 8. Internal reward function G^m is defined as

$$G^m(x) = \begin{cases} 5 & \text{if } \|C - Goal_m\| < 1 \\ \kappa(1 - \|C - Goal_m\|) & \text{otherwise,} \end{cases} \tag{19}$$

where C is the position of the truck, and $Goal_m$ is the position of the m-th goal.

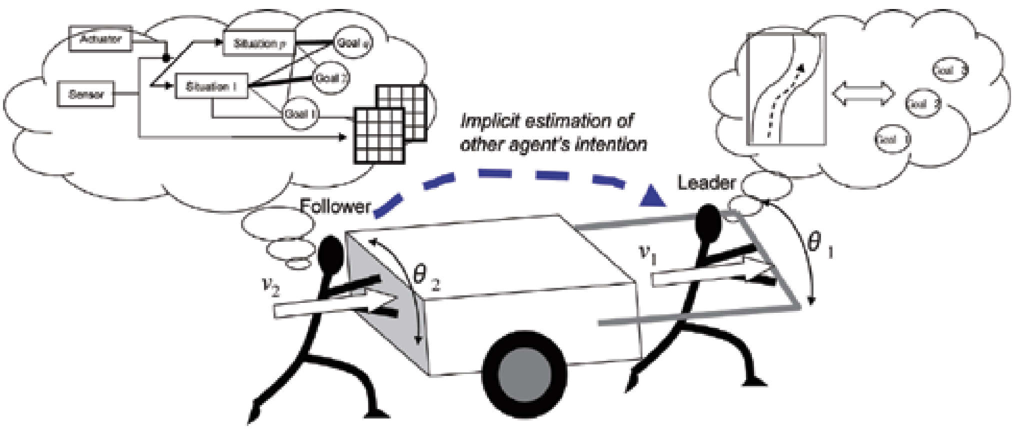


Fig. 7. Simple truck-pushing task by pair of agents

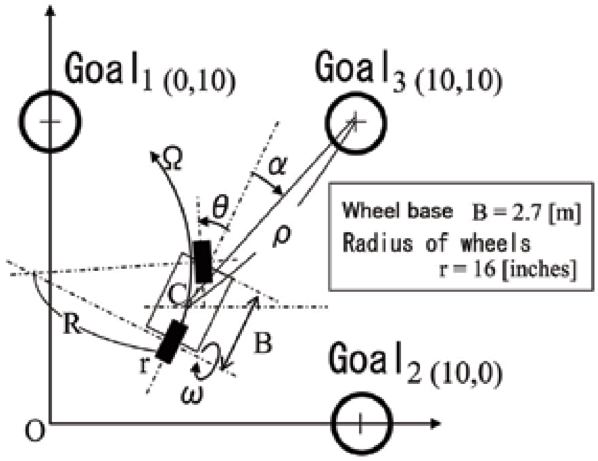


Fig. 8. State variables and parameters in task environment

5.2 Experiment 1: implicit estimation of another's intention

Wwe fisrt conducted an experiment in which the Follower estimated the Leader's goal, where the Leader selected one of three sub-goals, and learned how to achieve the collaborative task (Fig. 9, top). There were three goals, and the Leader changed its goals from $G_1 -> G_2 -> G_3$ alternately every 1000 trials.

In contrast to simple reinforcement learning, the Follower agent not only has to learn the policies for the goals but also the state for predictors the relationship between the current situation and the internal goal by updating these parameters.

The 1000 trajectories of the truck corresponding to all 1000 trials in this experiment are shown in Figs. 10 and 11. Simple Q-learning with explicitly given internal goals and SSRL are compared. Fig. 10 shows the results obtained from the experiment using Q-learning, and Fig. 11 shows those from the experiment using SSRL. The task success rate is indicated in each figure. The red curves represent the trajectories for the team that reached the goal, and the gray curves represent the trajectories for the team that did not reach the goal. This shows that simple Q-learning achieves a single task. However, the Follower could not coordinate with the Leader agent after it had changed its goal because it could not discover the Leader agent's intentions. SSRL performs better when the Leader changes its intentions. Fig.13 shows that three predictors were generated that discover the Leader's intentions. Furthermore, Fig. 12 shows that appropriate internal goals were selected inside the Follower agent.

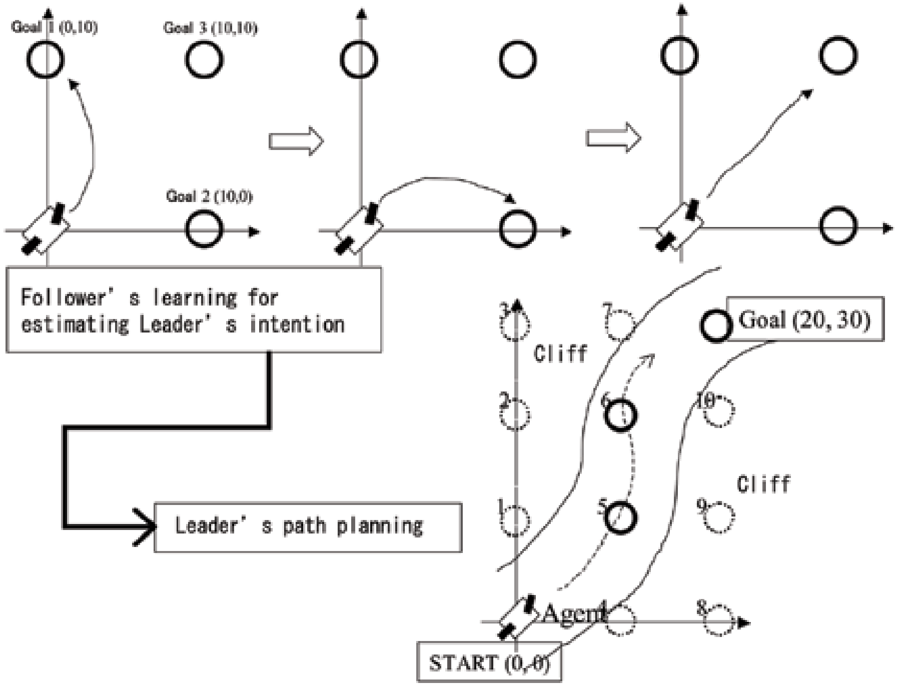


Fig. 9. Top: cooperative action is acquired by Follower, bottom: plan toward the goal is acquired by Leader

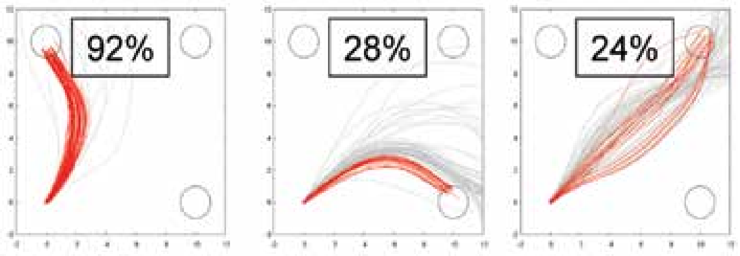


Fig. 10. Behaviors of truck at Follower's learning stage with single Q-table and internal goal-switching module without state predictors

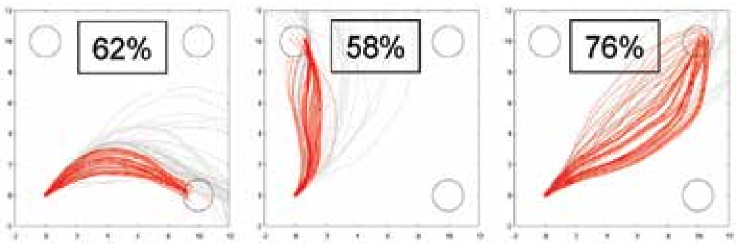


Fig. 11. Behaviors of truck at Follower's learning stage with SSRL

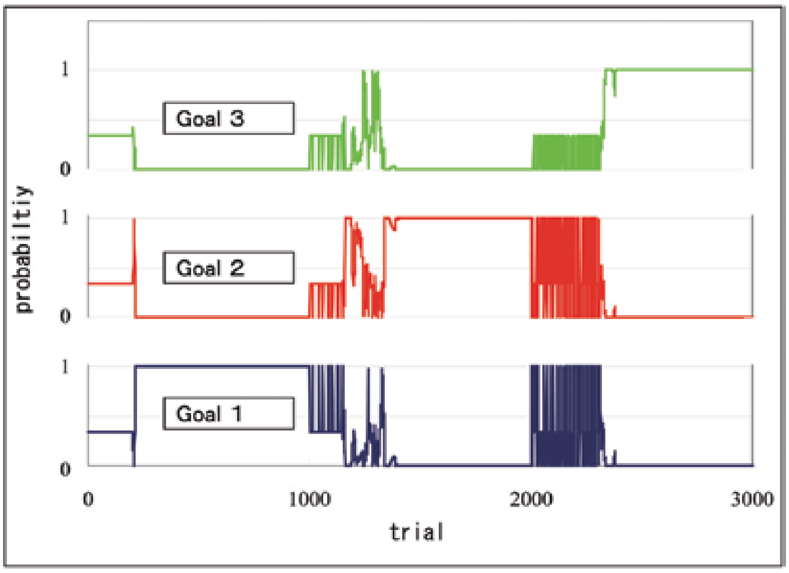


Fig. 12. Time course of probabilities where *m*-th internal goal is selected

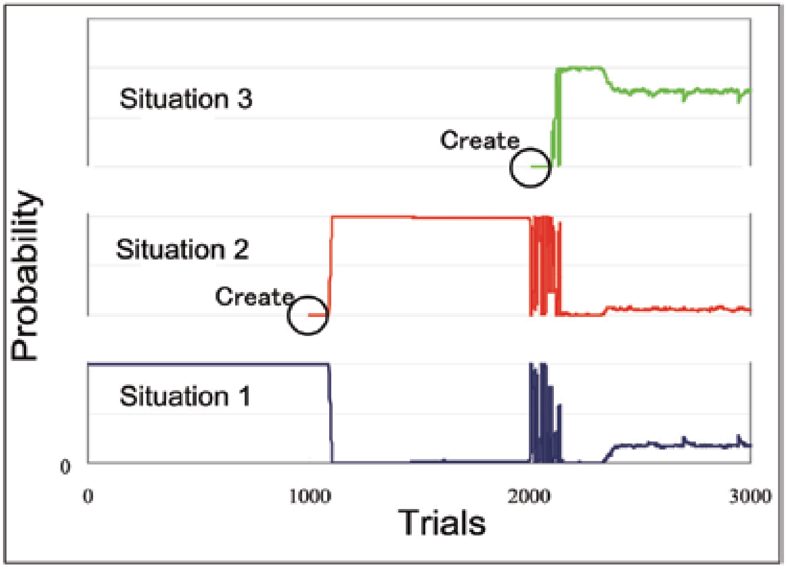


Fig. 13. Time course of probabilities that environment being faced is the *i*-th situation

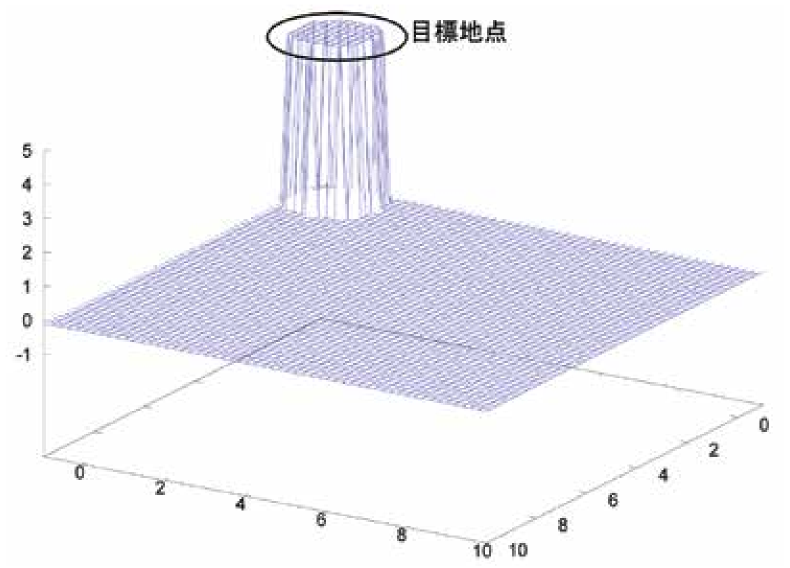


Fig. 14. Reward function for Follower's internal goals

These results show that SSRL enabled the Follower to implicitly estimate the Leader's intention.

5.3 Experiment 2: sequential collaborative task

After the follower had acquired the ability to implicitly estimate the leader's intentions, the next experiment was carried out. The experimental environment is shown at the bottom of Fig. 9. The task required the agents to go through several checkpoints (sub-goals), and reach the final goal. The Follower in the next experiment exploited the SSRL acquired through Experiment 1, and the Leader explored and planed the path to the final goal. The Leader agent can chose the next sub-goal out of three check points that correspond to three goals in Experiment 1, i.e., "up," "upper right," and "right," from the current checkpoint as shown in Fig. 9. There are also two "cliffs" in this task environment. If the truck enters the cliffs, it can no longer move. The Leader learned the path to the final goal by using a simple *Q*-learning. The reward function for the Leader is shown in Fig. 15. Two kinds of Follower agents are compared in this experiment. The first has a single Q-learning architecture and a perfect internal goal switch. The second has SSRL.

Fig. 16 shows the results for the experiment using simple Q-learning. Fig. 17 shows the results for the experiment using SSRL. Fig. 18 shows the success rate representing the probability that the team will finally reach the final goal. The results reveal that the team whose Follower agent could not discriminate the Leader's intentions performed worse than the team whose Follower agent could distinguish the Leader's intentions. Without such a distributed memory system like SSRL, the Follower would not be able to up with in the Leader's intentions. In addition to disadvantage, the poor performance of the Follower agent adversely affects the Leader's learning process. However, the Follower with SSRL could estimate the Leader's intentions and keep up with the Leader's plans although there was no explicit communication between the two agents.

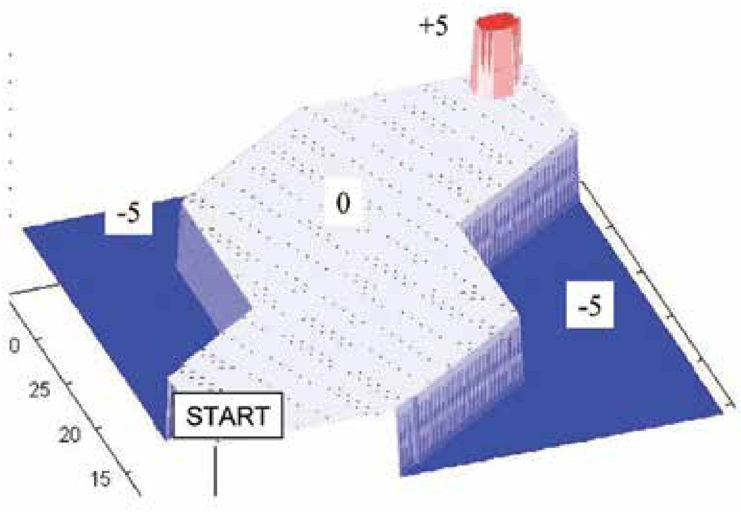


Fig. 15. Reward function for Leader agent for planning path

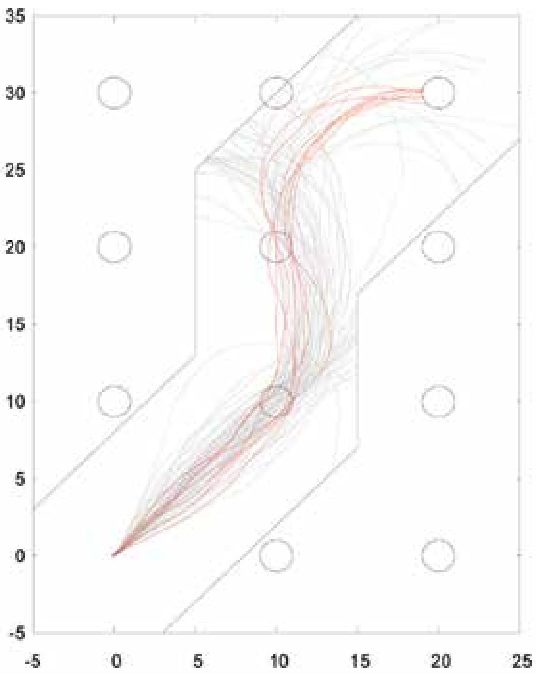


Fig. 16. Behaviors of truck at Leader's learning stage with single Q-table and internal goal switching module without Situation Recognizer

However, the success rate for the collaborative task saturated at about 40%. The reason for this is that the Follower notices changes in the Leader's intentions after these changes have sufficiently affected the state variables. The delay until the Follower becomes aware of the changes is sometimes critical, and the truck occasionally fell into the cliffs. To estimate the other's intentions without any explicit signs outside the state variables, the information has

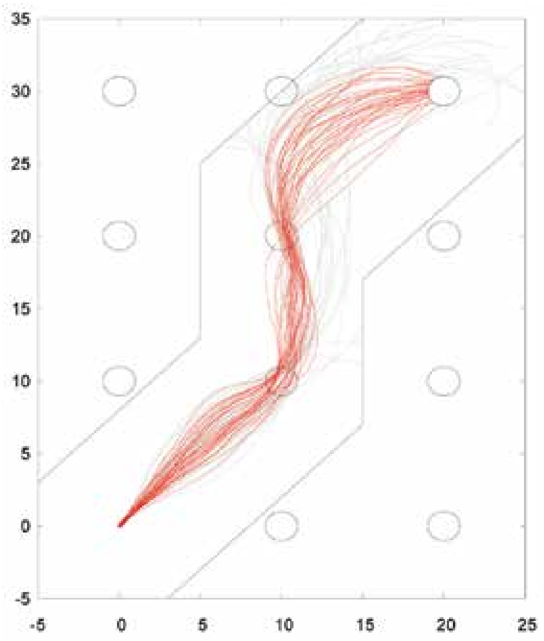


Fig. 17. Behaviors of truck at Leader's learning stage with SSRL

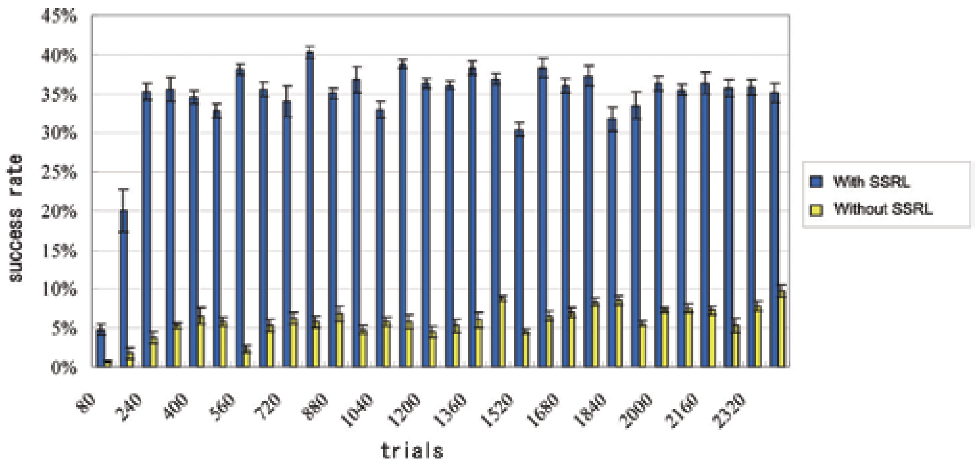


Fig. 18. Success rate for cooperative task

to be embedded in the state variables, which are the objectives of the team's control task. Our results suggest that it is not impossible to implicitly estimate the other's intentions, but it is important to have a communication channel whose variables are not related to the state variables, which are the objectives of the task, e.g., voice, colar sign, or marker. This must be the reason why we use explicit sign in collaborative tasks. As we mentioned, the "implicit estimation" must back up and complement "explicit estimation." "Explicit estimation" must be faster and better than "implicit estimation" as far as a code table was shared in a team. However, this does not mean "explicit estimation" is superior to"implicit estimation." They are complementary architectures.

6. Conclusion

We described a framework for implicitly estimating another's intentions based on modular reinforcement learning. We applied the framework to a truckpushing task by two agents as a concrete example. In the experiment, the Follower agent could perceive changes in the Leader's intentions and estimate his intentions without observing any explicit signs on any action outputs from the Leader. This demonstrated that autonomous agents can cooperatively achieve a task without any explicit communication. Self-enclosed autonomous agents can indirectly perceive the other's changes in intentions from changes in their surrounding environment. It is revealed that multiple internal models help an autonomous agent to achieve collaborative task.

In the context of artificial intelligence, "symbol grounding problem" is considered as an important problem. The problem deals with how robots and people can relate their symbolic system to their physical and embodied experiences. The symbolic system mentioned here is also used in communication, usually. Takamuku et al. presented a system for lexicon acquisition through behavior learning which is based on a modified multi-module reinforcement. The robot in their work is able to automatically associate words to objects with various visual features based on similarities in features of dynamics[8]. At the same time, Taniguchi et al. described an integrative learning architecture for spike timing-dependent plasticity (STDP) and the reinforcement learning schemata model (RLSM) [12, 11]. The learning architecture enables an autonomous robot to acquire behavioral concepts and signs representing the situation where the robot should initiate the behavior. They called this process "symbol emergence." The symbolic system plays a important role in human social communication.They also utilize modular learning architecture to describe the process of symbol organization. However, they treat bottomup organization of "explicit symbols," which is assumed to be used explicit communication.

In many researches, "symbolic communication" means exchanging discrete signals. However, the essential point of symbolic communication is not such an externalized signs, but an adaptive formation of "interpretant" from the viewpoint of Peirce's semiotics. Therefore, we focus on the implicit communication and its bottom-up process of organization.

However, the system we treated in this chapter is constrained to some extent. This framework for implicit estimates does not always work well. If the system does not satisfy the assumptions made in Section 4, the framework is not guaranteed to work. The Leader's policies are fixed when the Follower agent is learning its policies, predictors, and network connections in our framework. The model described in this chapter may not work in the simultaneous multi-agent reinforcement learning environment. We intend to take these into account in future work.

7. References

[1] K. Doya, N. Sugimoto, D. Wolpert, and M. Kawato. Selecting optimal behaviors based on context. *International symposium on emergent mechanisms of communication*, 2003.

[2] Charles Hartshorne, Paul Weiss, and Arthur W. Burks, editors. *Collected Papers of Charles Sanders Peirce*. Thoemmes Pr, 4 1997.

[3] M. Haruno, D.M.Wolpert, and M. Kawato. Mosaic model for sensorimotor learning and control. *Neural Computation*, 13:2201–2220, 2001.

[4] K. Murphy. Learning switching kalman-filter models. *Compaq Cambridge Research Lab Tech Report*, pages 98–10, 1998.

[5] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal,*, 27:379–423, and 623–656, 1948.

[6] R. Sutton and A.G. Barto. *Reinforcement Learning : An Introduction*. The MIT Press, 1998.

[7] Y. Takahashi et al. Modular learning syatem and scheduling for behavior acquisition in multi-agent environment. In *RoboCup 2004 Symposium papers and team description papers, CD-ROM*, 2004.

[8] Shinya Takamuku, Yasutake Takahashi, and Minoru Asada. Lexicon acquisition based on object-oriented behavior learning. *Advanced Robotics*, 20(10):1127–1145, 2006.

[9] J. Tani, M. Ito, and Y. Sugita. Seif-organization of distributedly represented mulyiple behavior schemata in a mirror system:reviews of robots using rnnpb. *Neural Networks,*, 17:1273–1289, 2004.

[10] J. Tani and S. Nolfi. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12:1131–1141, 1999.

[11] T. Taniguchi and T. Sawaragi. Symbol emergence by combining a reinforcement learning schema model with asymmetric synaptic plasticity. In *5th International Conference on Development and Learning*, 2006.

[12] T. Taniguchi and T. Sawaragi. Incremental acquisition of behaviors and signs based on a reinforcement learning schemata model and a spike timing-dependent plasticity network. *Advanced Robotics*, 21(10):1177–1199, 2007.

[13] T. Taniguchi and T. Sawaragi. Incremental acquisition of multiple nonlinear forward models based on differentiation process of schema model. *Neural Networks*, 21(1):13–27, 2008.

[14] C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.

[15] D.M. Wolpert, K. Doya, and M. Kawato. A unifying comuputational framework for motor control and social interaction. *Phil Trans R Soc Lond B*, 358:593–602, 2003.

[16] D.M. Wolpert, Z. Ghahramani, and M. I. Jordan. An internal model for sensorimotor integration. *science*, 269:1880–1882, 1995.

[17] D.M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.

20

Machine Learning for Sequential Behavior Modeling and Prediction

Xin Xu
Institute of Automation, National University of Defense Technology, Changsha, 410073, China

1. Introduction

In the information era, as computer networks and related applications become more and more popular, security problems are more and more serious in global information infrastructure. It was reported that in the past two years, large amounts of network attacks and computer viruses caused great damages to global economy and the potential threats to the global information infrastructure have increased a lot. To defend various cyber attacks and computer viruses, lots of computer security techniques have been studied, which include cryptography, firewalls and intrusion detection, etc. As an important computer security technique, intrusion detection [1,2] has been considered to be more promising for defending complex computer attacks than other techniques such as cryptography, firewalls, etc. The aim of intrusion detection is to find cyber attacks or non-permitted deviations of the characteristic properties in a computer system or monitored networks. Thus, one of the central problems for intrusion detection systems (IDSs) is to build effective behavior models or patterns to distinguish normal behaviors from abnormal behaviors by observing collected audit data. To solve this problem, earlier IDSs usually rely on security experts to analyze the audit data and construct intrusion detection rules manually [2]. However, since the amount of audit data, including network data, process execution traces and user command data, etc., increases vary fast, it becomes a time-consuming, tedious and even impossible work for human experts to analyze dynamic, huge volumes of audit data and extract attack signatures or detection rules. Furthermore, detection rules constructed by human experts are usually based on fixed features or signatures of existing attacks, so it will be very difficult for these rules to detect deformed or even completely new attacks.
According to the differences in the monitored data, IDSs can be mainly classified into two categories, i.e., network-based intrusion detection and host-based intrusion detection. Network-based intrusion detection observes data from network packets and extracts various features from them, which usually include connection features, traffic features, and content features. A systematic discussion on feature representation in network-based intrusion detection can be found in [3]. For host-based intrusion detection, various observation data from the corresponding operation systems are collected, which mainly include system call data and shell command data [4], etc. Despite of having different observation data, both host-based and network-based intrusion detection need to improve the detection accuracy for large volumes and variability of normal and attack behaviors. Aiming at this problem,

lots of research work has been devoted to develop intrusion detection systems (IDSs) using various artificial intelligence (AI) methods and tools [3-5]. Thus, the motivations for applying AI techniques in IDSs are due to large amounts of dynamic behaviors and the lack of *a priori* knowledge for unknown attacks. How to establish appropriate behavior models has been a central problem in the development of IDSs since the distinctions between normal behaviors and computer attacks are usually very vague. In earlier research on IDSs, it was very popular to separately construct behavior models either for normal usages or attacks. To model intrusion behaviors alone is called misuse detection and anomaly detection refers to establish profiles of normal usages. In misuse detection, behavior patterns or models of known attacks are constructed and alarms are raised when the patterns of observation data match the attack models. On the other hand, anomaly detection only models the patterns of normal behaviors and detects any possible attacks as deviations from the normal behavior model. Until now, although there have been many advances in misuse detection and anomaly detection, some significant challenges still exist to meet the requirements of defending computer systems from attacks with increasing complexity, intelligence, and variability. For misuse detection, the inability of detecting new attacks is its inevitable weakness and it is very hard to improve the performance of pure misuse detection systems for the sake of increasing amounts of novel attacks. Although anomaly detection has the ability of detecting new attacks, it usually suffers from high rates of false alarms since it is very difficult to obtain a complete model of normal behaviors.

To solve the above problems in IDSs, machine learning and data mining methods for intrusion detection have received a lot of research interests in recent years [4-10]. One motivation for applying machine learning and data mining techniques in IDSs is to construct and optimize detection models automatically, which will eliminate the tedious work of human experts for data analysis and model building in earlier IDSs. To detect novel attacks, several adaptive anomaly detection methods were proposed by employing data mining methods based on statistics [7], or clustering techniques [10]. Recently, there have been several efforts in designing anomaly detection algorithms using supervised learning algorithms, such as neural networks [8], support vector machines [11], etc. In addition to supervised or inductive learning methods for misuse and anomaly detection, another approach to adaptive intrusion detection is to use unsupervised learning methods. Unlike supervised learning methods, where detection models are constructed by careful labeling of normal behaviors, unsupervised anomaly detection tries to detect anomalous behaviors with very little a priori knowledge about the training data. However, as studied in [12], the performance of pure unsupervised anomaly detection approaches is usually unsatisfactory, e.g., it was demonstrated in [12] that supervised learning methods significantly outperform the unsupervised ones if the test data contains no unknown attacks.

Despite of many advances that have been achieved, existing IDSs still have some difficulties in improving their performance to meet the needs of detecting increasing types of attacks in high-speed networks. One difficulty is to improve detection abilities for complex or new attacks without increasing false alarms. Since misuse IDSs employ signatures of known attacks, it is hard for them to detect deformed attacks, notwithstanding completely new attacks. On the other hand, although anomaly detection can detect new types of attacks by constructing a model of normal behaviors, the false alarm rates in anomaly-based IDSs are usually high. How to increase the detecting ability while maintaining low false alarms is still an open problem of IDS research.

In addition to the ability of realizing automatic model construction for misuse detection and anomaly detection, another promising application of machine learning methods in intrusion detection is to build dynamic behavior modeling frameworks which can combine the advantages of misuse detection and anomaly detection while eliminate the weakness of both. Many previous results on misuse detection and anomaly detection were usually based on static behavior modeling, i.e., normal behaviors or attack behaviors were modeled as static feature patterns and the intrusion detection problem was transformed to a pattern matching or classification procedure. However, dynamic behavior modeling is different from static behavior modeling approaches in two aspects. One aspect is that the relationships between temporal features are explicitly modeled in dynamic modeling approaches while static modeling only considers time independent features. The other aspect is that probabilistic frameworks are usually employed in dynamic behavior models while most static models make use of deterministic decision functions. Furthermore, many complex attacks are composed of multiple stages of behaviors, for example, a remote-to-local (R2L) attack commonly performs probe attacks to find target computers with vulnerabilities at first, and later realizes various buffer overflow attacks by utilizing the vulnerabilities in the target host computers. Therefore, sequential modeling approaches will be more beneficial to precisely describe the properties of complex multi-stage attacks. In [4], dynamic behavior modeling and static behavior modeling approaches were discussed and compared in detail, where a Hidden Markov Model was proposed to establish dynamic behavior models of audit data in host computers including system call data and shell command data. It was demonstrated in [4] that dynamic behavior modeling is more suitable for sequential data patterns such as system call data of host computers. However, the main difficulty for applying HMMs in real-time IDS applications is that the computational costs of HMM training and testing increase very fast with the number of states and the length of observation traces.

In this Chapter, some recently developed machine learning techniques for sequential behavior modeling and prediction are studied, where adaptive intrusion detection in computer systems is used as the application case. At first, a general framework for applying machine learning to computer intrusion detection is analyzed. Then, reinforcement learning algorithms based on Markov reward models as well as previous approaches using Hidden Markov Models (HMMs) are studied for sequential behavior modeling and prediction in adaptive intrusion detection. At last, the performance of different methods are evaluated and compared.

2. A general framework of ML applications in intrusion detection

In [9], based on a comprehensive analysis for the current research challenges in intrusion detection, a framework for adaptive intrusion detection using machine learning techniques was presented, which is shown in Fig.1. The framework is composed of three main parts. The first one is for data acquisition and feature extraction. Data acquisition is realized by a data sensing module that observes network flow data or process execution trajectories from network or host computers. After pre-processing of the raw data, a feature extraction module is used to convert the raw data into feature vectors that can be processed by machine learning algorithms and an extraction model based on unsupervised learning can be employed to extract more useful features or reduce the dimensionality of feature vectors. This process for automated feature extraction is a component of the machine learning part in

the framework. In the machine learning part, audit data for training are stored in databases and they can be dynamically updated by human analysts or by machine learning algorithms. The third part in the framework depicted in Fig.1 is for real-time detection, which is to make use of the detection models as well as the extracted feature vectors to determine whether an observed pattern or a sequence of patterns is normal or abnormal.
To automatically construct detection models from the audit data, various machine learning methods can be applied, which include unsupervised learning, supervised learning and reinforcement learning. In addition, there are three perspectives of research challenges for intrusion detection, which include feature extraction, classifier construction and sequential behavior prediction. Although various hybrid approaches may be employed, it was illustrated that these three perspectives of research challenges are mainly suitable for machine learning methods using unsupervised, supervised and reinforcement learning algorithms, respectively. In contrast, in the previous adaptive IDS framework in [13], feature selection and classifier construction of IDSs were mainly tackled by traditional association data mining methods such as the *Apriori* algorithm.

2.1 Feature extraction

As illustrated in Fig.1, feature extraction is the basis for high-performance intrusion detection using data mining since the detection models have to be optimized based on the selection of feature spaces. If the features are improperly selected, the ultimate performance of detection models will be influenced a lot. This problem has been studied during the early work of W.K. Lee and his research results lead to the benchmark dataset KDD99 [13-14], where a 41-dimensional feature vector was constructed for each network connection. The feature extraction method in KDD99 made use of various data mining techniques to identify some of the important features for detecting anomalous connections. The features employed in KDD99 can serve as the basis of further feature extraction.
In KDD99, there are 494,021 records in the 10% training data set and the number of records in the testing data set is about five million, with a 10 percent testing subset of 311028 records. The data set contains a total of 22 different attack types. There are 41 features for each connection record that have either discrete values or continuous values. The 41-dimensional feature can be divided into three groups. The first group of features is called basic or intrinsic features of a network connection, which include the duration, prototype, service, number of bytes from source IP addresses or from destination IP addresses, and some flags in TCP connections. The second group of features in KDD99 is composed of the content features of network connections and the third group is composed of the statistical features that are computed either by a time window or a window of certain kind of connections.
The feature extraction method in the KDD99 dataset has been widely used as a standard feature construction method for network-based intrusion detection. However, in the later work of other researchers, it was found that the 41-dimensional features are not the best ones for intrusion detection and the performance of IDSs may be further improved by studying new feature extraction or dimension reduction methods [11]. In [11], a dimension reduction method based on principal component analysis (PCA) was developed so that the classification speed of IDSs can be improved a lot without much loss of detection precision.

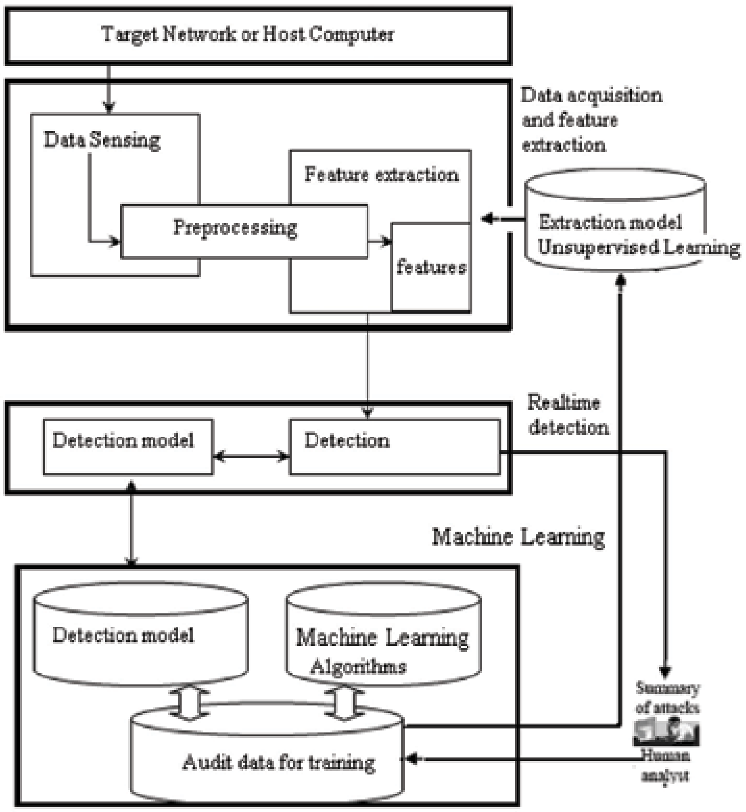


Fig. 1. A framework for adaptive IDSs based on machine learning

2.2 Classifier construction

After performing feature extraction of network flow data, every network connection record can be denoted by a numerical feature vector and a class label can be assigned to the record, i.e.,

$$\{(\vec{x}_i, y_i)\}, \quad i = 1,2,...,N \quad y_i \in \{1,2,...,m\}$$

For the extracted features of audit data such as KDD99, when labels were assigned to each data record, the classifier construction problem can be solved by applying various supervised learning algorithms such as neural networks, decision trees, etc. However, the classification precision of most existing methods needs to be improved further since it is very difficult to detect lots of new attacks by only training on limited audit data. Using anomaly detection strategy can detect novel attacks but the false alarm rate is usually very high since to model normal patterns very well is also hard. Thus, the classifier construction in IDSs remains another technical challenge for intrusion detection based on machine learning.

2.3 Sequential behavior prediction

As discussed above, host-based IDSs are different from network-based IDSs in that the observed trajectories of processes or user shell commands in a host computer are sequential patterns. For example, if we use system call traces as audit data, a trajectory of system calls can be modeled as a state transition sequence of short sequences. In the following Fig. 2, it is shown that every state is a short sequence of length 3 and different system call traces can form different state transitions, where *a*, *b*, and *c* are symbols for system calls in a host computer.

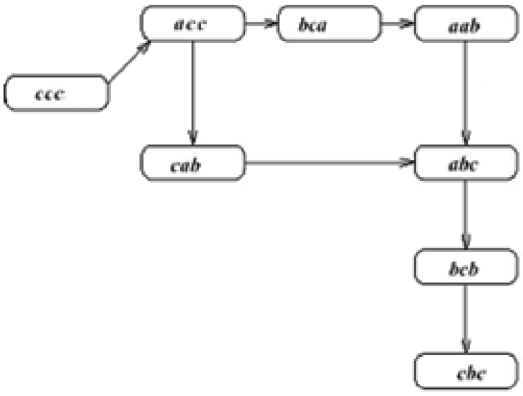


Fig. 2. A sequential state transition model for host-based IDSs

Therefore, the host-based intrusion detection problem can be considered as a sequential prediction problem since it is hard to determine a single short sequence of system calls to be normal and normal and there are intrinsic temporal relationships between sequences. Although we can still transform the above problem to a static classification problem by mapping the whole trace of a process to a feature vector [15], it has been shown that dynamic behavior modeling methods, such as Hidden Markov Models (HMMs) [4], are more suitable for this kind of intrusion detection problem. In the following, a host-based intrusion detection method will be studied based on reinforcement learning, where a Markov reward model is established for sequential pattern prediction and temporal difference (TD) algorithms [16] are used to realize high-precision prediction without many computational costs. At first, the popular HMMs for sequential behavior modeling will be introduced in the next section.

3. Hidden Markov Models (HMMs) for sequential behavior modeling

Due to the large volumes of audit data, to establish and modify detection models manually by human experts becomes more and more impractical. Therefore, machine learning and data mining methods have been widely considered as important techniques for adaptive intrusion detection, i.e., to construct and optimize detection models automatically. Previous work using supervised learning mainly focused on static behavior modeling methods based on pre-processed training data with class labels. However, training data labeling is one of the most important and difficult tasks since it is hard to extract signatures precisely even for known attacks and there are still increasing amounts of unknown attacks. In most of the previous works using static behavior modeling and supervised learning algorithms, every single sample of the training data was either labeled as normal or abnormal. However, the distinctions between normal and abnormal behaviors are usually very vague and improper labeling may limit or worsen the detection performance of supervised learning methods.

More importantly, for complex multi-stage attacks, it is very difficult or even impossible for static behavior models based on supervised learning to describe precisely the temporal relationships between sequential patterns. The above problems become the main reasons leading to the unsatisfactory performance of previous supervised learning approaches to adaptive IDSs, especially for complex sequential data. The recent works on applying HMMs [4] and other sequence learning methods [17] have been focused on dynamic behavior modeling for IDSs, which tried to explicitly estimate the probabilistic transition model of sequential patterns. For the purpose of comparisons, in the following, a brief introduction on HMM-based methods for intrusion detection will be given.
As a popular sequential modeling approach, HMMs have been widely studied and applied in lots of areas such as speech recognition [18], protein structure prediction, etc. A discrete state, discrete time, first order hidden Markov model describes a stochastic, memory-less process. A full HMM can be specified as a tuple: $\lambda = (N, M, A, B, \pi)$, where N is the number of states, M is the number of observable symbols, A is the state transition probability matrix which satisfies the Markov property:

$$a_{ij} = P(q_{t+1} = j|q_t = i) = P(q_{t+1} = j|q_t = i, q_{t-1}, \dots, q_0) \tag{1}$$

B is the observation probability distribution

$$b_j(k) = P(o_t = k|q_t = j)\ , i \leqslant k \leqslant M \tag{2}$$

and π is the initial state distribution. The initial state distribution π satisfies:

$$\pi_i = P(x_0 = i) \tag{3}$$

$$0 \le \pi_i \le 1 \tag{4}$$

$$\sum_{i=1}^{N} \pi_i = 1 \tag{5}$$

For discrete state HMMs, we can let $Q = \{q_1, q_2, \dots, q_M\}$ denote the set of all states, $O = \{O_1, O_2, \dots, O_N\}$ denote the set of all observation symbols. A typical trace of HMMs is shown in the following Fig.3, where O_i (i=1,2,…,T) are observation symbols and q_i (i=1,2,…,T) are the corresponding states.

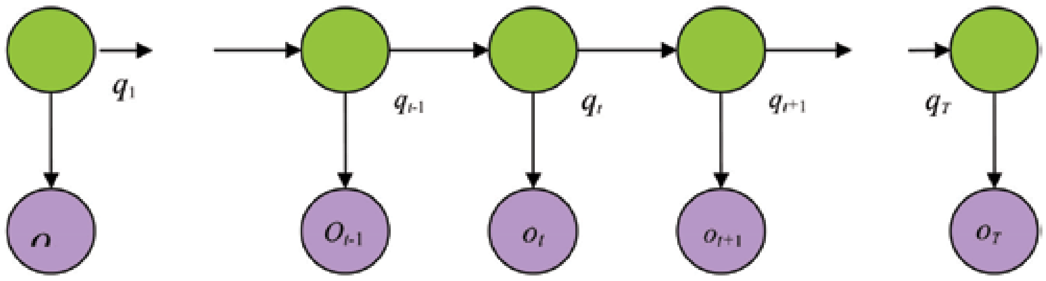


Fig. 3. An HMM model

In practice, there might be *a priori* reasons to assign certain values to each of the initial state probabilities. For example, in some applications, one typically expects HMMs to start in a particular state. Thus, one can assign probability one to that state and zero to others.

For HMMs, there are two important algorithms to compute the data likelihood when the model of an HMM is given. One algorithm is the Forward-Backward algorithm which calculates the incomplete data likelihood and the other is the Viterbi algorithm which calculates the complete data likelihood. Implicitly, both Forward-Backward and Viterbi find the most likely sequence of states, although differently defined. For detailed discussion on the two algorithms, please refer to [8].

Another important problem in HMMs is the model learning problem which is to estimate the model parameters when the model is unknown and only observation data can be obtained. The model learning problem is essential for HMMs to be applied in intrusion detection since a detection model must be constructed only by training data samples. For model learning in HMMs, the Expectation-Maximization (EM) algorithm is the most popular one which finds maximum a posteriori or maximum likelihood parameter estimate from incomplete data. The Baum-Welch algorithm is a particular form of EM for maximum likelihood parameter estimation in HMMs. For a detailed discussion on HMMs, the readers may refer to [18].

In intrusion detection based on HMMs, the Baum-Welch algorithm can be used to establish dynamic behavior models of normal data and after the learning process is completed, attack behaviors can be identified as deviations from the normal behavior models.

4. Reinforcement learning for sequential behavior prediction

4.1 Intrusion detection using Markov reward model and temporal-difference learning

In HMM-based dynamic behavior modeling for intrusion detection, the probabilistic transition model of the IDS problem is explicitly estimated, which is computationally expensive when the number of states and the length of traces increase. In this Section, an alternative approach to adaptive intrusion detection will be presented. In the alternative approach, Markov state transition models are also employed but have an additional evaluative reward function, which is used to indicate the possibility of anomaly. Therefore, the intrusion detection problem can be tackled by learning prediction of value functions of a Markov reward process, which have been widely studied in the reinforcement learning community. To explain the principle of the RL-based approach to intrusion detection, the sequential behavior modeling problem in host-based IDSs using sequences of system calls is discussed in the following.

For host-based intrusion detection, the audit data are usually obtained by collecting the execution trajectories of processes or user commands in a host computer. As discussed in [19], host-based IDSs can be realized by observing sequences of system calls, which are related to the operating systems in the host computer. The execution trajectories of different processes form different traces of system calls. Each trace is defined as the list of system calls issued by a single process from the beginning of its execution to the end. If a state at a time step is defined as m successive system calls and a sliding window with length l is defined, the traces of system calls can be transformed to a state transition sequences and different traces correspond to different state transition sequences. For example, if we select a sequence of 4 system calls as one state and the sliding length between sequences is 1, the state transitions corresponding to a short trace *tr*={ open, read, mmap, mmap, open, read, mmap} are:

State $S1$: *open, read, mmap, mmap*

State $S2$: *read, mmap, mmap, open*

State S3: mmap, mmap, open, read

State $S4$: *mmap, open, read, mmap*

Then the state transition sequence of the above trace *tr* is:

$$S1 \to S2 \to S3 \to S4$$

As studied and verified in [4], dynamic behavior models for sequential pattern prediction are superior to static models when temporal relationships between feature patterns need to be described accurately. Different from the previous work in [4], where an HMM-based dynamic behavior modeling approach was studied, the following dynamic behavior modeling method for intrusion detection is based on learning prediction using Markov reward models. The method is focused on a learning prediction approach, which has been popularly studied in RL research [21-22], by introducing a Markov reward model of the IDS problem so that high accuracy and low computational costs can both be guaranteed [20]. Firstly, the Markov reward model for the IDS problem is introduced as follows.

Markov reward processes are popular stochastic models for sequential modeling and decision making. A Markov reward process can be denoted as a tuple $\{S, R, P\}$, where S is the state space, R is the reward function, P is the state transition probability. Let $\{x_t \mid t=0,1,2,\ldots;\ x_t \in S\}$ denote a trajectory generated by a Markov reward process. For each state transition from x_t to x_{t+1}, a scalar reward r_t is defined. The state transition probabilities satisfy the following Markov property:

$$P\{x_{t+1} | x_t, x_{t-1}, \ldots, x_1, x_0\} = P\{x_{t+1} | x_t\} \tag{6}$$

The reward function of the Markov reward plays an important role for dynamic behavior modeling in intrusion detection problems. As described in the following Fig.2, in a Markov reward model for intrusion detection based on system calls, each state is defined as a short sequence of successive system calls and after each state transition, a scalar reward r_t is given to indicate whether there is a possibility to be normal or attack behaviors. The design of the reward function can make use of available *a priori* information so that the anomaly probability of a whole state trajectory can be estimated based on the accumulated reward function. In one extreme case, we can indicate every state to be normal or abnormal with high confidence and the immediate reward of each state is designed as

$$r_t = \begin{cases} -1, & \textit{for normal state} \\ 1, & \textit{for abnormal state} \end{cases}, \qquad \text{for } t=1,2,\ldots T \tag{7}$$

The above extreme case is identical to transform the dynamic behavior modeling problem to a static pattern classification problem since we have class labels for every possible states, where the reward becomes a class label for every state. However, in fact, due to the sequential properties of system call data and the vague distinctions between normal traces and abnormal traces, it is usually not appropriate or even impossible to tell whether an intermediate state to be normal or abnormal definitely. Moreover, even if it is reasonable to assign precise class labels to every states, it is also very hard to obtain precise class labels for large amounts of audit data. Therefore, it is more reasonable to develop dynamic behavior modeling approaches which not only incorporate the temporal properties of state transitions but also need little *a priori* knowledge for class labeling. An extreme case toward this direction is to provide evaluative signals to a whole state transition trajectory, i.e., only a whole state trajectory is indicated to be normal or abnormal while the intermediate states are not definitely labeled. For example, in the following Fig.4, the reward at the terminal state r_T can be precisely given as:

$$r_T = \begin{cases} -1, & \text{for normal trace} \\ 1, & \text{for abnormal trace} \end{cases} \tag{8}$$

For intermediate states $s_1,\ldots, s_{T-1}$, a zero reward can be given to each state when there is no *a priori* knowledge about the anomaly of the states. However, in more general cases, the intermediate rewards can be designed based on available prior knowledge on some features or signatures of known attacks.

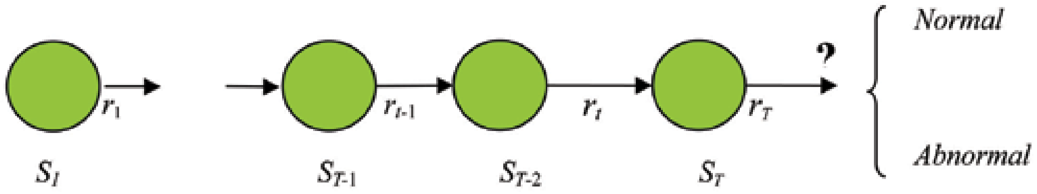


Fig. 4. A Markov reward process for intrusion detection

According to the above Markov reward process model, the detection of attack behaviors can be tackled by the sequential prediction of expected total rewards of a state in a trajectory since the reward signals, especially the terminal reward at the end of the trajectory provide information about whether the trajectory is normal or abnormal. Therefore, the intrusion detection problem becomes a value function prediction problem of a Markov reward process, which has been popularly studied by many researchers in the framework of reinforcement learning [21-24]. Among the learning prediction methods studied in RL, temporal difference learning (TD) is one of the most important one and in the following discussions, we will focus on the TD learning prediction algorithm for intrusion detection. Firstly, some basic definitions on value functions and dynamic programming are given as follows.

In order to predict the expected total rewards received after a state trajectory starting from a state x, the value function of state x is defined as follows:

$$V(x) = E\{\sum_{t=0}^{\infty} \gamma^t r_t | x_0 = x\} \tag{9}$$

where $x \in S$, $0 < \gamma \leq 1$ is the discount factor, r_t is the reward received after state transition $x_t \rightarrow x_{t+1}$ and $E\{.\}$ is the expectation over the state transition probabilities.

According to the theory of dynamic programming, the above value function satisfies the following Bellman equation.

$$V(s_t) = R_t + \gamma E[V(s_{t+1})] \tag{10}$$

where R_t is the expected reward received after state transition $x_t \rightarrow x_{t+1}$.
The aim of RL is to approximate the optimal or near-optimal policies from its experiences without knowing the parameters of this process. To estimate the optimal policy of an MDP, RL algorithms usually predict the value functions by observing data from state transitions and rewards. Thus, value function prediction of Markov reward models becomes a central problem in RL since optimal policies or optimal value functions can be obtained based on the estimation of value functions. However, in RL, learning prediction is more difficult that in supervised learning. As pointed out by Sutton [22], the prediction problems in supervised learning are single-step prediction problems while learning prediction in reinforcement learning belongs to multi-step prediction, which is to predict outcomes that depend on a future sequence of decisions.
Until now, temporal difference learning or TD learning has been considered as one of the most efficient approaches to value function prediction without any *a priori* model information about Markov reward processes. Different from supervised learning for sequential prediction such as Monte Carlo estimation methods, TD learning is to update the estimations based on the differences between two temporally successive estimations, which constitutes the main ideas of a popular class of TD learning algorithms called TD(λ) [22]. In TD(λ), there are two basic mechanisms which are the temporal difference and the eligibility trace, respectively. Temporal differences are defined as the differences between two successive estimations and have the following form

$$\delta_t = r_t + \gamma \tilde{V}_t(x_{t+1}) - \tilde{V}_t(x_t) \tag{11}$$

where x_{t+1} is the successive state of x_t, $\tilde{V}$ (x) denotes the estimate of value function $V(x)$ and r_t is the reward received after the state transition from x_t to x_{t+1}.
As discussed in [22], the eligibility trace can be viewed as an algebraic trick to improve learning efficiency without recording all the data of a multi-step prediction process. This trick is originated from the idea of using a truncated reward sum of Markov reward processes. In TD learning with eligibility traces, an n-step truncated return is defined as

$$R_t^n = r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n \tilde{V}_t(s_{t+n}) \tag{12}$$

For an absorbing Markov reward process whose length is T, the weighted average of truncated returns is

$$R_t^{\lambda} = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^n + \lambda^{T-t-1} R_T \tag{13}$$

where $0 \leq \lambda \leq 1$ is a decaying factor and

$$R_T = r_t + \gamma r_{t+1} + \ldots + \gamma^T r_T \tag{14}$$

R_T is the Monte-Carlo return at the terminal state. In each step of TD(λ), the update rule of value function estimation is determined by the weighted average of truncated returns defined above, i.e.,

$$\Delta\tilde{V}_t(s_i) = \alpha_t(R_t^\lambda - \tilde{V}_t(s_i)) \tag{15}$$

where α_t is a learning factor.
The update equation (25) can be used only after the whole trajectory of the Markov reward process is observed. To realize incremental or online learning, eligibility traces are defined for each state as follows:

$$z_{t+1}(s_i) = \begin{cases} \gamma\lambda\, z_t(s_i) + 1, & \text{if } s_i = s_t \\ \gamma\lambda\, z_t(s_i), & \text{if } s_i \neq s_t \end{cases} \tag{16}$$

The online TD(λ) update rule with eligibility traces is

$$\tilde{V}_{t+1}(s_i) = \tilde{V}_t(s_i) + \alpha_t \delta_t z_{t+1}(s_i) \tag{17}$$

where δ_t is the temporal difference at time step t, which is defined in (21) and $z_0(s)$=0 for all s.
Based on the above TD learning prediction principle, the intrusion detection problem can be solved by a model learning process and an online detection process. In the model learning process, the value functions are estimated based on the online TD(λ) update rules and in the detection process, the estimated value functions are used to determine whether a sequence of states belongs to a normal trajectory or an abnormal trajectory. For the reward function defined in (18), when an appropriate threshold μ is selected, the detection rules of the IDS can be designed as follows:

If $V(x) > \mu$, then raise alarms for attacks,

Else there are no alarms.

Since the state space of a Markov reward process is usually large or infinite in practice, function approximators such as neural networks are commonly used to approximate the value function. Among the existing TD learning prediction methods, TD(λ) algorithms with linear function approximators are the most popular and well-studied ones, which can be called linear TD(λ) algorithms.
In linear TD(λ), consider a general linear function approximator with a fixed basis function vector

$$\phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_n(x))^T \tag{18}$$

The estimated value function can be denoted as

$$\tilde{V}_t(x) = \phi^T(x) W_t \tag{19}$$

where $W_t = (w_1, w_2, \ldots, w_n)^T$ is the weight vector.
The corresponding incremental weight update rule is

$$W_{t+1} = W_t + \alpha_t (r_t + \gamma \phi^T (x_{t+1}) W_t - \phi^T (x_t) W_t) \vec{z}_{t+1} \tag{20}$$

where the eligibility trace vector $\vec{z}_t(s) = (z_{1t}(s), z_{2t}(s), \ldots, z_{mt}(s))^T$ is defined as

$$\vec{z}_{t+1} = \gamma \lambda \vec{z}_t + \phi(x_t) \tag{21}$$

In [19], the above linear TD(λ) algorithm is proved to converge with probability 1 under certain assumptions and the limit of convergence W^* is also derived, which satisfies the following equation

$$E_0[A(X_t)]W^* - E_0[b(X_t)] = 0 \tag{22}$$

where $X_t = (x_t, x_{t+1}, z_{t+1})$ (t=1,2,…) form a Markov process, $E_0[\cdot]$ stands for the expectation with respect to the unique invariant distribution of $\{X_t\}$, and $A(X_t)$, $b(X_t)$ are defined as

$$A(X_t) = \vec{z}_t (\phi^T (x_t) - \gamma \phi^T (x_{t+1})) \tag{23}$$

$$b(X_t) = \vec{z}_t r_t \tag{24}$$

Then, based on a set of observation data $\{(x_t, r_t)\}$ (t=1,2,…,T), a least-squares solution to the above problem can be obtained as [24]:

$$W = (\sum_{t=1}^{T} A(X_t))^{-1} \sum_{t=1}^{T} b(X_t) \tag{25}$$

4.2 Kernel-based RL for sequential behavior learning

After introducing the above Markov reward model, the intrusion detection problem using system call traces can be solved by a class of reinforcement learning algorithms called temporal-difference (TD) learning. The aim of TD learning is to predict the state value functions of a Markov reward process by updating the value function estimations based on the differences between temporally successive predictions rather than using errors between the real values and the predicted ones. And it has been verified that TD learning is more efficient than supervised learning in multi-step prediction problems [22].

Until now, TD learning algorithms with linear function approximators have been widely studied in the literature [23-24]. In [24], a linear TD learning algorithm was applied to host-based intrusion detection using sequences of system calls and very promising results have been obtained. Nevertheless, the approximation ability of linear function approximators is limited and the performance of linear TD learning is greatly influenced by the selection of linear basis functions. In the following, a sparse kernel-based LS-TD(λ) algorithm will be presented for value function prediction in host-based IDSs [25]. The sparse kernel-based LS-TD algorithm was recently developed in [26] and it was demonstrated that by realizing least-squares TD learning in a kernel-induced high-dimensional feature space, nonlinear value function estimation can be implicitly implemented by a linear form of computation with high approximation accuracy. Therefore, by making use of the kernel-based LS-TD learning algorithm, the predictions of anomaly probabilities for intrusion detection will have higher precision and it will be more beneficial to realize high-performance IDSs based on dynamic behavior modeling.

In the kernel-based LS-TD learning method [26], the same solution to the following LS-TD problem was considered:

$$E_0[\vec{z}_t(\phi^T(x_t)-\gamma\phi^T(x_{t+1}))]W^* - E_0[\vec{z}_t r_t] = 0 \tag{26}$$

where the corresponding value functions are estimated by

$$V(x_t) = \phi^T(x_t)W^*$$

Using the average value of observations as the estimation of expectation $E_0[\cdot]$, equation (26) can be expressed as follows:

$$\sum_{i=1}^{N}[\bar{z}(s_i)(\phi^T(s_i)-\gamma\phi^T(s_{i+1}))]W = \sum_{i=1}^{N}\bar{z}(s_i)r_i \tag{27}$$

Based on the idea of kernel methods, a high-dimensional nonlinear feature mapping can be constructed by selecting a Mercer kernel function $k(x_1, x_2)$ in a reproducing kernel Hilbert space (RKHS). In the following, the nonlinear feature mapping based on the kernel function $k(.,.)$ is also denoted by $\phi(s)$ and according to the Mercer Theorem [27], the inner product of two feature vectors is computed by

$$k(x_i, x_j) = \phi^T(x_i)\phi(x_j) \tag{28}$$

Due to the properties of RKHS [27], the weight vector W can be represented by the weighted sum of the state feature vectors:

$$W = \Phi_N\alpha = \sum_{t=1}^{N} \phi(s(x_t))\alpha_i \tag{29}$$

where x_i (i = 1,2,..., N) are the observed states, N is the total number of states and $\alpha = [\alpha_1, \alpha_2, ..., \alpha_N]^T$ are the corresponding coefficients, and the matrix notation of the feature vectors is denoted as

$$\Phi_N = (\phi(s_1), \phi(s_2), ..., \phi(s_N)) \tag{30}$$

For a state sequence x_i (i = 1, 2,..., N) , let the corresponding kernel matrix K be denoted as $K=(k_{ij})_{N\times N}$, where $k_{ij}=k(x_i, x_j)$.

$$\bar{Z}_N H_N K\alpha = \bar{Z}_N R_N \tag{31}$$

By substituting (28), (29) and (30) into (27), and multiplying the two sides of (27) with Φ_N^T we can get

$$\bar{Z}_N = [\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_{N-1}] = [k_{1N}, k_{2N} + \gamma\lambda\bar{z}_1, \ldots, k_{(N-1)N} + \gamma\lambda\bar{z}_{N-2}] \tag{32}$$

$$R_N = [r_1, r_2, \ldots, r_{N-1}]^T \tag{33}$$

$$H_N = \begin{bmatrix} 1 & \beta_1\gamma & & & \\ & 1 & \beta_2\gamma & & \\ & & \cdots & & \\ & & & 1 & \beta_{N-1}\gamma \end{bmatrix}_{(N-1)\times N} \tag{34}$$

In (34), the values of β_i (i=1,2,…,N-1) are determined by the following rule: when state x_{i-1} is not an absorbing state, β_i is equal to -1, otherwise, β_i is set to zero.
As discussed in [26], by using the techniques of generalized inverse matrix in [28], the kernel-based LS-TD solution to (26) is as follows:

$$\alpha = (H_N K)^+ \bar{Z}_N^+ \bar{Z}_N R_N \tag{35}$$

where $(.)^+$ denotes the generalized inverse of a matrix.
One problem remained for the above kernel-based LS-TD learning algorithm is that the dimension of the kernel-based LS-TD solution is equal to the number of state transition samples, which will cause huge computational costs when the number of observation data is large. To make the above algorithm be practical, one key problem is to decrease the dimension of kernel matrix K as well as the dimensional of α. The problem has been studied in [29] by employing an approximately linear dependence (ALD) analysis method [30] for the sparsification of kernel matrix K.
The main idea of ALD-based sparcification is to represent the feature vectors of the original data samples by an approximately linearly independent subset of feature vectors, which is to compute the following optimization problem

$$\delta_t = \min_{\bar{a}} \left\| \sum_j \bar{a}_j \phi(x_j) - \phi(x_t) \right\|^2 \tag{36}$$

During the sparsification procedure, a data dictionary is incrementally constructed and every new data sample x_t is tested by compute the solution δ_t of (36). Only if δ_t is greater than a predefined threshold, the tested data sample x_t will be added to the dictionary. For detailed discussion of the sparsification process, please refer to [29] and [30]. After the sparsification procedure, a data dictionary D_N with reduced number of feature vectors will be obtained and the approximated state value function can be represented as:

$$\tilde{V}(x) = \sum_{j=1}^{n(D_N)} \hat{\alpha}_j \hat{k}(x_j, x) \tag{37}$$

where $n(D_N)$ is the size of the dictionary.

When the above learning and sparcification process is completed, a value function model of the IDS problem can be obtained. And the accumulated anomaly probability of a state sequence $S_n=\{x_1, x_2, \ldots x_n\}$ can be computed as

$$P(s)=\frac{1}{n}\sum_{i=1}^{n}\widetilde{V}(x_i) \tag{38}$$

By selecting an appropriate threshold μ, the detection output of the adaptive IDS can be simply determined as follows:

$$\textit{If}\quad P(S_n)>\mu\text{, then raise alarms}$$

4.3 Performance evaluations

Generally speaking, previous works on machine learning methods for adaptive intrusion detection can be mainly classified into four categories, i.e., supervised learning methods, unsupervised learning methods, semi-supervised methods and statistical modeling methods. Compared with the supervised learning methods in intrusion detection, the proposed model does not require precise labeling of every observed feature, which is a difficult task and may usually lead to the poor performance of supervised methods, especially for complex sequential attacks. For unsupervised learning algorithms in intrusion detection, e.g., SOM, clustering, due to the lack of prior information, the performance of IDSs can not be optimized adequately [12].

The proposed RL-based dynamic behavior modeling approach for intrusion detection estimates the anomaly probability of states based on the learning prediction of state value functions. Therefore, it can be applied to detect complex attack behaviors with complex sequential patterns. The computational complexity of TD learning algorithms is linear with respect to the number k of state features and the length m of traces, i.e., it has time complexity of $O(km)$, which is lower than the training algorithm for HMMs, which runs in time $O(nm^2)$, where n is the number of states in the HMM and m is the size of the trace. Furthermore, since TD learning prediction methods using function approximators are commonly used, the number k of state features can become much smaller than n and the computational efficiency will be further improved.

For the RL-based approach, the most related methods are based on Markov chain modeling or Hidden Markov models (HMMs), which are anomaly detection techniques that aim to establish the probabilistic structure model of the normal data sequences explicitly. However, the Markov reward model and the TD prediction method are based on hybrid modeling strategy where the intrusion data can be combined with normal data to train the detection model. Moreover, the RL-based method only implicitly constructs the probabilistic model and the detection of anomalies is based on the estimated value functions. In [31], the robustness of Markov chain modeling techniques was studied and it was shown that when explicitly estimating the probabilistic structure of the Markov chain model for normal data, the detection accuracy was very sensitive to the noise of data, i.e., when the intrusion data were mixed with normal data, the performance of the Markov chain model would become worse. Nevertheless, in our approach, the detection accuracy is not influenced by the mixing of normal and abnormal data due to the hybrid modeling strategy.

To compare the performance between the previous HMM-based approach and the RL-based approach, experiments on host-based intrusion detection using system calls were conducted. In the experiments, two types of data sets were used, which include system call traces from the "live" lpr and the Sendmail programs. Table 1 shows some of the details of the data, which include two kinds of attack data and corresponding normal data. All of these data sets are publicly available at the website of University of New Mexico [32].

In the data sets, each trace is a sequence of system calls generated by a single process from the beginning of its execution to the end. Since the traces were generated by different programs under different environments, the number of system calls per trace varies widely. In the MIT environment, lpr was traced by running the program on 77 different hosts, each running SunOS, for two weeks, to obtain traces of a total of 2766 print jobs. For detailed discussion of the properties of the data sets, please refer to [32-33].

The two types of system call traces were divided into two parts. One part is for model training and threshold determination and the other part is for performance evaluation. Table 1 shows the numbers of normal and attack traces for training and testing. As can be seen in the table, the numbers of testing traces are usually larger than those of testing traces.

		lpr	*sendmail*
Training & Threshold Selection	Normal Trace Number	10	13
	Attack Trace Number	20	5
Testing	Normal Trace Number	2703	67
	Attack Trace Number	1001	7
Total system call number		1023950	223733

Table 1. Experimental data for host-based IDS

During the threshold determination process, the same data sets were used as the training process, i.e., the training data sets and the data sets for threshold determination are the same. For performance testing, the data sets are different from those in model training and their sizes are usually larger than the training data. In the testing stage, two criterions for performance evaluations were used, which are the detection rate Dr and the false alarm or false positive rate Fp, and they are computed as follows:

$$Dr = \frac{n_d}{n_a} \tag{36}$$

$$Fp = \frac{N_a}{N} \tag{37}$$

where n_d is the number of abnormal traces that have been correctly identified by the detection model and n_a is the total number of abnormal traces, N_a is the number of normal states that have been incorrectly identified as anomaly by the detection model, and N is the total number of normal states. In the computation of false alarm rates, we use the same ideas discussed in [4], where every possible false alarms during a long state traces are all counted and the total sum of false alarms is divided by the number of all states in traces. Therefore, the false positives were measured differently from the detection rates or true positives. To detect an intrusion, it is only required that the anomaly probabilities exceed a preset threshold at some point during the intrusion. However, making a single decision as to whether a normal trace is abnormal or not is not sufficient, especially for very long traces. For example, if a program runs for several days or more, each time that it is flagged as anomalous must be counted separately. As pointed out in [17], the simplest way to measure this is to count all the individual decisions. Then, the false-positive rate is selected as the percentage of decisions in which normal data were detected as anomalous.
In the experiments, the TD learning prediction method was applied to the above data sets. Every state in the Markov reward model has a system-call sequence length of 6, which has been widely employed in previous works. The reward function is defined by (18). A linear function approximator, which is a polynomial function of the observation states and has a dimension of 24, was used as the value function approximator. To compare the performance of TD learning prediction and previous approaches, the experimental results in [4], where HMM-based dynamic behavior modeling methods were applied to the same data sets, are also shown in the following Table 2.

	TD Prediction		HMM [9]	
	Detection rate	False alarm rate	Detection rate	False alarm rate
lpr	100%	0.00749	100%	3e-4
sendmail	100%	0.002951	61.5%	0.05
			84.6%	0.10
			92.3%	0.20

Table 2. Performance comparisons between TD and HMM methods

To compare the performance between the kernel LS-TD approach with the linear LS-TD [16] and the HMM-based approach [4], experiments on host-based intrusion detection using system calls were conducted. In the experiments, the data set of system call traces generated from the *Sendmail* program was used. The system call traces were divided into two parts.

One part is for model training and threshold determination and the other part is for performance evaluation. The normal trace numbers for training and testing are 13 and 67, respectively. The numbers of attack traces used for training and testing are 5 and 7. The total number of system calls in the data set is 223733. During the threshold determination process, the same traces were used as the training process. The testing data are different from those in model training and their sizes are usually larger than the training data.
In the learning prediction experiments for intrusion detection, the kernel LS-TD algorithm and previous linear TD(λ) algorithms, i.e., LS-TD(λ), are all implemented for the learning prediction task. In the kernel-based LS-TD algorithm, a radius basis function (RBF) kernel is selected and its width parameter is set to 0.8 in all the experiments. A threshold parameter δ=0.001 is selected for the sparsification procedure of the kernel-based LS-TD learning algorithm. The LS-TD(λ) algorithm uses a linear function approximator, which is a polynomial function of the observation states and has a dimension of 24.

	Kernel LS-TD		LS-TD[11]		HMM [7]	
	Dr	*Fp*	*Dr*	*Fp*	*Dr*	*Fp*
sendmail	1.00	0.00016	1.00	0.0029	0.615	0.05*
					0.846	0.10*
					0.923	0.20*

* The false alarm rates were only computed for trace numbers, not for single state

Table 3. Performance comparisons between different methods

The experimental results are shown in Table 3. It can be seen from the results that both of the two RL methods, i.e., the kernel LS-TD and linear LS-TD, have 100% detection rates and the kernel-based LS-TD approach has better performance in false alarm rates than the linear LS-TD method. The main reason is due to the learning prediction accuracy of kernel-based LS-TD for value function estimation. It is also illustrated that the two TD learning prediction methods have much better performance than the previous HMM-based method. Therefore, the applications of kernel-based reinforcement learning methods, which are based on the Markov reward model, will be very promising to realize dynamic behavior modeling and prediction for complex multi-stage attacks so that the performance of IDSs can be efficiently optimized.

5. Conclusions

Although in recent years, there are many research works on applying machine learning and statistical modeling methods to intrusion detection problems, the sequential modeling problem in intelligent intrusion detection has not been well solved yet. In this Chapter, the TD learning prediction method is introduced to construct detection models and improve the performance of IDSs only by simplified labeling schemes using evaluative signals or feedbacks for sequential training data. It is illustrated that compared with previous anomaly detection approaches using machine learning, the TD learning and prediction method can obtain comparable or even better detection accuracies for complex sequential attacks. More importantly, the proposed TD learning and prediction approach provides an efficient anomaly detection technique with simplified labeling procedure and reduced computational complexity. Future work may need to be focused on the extension of the proposed method to more general intrusion detection systems with real-time applications.

6. References

[1] D. Denning: An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2) (1987) 222-232

[2] M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. Alan Whitehurst. Expert systems in intrusion detection: A case study. In *Proceedings of the 11th National Computer Security Conference*, Baltimore, Maryland, October, (1988) 74-81

[3] W. K. Lee, Stolfo, S., and Mok, K.: Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, 14(6), (2000) 533 – 567

[4] D.Y. Yeung, Y.X. Ding, Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36 (2003) 229 – 243

[5] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. in *Proceedings of the 8th USENIX Security Symposium*, (1999).

[6] H.Shah, J.Undercoffer and A.Joshi: Fuzzy clustering for intrusion detection. In: *Proceedings of the 12th IEEE International Conference on Fuzzy Systems*. (2003) 1274-1278

[7] D. Barbara, N. Wu, S. Jajodia, Detecting novel network intrusions using Bayes estimators, *First SIAM Conference on Data Mining*, Chicago, IL, (2001).

[8] J. Ryan, M-J. Lin, R. Miikkulainen, Intrusion detection with neural networks, *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, AAAI Press, (1997) 72-77.

[9] X. Xu. Adaptive Intrusion Detection Based on Machine Learning: Feature Extraction, Classifier Construction and Sequential Pattern Prediction. *International Journal of Web Services Practices*, Vol.2, No.1-2 (2006), pp. 49-58

[10] M. Mahoney, P.Chan: Learning nonstationary models of normal network traffic for detecting novel attacks. In: *Proceedings of 8th International Conference on Knowledge Discovery and Data Mining*, (2002) 376-385

[11] X. Xu, X. N. Wang, Adaptive network intrusion detection method based on PCA and support vector machines . *Lecture Notes in Artificial Intelligence*, ADMA 2005, LNAI 3584, (2005) 696 – 703.

[12] P. Laskov, P. Düssel, C. Schäfer, K. Rieck, Learning intrusion detection: supervised or unsupervised? Proc. ICIAP 2005, September, *Lecture Notes in Computer Science* , LNCS 3617 (2005) 50-57

[13] W.K. Lee, S.J.Stolfo: A data mining framework for building intrusion detection model. In: Gong L., Reiter M.K. (eds.): *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, CA: IEEE Computer Society Press (1999) 120~132

[14] http://www.kdnuggets.com/datasets/kddcup.html

[15] Y. H. Liao, V. Rao Vemuri, Using text categorization techniques for intrusion detection, *Proceedings of the 11th USENIX Security Symposium*, August, (2002) 51-59.

[16] X.Xu, Intrusion Detection Based on Dynamic Behavior Modeling: Reinforcement Learning versus Hidden Markov Models, *International Journal of Computational Intelligence Theory and Practice*, 2(1), (2007) 57-66

[17] T. Lane, C. Brodley, Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3) (1999) 295–331

[18] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2): 257-286, 1986.

[19] S. Hofmeyr et al., Intrusion detection using sequences of systems call, *Journal of Computer Security*, 6 (1998) 151-180

[20] X.Xu, A Reinforcement Learning Approach for Host-Based Intrusion Detection Using Sequences of System Calls. *Lecture Notes in Computer Science*, LNCS 3644, pp. 995 – 1003

[21] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, vol. 4, (1996) 237--285.

[22] R. Sutton, Learning to predict by the method of temporal differences. *Machine Learning*, 3(1), (1988) 9-44

[23] X. Xu, H. G. He, D. W. Hu: Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, vol.16, (2002) 259-292

[24] J. A. Boyan, Technical Update: Least-squares temporal difference learning. *Machine Learning*, 49, (2002) 233-246

[25] X. Xu, Yirong Luo, A Kernel-Based Reinforcement Learning Approach to Dynamic Behavior Modeling of Intrusion Detection, In : D. Liu et al. (Eds.): ISNN 2007, *Lecture Notes in Computer Science*, LNCS 4491, Part I, (2007) 459–468

[26] X. Xu, et al., Kernel Least-Squares Temporal Difference Learning, *International Journal of Information Technology*,11(9), (2005) 54-63

[27] Schölkopf, B., Smola, A.: *Learning with Kernels*. Cambridge, MA: MIT Press (2002)

[28] Nashed, M. Z., ed.: *Generalized Inverses and Applications*. Academic Press, New York, (1976)

[29] Xu, X.: A Sparse Kernel-Based Least-Squares Temporal Difference Algorithm for Reinforcement Learning. In: Proceedings of International Conference on Intelligent Computing. 2006, *Lecture Notes in Computer Science*, LNCS 4221 (2006) 47-56

[30] Engel, Y., Mannor, S., Meir, R.: The Kernel Recursive Least-Squares Algorithm. *IEEE Transactions on Signal Processing*, 52 (8) (2004) 2275-2285

[31] N. Ye, Y. Zhang, and C. M. Borror. Robustness of the Markov-Chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1), (2004) 116-123.

[32] http://www.cs.unm.edu/~immsec/data/

[33] C. Warrender, S. Forrest, B. Pearlmutter. Detecting intrusions using system calls: alternative data models. in the *1999 IEEE Symposium on Security and Privacy*, May 9-12, (1999)

Edited by Abdelhamid Mellouk and Abdennacer Chebira

Machine Learning can be defined in various ways related to a scientific domain concerned with the design and development of theoretical and implementation tools that allow building systems with some Human Like intelligent behavior. Machine learning addresses more specifically the ability to improve automatically through experience.

Photo by agsandrew / iStock

IntechOpen

ISBN 978-953-51-5838-7