IntechOpen

Recent Applications in Data Clustering

Edited by Harun Pirim

RECENT APPLICATIONS IN DATA CLUSTERING

Edited by **Harun Pirim**

Recent Applications in Data Clustering
http://dx.doi.org/10.5772/intechopen.71315
Edited by Harun Pirim

Contributors

Reda Gharieb, Khaled Abdalgader, Hadeel Aljobouri, Hussain A. Jaber, Ilyas Çankaya, Xiaodong Feng, Vladimir Ryazanov, Masafumi Nakagawa, Milan Vukicevic, Vladimir Urosevic, Ana Kovacevic, Firas Kaddachi, Loai Abdallah, F. Marta L. Di Lascio, Uğurhan Kutbay, Chuan Chen, Zibin Zheng, Fanghua Ye, Zitai Chen, Hui Qian, Rui Li, Fred Glover, Yang Wang

Notice

First published in London, United Kingdom, 2018 by IntechOpen
eBook (PDF) Published by IntechOpen, 2019
IntechOpen is the global imprint of INTECHOPEN LIMITED, registered in England and Wales, registration number: 11086078, The Shard, 25th floor, 32 London Bridge Street
London, SE19SG – United Kingdom
Printed in Croatia

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

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

Recent Applications in Data Clustering
Edited by Harun Pirim

p. cm.

Print ISBN 978-1-78923-526-5

Online ISBN 978-1-78923-527-2

eBook (PDF) ISBN 978-1-83881-560-8

Meet the editor

Harun Pirim received his PhD degree in Industrial and Systems Engineering (ISE) from the Mississippi State University since 2011. He received his MA degree in Operations Research and his BSc degree in Industrial Engineering from the Dokuz Eylul University, Turkey. He worked on microarray data analysis in collaboration with computer science and forestry departments. He recently developed a minimum spanning tree-based algorithm and a mixed-integer linear programming model for clustering applications. His research interests include mathematical programming, discrete optimization, and graph mining applications in biology, sociology, and supply chain fields. He has published several conference papers, journal papers, and book chapters. He is funded by internal projects from the King Fahd University of Petroleum and Minerals (KFUPM) as a principal investigator and a coinvestigator. He served as a reviewer of many reputable journals.

Contents

Preface

We are experiencing a transition from an information age to a wisdom age driven by an explosion in data available for analysis. A major consequence of this transition is an evolution of data mining to become data analytics, a discipline involving engineers, statisticians, and computer scientists together with the diverse realms they serve. The potential value that resides in data is suggested in the famous saying "data is the oil of the age," but suitable theories and tools are needed to tease this value into the open.

Clustering has emerged as one of the more fertile fields within data analytics, widely adopted by companies, research institutions, and educational entities as a tool to describe similar/different groups, communities, patterns, modules, and objects and broadly to predict assignment of certain members to unlabeled groups in an unsupervised fashion. Often classed as an instance of machine learning, clustering has found applications to generate groups consisting of market segments, genes, constellations of stars, movies to recommend, facilities to serve critical functions, and remarkable communities within a society.

The history of clustering dates back to ancient times, manifest in Aristotles' taxonomy of living things, and quite possibly can be traced even earlier. Just as counting is essential to computation, clustering is essential for learning and predicting. Hence, clustering algorithms have been developed in rich abundance.

This book is intended to provide a view of recent contributions to the vast clustering literature that offers useful insights within the context of modern applications for professionals, academics, and students. The book spans the domains of clustering in image analysis, lexical analysis of texts, replacement of missing values in data, temporal clustering in smart cities, comparison of artificial neural network variations, graph theoretical approaches, spectral clustering, multiview clustering, and model-based clustering in an R package. Image, text, face recognition, speech (synthetic and simulated), and smart city datasets are used. The table below is a summary of chapters according to the types of theory and applications they represent:

Chapter	Theory	Applications
1	Missing values, K-means, mean shift	Six datasets, speech and image processing unit
2	K-means, C-means, colored C-means, genetic algorithm	İmage clustering
3	KL divergence C-means	İmage clustering
4	K-means, semantic similarity	Text fragmentation

Chapter	Theory	Applications
5	Point cloud clustering, surface, terrestrial laser scanning	Three laser scanner datasets
6	Model-based clustering	R package, simulated examples
7	Hidden Markov temporal clustering models	Smart cities, IoT, anomaly detection, City4Age pilot sites
8	Tree-based clustering, spanning forest	
9	Spectral clustering, weight matrix construction	Three datasets from UCI, three face recognition datasets
10	ANNs, neural gas, and two variations	MatlabGUI, six synthetic datasets
11	Multiview clustering methods	K-means, spectral, matrix factorization, tensor decomposition, deep learning
12	Ensemble clustering, stability	

Some of the distinguishing features of these contributions are as follows:

Loai AbdAllah and Ilan Shimshoni develop a new distance function that computes distances over incomplete datasets. The distances are employed in K-means and mean shift algorithms. The procedure is compared with mean imputation (MI), mean attribute (MA), and most common value (MCA) replacements. Experiments are run on six standard numerical datasets.

Uğurhan Kutbay reviews partitional clustering focusing on K-means, fuzzy C-means, colored fuzzy C-means, and a genetic algorithm. Algorithms are applied on the same image data.

Reda R. Gharieb incorporates the influence of an object's neighborhood employing two Kullback-Leibler (KL) membership divergences for clustering image data. A local membership function is embedded into the objective function of hard C-means, which prevents additive noise. Partition coefficient and partition entropy measures are adopted to evaluate the performance of fuzzy clustering algorithms. A synthetic image, a simulated MRI image, and the standard Lena image are used to compare conventional C-means algorithms with two different KL divergence fuzzy C-means algorithms.

Khaled Abdalgader presents a new version of the original K-means algorithm to cluster small-sized text fragments. This new variation measures the semantic similarity between sentences based on the idea of generating a synonym expansion set to be used in the compared semantic vectors. The algorithm is compared with Spectral Clustering Affinity Propagation, K-medoids, STC-LE, and K-means (TF-IDF) using Reuters-21578, Aural Sonar, Protein, Voting, SearchSnippets, StackOverflow and Biomedical datasets, and Purity, Entropy, V-measure, Rand Index, and F-measure validation metrics.

Masafumi Nakagawa focuses on region-based point cloud clustering to improve 3D visualization and modeling using massive point clouds, based on a combined point cloud clustering methodology and point cloud filtering on a multilayered panoramic range image. Indoor MMS data and two terrestrial laser scanner data are used to test the approach.

Marta presents a clustering algorithm based on the copula function and the R package CoClust. The range (or set) of clusters from which the procedure automatically selects the best one and the sample size to be used to select it can be varied. The algorithm is able to find

clusters according to the complex multivariate dependence structure of the data-generating process. The main R commands are used to perform a fully developed clustering of multivariate-dependent data through numerical examples.

Milan Vukicevic et al. propose a methodology for behavior variation and anomaly detection from acquired sensory data, based on hidden Markov temporal clustering models (HMMs). Data are collected from five prominent European smart cities and Singapore, which aim to become fully "elderly friendly."

Glover and Wang introduce a class of tree-based clustering methods based on a single parameter W and show how to generate the full collection of cluster sets C(W), without duplication, by varying W according to conditions identified automatically during the algorithm's execution. The number of clusters within C(W) for a given W is also determined automatically and provides a wide range of clusters with different structures from the same dataset.

Xiaodong Feng presents robust spectral clustering via sparse representation proposing two approaches of weight matrix construction according to the similarity of the sparse coefficient vectors. The method is compared with K-means and spectral clustering approaches using Gaussian RBF, SIS, l1 -Directed Graph Construction, and Nonnegative SIS using external metrics clustering accuracy (CA) and normalized mutual information (NMI).

Hadeel K. Aljobouri et al. compare the performances of three artificial neural network (ANN) applications: neural gas (NG), growing neural gas (GNG), and robust growing neural gas (RGNG) algorithms.

Chuan Chen et al. summarize K-means, spectral clustering, matrix factorization, tensor decomposition, and deep learning with their multiview learning versions.

Ryazanov V.V. addresses the problem of finding the best committee synthesis of ensemble clustering formulated as a discrete optimization problem.

I express my sincere congratulations to all the contributing authors for their outstanding innovations and insights. Special thanks go to Fred W. Glover for our fertile interactions and discussions of shared research interests.

Harun Pirim, Ph.D.
Assistant Professor
Systems Engineering
King Fahd University of Petroleum and Minerals
Saudi Arabia

Chapter 1

Clustering Algorithms for Incomplete Datasets

Loai AbdAllah and Ilan Shimshoni

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.78272

Abstract

Many real-world dataset suffers from the problem of missing values. Several methods were developed to deal with this problem. Many of them filled the missing values within fixed value based on statistical computation. In this research, we developed a new versions of the *k-means* and the *mean shift* clustering algorithms that deal with datasets with missing values without filling their values. We developed a new distance function that is able to compute distances over incomplete datasets. The distance was computed based only on the *mean* and *variance* of the data for each attribute. As a result, the runtime complexity of our computation was $O(1)$. We experimented on six standard numerical datasets from different fields. On these datasets, we simulated missing values and compared the performance of the developed algorithms using our distance and the suggested mean computations to other three basic methods. Our experiments show that the developed algorithms using our distance function outperform the existing *k-means and mean shift* using other methods for dealing with missing values.

Keywords: missing values, distance metric, weighted Euclidean distance, clustering, mean shift, k-means

1. Introduction

Missing values in data are common in real-world applications. They can be caused by human error, equipment failure, system-generated errors, and so on.

In this research, we developed two popular clustering algorithms to run over incomplete datasets: (1) k-means clustering algorithm [1] and (2) mean shift clustering algorithms [2].

Based on [3–6], there are three main types of missing data:

1. Missing completely at random (MCAR): when the missing value is not related to any other sample;
2. Missing at random (MAR): when the probability that a value is missing may depend on some known values but it does not depend on the other missing values;
3. Not missing at random (NMAR): when the probability that a known value is missing depends on the value that would have been observed.

There are two basic types of methods to deal with the problem of incomplete datasets. (1) Deletion: methods from this category ignore all the incomplete instances. These methods may change the distribution of the data by decreasing the volume of the dataset [7]. (2) Imputation: in these methods, the missing values were replaced with known value according to statistical computation. Based on these methods, we convert then incomplete data to complete data, and as a result, the existing machine learning algorithms can be run as they deal with complete data.

One of the most common approaches in this domain is the mean imputation (MI) method that replaces each incomplete data point with the mean of the data. There are several obvious disadvantages to this method: (a) using a fixed instance to replace all the incomplete instances will change the distribution of the original dataset and (b) ignoring the relationship among attributes will bias the performance of subsequent data mining algorithms. These problems were caused since we replace all the incomplete instances with a fixed one. On the other hand, a variant of this method is to replace the missing values only based on the distribution of the attributes. It means that the algorithm will replace each missing value with the mean of the of its attribute (MA) and the whole instance [8]. And in a case that the values were discrete, the missing value will be replaced by the most common (MCA) value in the attribute [9] (i.e., filling the unknown values of the attribute with the value that occurs most often for the same attribute). All those methods ignore the other possible values of the attribute and their distribution and represent the missing value with one value, that is, wrong in real-world datasets.

Finally, the ***k*-Nearest Neighbor Imputation** method [10, 11] estimates the values that should be replaced based on the k nearest neighbors based only on the known values. The main obstacle of this method is the runtime complexity.

We can summarize the main drawbacks of each suggested method as: (1) inability to approximate the missing value and (2) inefficiency to compute the suggested value. Based on our suggested method [12], the distance between two points, that they may include missing value, is not only efficient but also takes into account the distribution of each attribute.

To do that in the computation procedure, we take into account all the possible values of the missing value with their probabilities, which are derived from the attribute's distribution. This is in contrast to the MCA and the MA methods, which replace each missing value only with the mode or the mean of each attribute.

There are three possible cases between the values: (a) both of them are known: in this case, the distance will be computed as the Euclidean distance; (b) both of them are missing; and (c) one

value is missing. In the last two cases, the distance will be computed based only on the *mean* and the *variance* of the attribute. As a result, the runtime of the developed distance is $O(1)$ as the Euclidean distance.

In this research, we integrated this distance function in order to develop the *k*-means and the mean shift clustering algorithms. To this end, we derived more two formulas to compute the mean (for *k*-means algorithm) and for computing the gradient function of the local estimated density (for mean shift clustering algorithm).

The developed algorithms yield better results than the other methods and preserve the runtime of the algorithms which deals with complete data as can be seen in the experiments. We experimented on six standard numerical datasets from different fields from the Speech and Image Processing Unit [13]. Our experiments show that the performance of the developed algorithms using our distance function was superior to using other methods.

This chapter is organized as follows. A review of our distance function (MD_E) is described in Section 2. The mean computation is presented in Section 3. Section 3 describes several directions for integrating the (MD_E) distance and the computed *mean* within the k-means clustering algorithm. The mean shift clustering algorithm is presented in Section 4. Section 4.1 describes how to integrate the (MD_E) distance and the derived mean shift vector within the mean shift clustering algorithm. Experimental results of running the developed clustering algorithms are presented in Section 5. Finally, our conclusions and future work are presented in Section 6.

2. Our distance measure

Firstly, we will give a short preview to basic distance function that is able to compute distances between points with missing values developed by [2].

Let $A \subseteq \mathbb{R}^K$ be a set of points. For the ith attribute A^i, the conditional probability for A_i will be computed according to the known values for this attribute from A (i.e., $P(A^i) \sim \chi^i$), where χ^i is the distribution of the ith coordinate.

Given two sample points $X, Y \subseteq \mathbb{R}^K$, the goal is to compute the distance between them. Let x^i and y^i be the ith coordinate values from points X, Y, respectively. There are three possible cases for the values of x^i and y^i:

1. Two values are known: the distance between them will be defined as the Euclidean distance.

2. One value is missing: Suppose that x^i is missing and the value y^i is given. Since the value of x^i is unknown, we cannot compute the distance using the Euclidean distance equation. Instead, we compute the expectation of all the distances between the given value y^i and all the possible values from attribute i according to its distribution χ^i.

   Therefore, we approximate the mean Euclidean distance (MD_E) between y^i and the missing value m^i as:

$$MD_E(m^i, y^i) = E\left[(x - y^i)^2\right] = \int p(x)(x - y^i)^2 dx = \left((y^i - \mu^i)^2 + (\sigma^i)^2\right).$$

That means, to measure the distance between known value y^i and unknown value, the algorithm will compute the expectation distance for all the distances between y^i and all the possible values of the missing value. These computations did not take into account the possible correlations between the missing values and the other known values (missing completely at random —MCAR) and the probability was computed according to the whole dataset. The resulting mean Euclidean distance will be:

$$MD_E(m^i, y^i) = \left((y^i - \mu^i)^2 + (\sigma^i)^2\right), \tag{1}$$

where μ^i and $(\sigma^i)^2$ are the *mean* and the *variance* for all the known values of the attribute.

3. Both values are missing: In this case, in order to measure the distance, we should compute all the distances between each possible pair of values one for each missing value x^i and y^i. Both these values are selected from distribution χ^i.

Then, we compute the expectation of the Euclidean distance between each selected value as we did for the one missing value problem. As a result, the distance is:

$$MD_E(x_i, y_i) = \int\int p(x)p(y)(x - y)^2 dxdy = \left((E[x] - E[y])^2 + \sigma_x^2 + \sigma_y^2\right).$$

As x and y belong to the same attribute, $E[x] = E[y] \coloneqq \mu^i$ and $\sigma_x = \sigma_y \coloneqq \sigma^i$. Thus:

$$MD_E(x^i, y^i) = 2(\sigma^i)^2. \tag{2}$$

As we mentioned, all these computations assume that the missing data is MCAR. However, in real-world datasets, the missing data are MAR. In this case, the probability $p(x)$ depends on the other observed values, and then, the distance will be computed as:

$$MD_E(m^i, y^i) = \int p(x|x_{obs})(x - y^i)^2 dx = \left(\left(y^i - \mu^i_{x|x_{obs}}\right)^2 + \left(\sigma^i_{x|x_{obs}}\right)^2\right),$$

where x_{obs} denotes the observed attributes of point X, and $\mu^i_{x|x_{obs}}$ and $\left(\sigma^i_{x|x_{obs}}\right)^2$ are the conditional *mean* and *variance*, respectively.

On the other hand, in the case that the missing values are NMAR, the probability $p(x)$ that was used in Eq. (1) will be computed based on this information, and then, the distance will be:

$$MD_E(m^i, y^i) = \int p(x|m^i)(x - y^i)^2 dx = \left(\left(y^i - \mu^i_{x|m^i}\right)^2 + \left(\sigma^i_{x|m^i}\right)^2\right),$$

where $p(x|m^i)$ is the distribution of x when x is missing.

3. Mean computation

Since one of our goals is developing a k-means clustering algorithm over incomplete datasets, we need to derive a formula to compute the mean of a given set that may contain incomplete points. We decide to derive this equation based on our distance function MD_E.

Let $A \subseteq \mathbb{R}^K$ be a set of n points that may contain points with missing values. Then, the *mean* of this dataset is defined as:

$$\overline{x} = \underset{x \in \mathbb{R}}{\arg\min} \sum_{i=1}^{n} \left(distance\left(x, p_i\right)\right)^2,$$

for any $x \in \mathbb{R}^K$, where $p_i \in A$ denotes each point from the set A, and $distance()$ is a distance function.

Let $f(x)$ be a multidimensional function: $f : \mathbb{R}^K :\rightarrow \mathbb{R}$ which is defined as:

$$f(x) = \sum_{i=1}^{n} \left(distance\left(x, p_i\right)\right)^2,$$

In our case, the $distance() = MD_E$. Thus,

$$f(x) = \sum_{i=1}^{n} \left(distance\left(x, p_i\right)\right)^2 = \sum_{i=1}^{n} \left(\underbrace{\sqrt{\sum_{j=1}^{K} MD_E\left(x^j, p_i^j\right)}}_{\text{The } MD_E() \text{ distance}} \right)^2 = \sum_{i=1}^{n} \sum_{j=1}^{K} MD_E\left(x^j, p_i^j\right),$$

where x^j is the coordinate j and p_i^j is the coordinate j in point p_i. Since each point p_i may contain missing attributes, and according to the definition of the MD_E distance in the previous section, $f(x)$ will be:

$$f(x) = \sum_{j=1}^{K} \left[\underbrace{\sum_{i=1}^{n_j} \left(x^j - p_i^j\right)^2}_{\text{there are } n_j \text{ known coordinates}} + \underbrace{\sum_{i=1}^{m_j} \left(\left(x^j - \mu^j\right)^2 + \left(\sigma^j\right)^2\right)}_{\text{there are } m_j \text{ missing coordinates}} \right].$$

$\overline{x}$ is the solution of $f'(x) = 0$, and in a multidimensional case: $\overline{x}$ is the solution of $\nabla f = \vec{0}$, where

$$\nabla f = \left(f'_{x^1}, f'_{x^2}, \ldots, f'_{x^k}\right) = 0,$$

is the gradient of function f. Firstly, we will deal with one coordinate, and then, we will generalize it for the other coordinates.

$$\Rightarrow f'_{x^l} = 2\sum_{i=1}^{n_l}\left(x^l - p_i^l\right) + 2\sum_{i=1}^{m_l}\left(x^l - \mu^l\right) = 0$$

$$\Rightarrow nx^l = \sum_{i=1}^{n_l} p_i^l + m_l\mu^l \Rightarrow x^l = \frac{\sum_{i=1}^{n_l} p_i^l}{n} + \frac{m_l\mu^l}{n}$$

$$\Rightarrow x^l = \frac{n_l}{n}\frac{\sum_{i=1}^{n_l} p_i^l}{n_l} + \frac{n-n_l}{n}\mu^l = \mu^l.$$

Thus, we simply get:

$$x^l = \mu^l. \tag{3}$$

Repeating this for all the coordinates yields $\overline{x} = \left(\mu^1, \mu^2, \ldots, \mu^k\right)$. In other words, each coordinate of the mean is the mean of the known values of that coordinate.

In the same way, we derive a formula for computing the weighted mean for each coordinate l, yielding:

$$\overline{x}_w^l = \frac{\sum_{i=1}^{n_l} w_i x_i^l + \sum_{i=1}^{n_l} w_i \mu^l}{\sum_{i=1}^{n} w_i},$$

where w_i is the weight of point x_i. It means, in order to compute the weighted mean of a set of numbers that some of them are unknown, we must distinguish between known and unknown values. If the value is known, we multiply it with its weight. On the other hand, if the value is missing, we replace it with the mean of the known values and then multiply it by the matching weight.

4. k-Means clustering using the MD_E distance

Based on the derived formulas, the MD_E distance and the *mean*, our aim in this research is to develop *k*-means clustering algorithms for incomplete datasets [1].

The MD_E distance and the *mean* are general and can be integrated within any algorithm that computes distances or mean computation. In this section, we describe our proposed method to integrate those formulas within the framework of the *k*-means clustering algorithm.

We developed three different versions for k-means. For simplicity, we assume that all the points are from $\mathbb{R}^2$. We have two way to look about incomplete points. The first one considers each point as a single point, this version is similar to the GMM algorithm described in [14, 15]. On the other hand, the second way is to replace each incomplete point with a set of points according to the data distribution (these are the other two methods). As will be shown in our experiments, they outperform the first algorithm.

The k-means clustering algorithm is constructed from two basic steps: (1) associate each point with its closest centroid, and then, (2) update the centroid based on the new association from Eq. (1). Given dataset D that may contain points with missing values. In the first step, the MD_E distance is used to compute the distances between each data point and the centroids in order to associate each point with the closest centroid. This association is general for all the three versions. However, there are several possible ways to then compute the new centroids of the clusters. We use **Figure 1(a)** in order to illustrate those possibilities. In this example, we see two clusters (i.e., $C1$ was assigned to be the yellow cluster and $C2$ was assigned to be the brown cluster). Our goal is to calculate the centers of each cluster. As an example, we will deal only with $C1$. If all the instances do not contain missing values, the centroid will be computed based on the Euclidean *mean* formula, resulting in the magenta star.

However, when the associated points for a given cluster contain incomplete points, it is not clear how to compute the mean. In the given example, let $(x_0, ?)$ (i.e., the red star) be a point with a missing y value and $x = x_0$. This point was associated with $C1$'s cluster using the MD_E distance. It is important to note that we are able to associate incomplete points with closest centroid even though their geometric locations are unknown since we use the MD_E distance.

On the other hand, using the MD_E distance is similar to use the MA-method based on the Euclidean distance, the point $(x_0, ?)$ will be replaced with $\left(x_0, \mu_y\right)$. It is clear that the difference between the two methods is only the variance of known values in coordinate y, a fixed value that does not influence the association result.

The naïve method to compute the new centroid is by replacing the point with the missing value with all the possible points

$$(x_0)_{possible} = \left\{ \left(x_0, y_p\right) | y_p \in Y_{possible} \right\},$$

the set of all the possible points that satisfy $x = x_0$. And

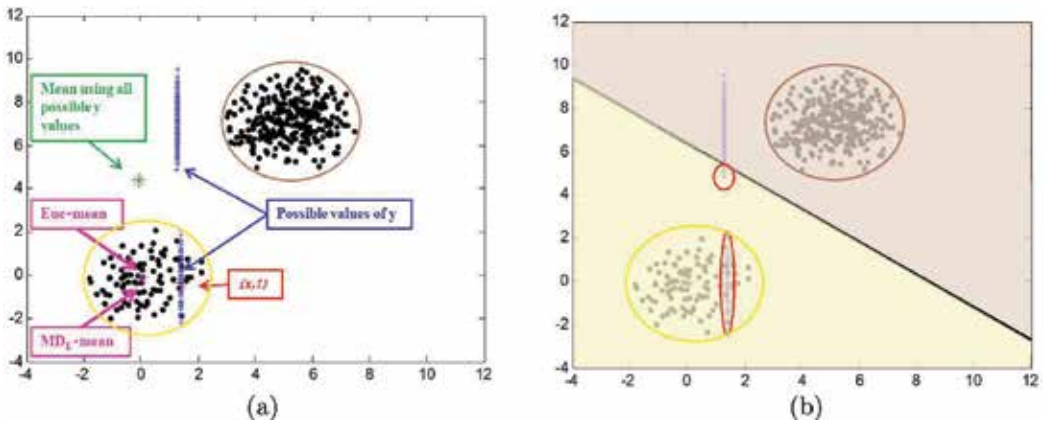

Figure 1. An example for computing the centroids for two clusters in a dataset with missing values. (a) shows the results of the different methods of computing the *mean*. (b) shows the Voronoi diagram.

$$Y_{possible} = \{y \in \mathbb{R} | \exists (x, y) \in D\},$$

denote all the possible values for attribute Y. And then computing the *mean* according to these points ($C1_{real}$ and $(x_0)_{possible}$), where each point from $C1_{real}$ has weight one and each point from $(x_0)_{possible}$ has weight $\frac{1}{|Y_{possible}|}$. Where

$$C1_{real} = \{(x, y) \in D | (x, y) \in C1\}$$

be the set of all the data points without missing values that are associated with the $C1$ cluster. As a result, the weighted *mean* of $C1$ is:

$$mean(C1) = \frac{\sum_{(x,y) \in C1_{real}} (x, y) + \left(x_0, \mu_y\right)}{|C1_{real}| + \sum \frac{1}{|Y_{possible}|}}. \tag{4}$$

This is identical to the Euclidean *mean* when the missing point is replaced with $\left(x_0, \mu_y\right)$ and is equivalent to the MA method when $\left(x_0, \mu_y\right)$ is associated with $C1$. As a result, the real centroid of the cluster (the magenta star) moves to the green star as described in **Figure 1(b)**, where not all the blue "+" marks are belonging to $C1$.

As a result, the mean computation must distinguish between two possible methods. The first method (which we call ***k-mean-***MD_E) takes into account all the possible points that their y coordinates are the y coordinates of the real data points from the yellow cluster in addition to the real points within the yellow circle. As a result, the *mean* of this set will be computed based on all the real points $C1_{real}$ and $C1_{(x_0)_{possible}}$ where,

$$C1_{(x_0)_{possible}} = \left\{ \left(x_0, y_p\right) \in (x_0)_{possible} | \exists (x, y) \in C1_{real} \wedge y = y_p \right\}.$$

Computing the new centroid using Eq. (3) yields not only the same centroid as using the Euclidean distance, but also preserves the runtime of the standard k-means using the Euclidean distance.

The second method (which we called ***k-mean-HistMD***$_E$): In this case, we first associate each of the points from $(x_0)_{Ypossible}$ with its nearest center, and after that compute a weighted *mean*. It means that to compute the *mean*, we will take into account all the real points $C1_{real}$, in addition to $PC1_{possible}$ where

$$PC1_{possible} = \left\{ \left(x_0, y_p\right) \in (x_0)_{possible} | \left(x_0, y_p\right) \in C1 \right\}.$$

According to this method, use all the points from $(x_0)_{possible}$ that are associated with the $C1$ cluster and not only the points from $(x_0)_{possible}$ whose y coordinates are from the real points associated with that cluster. Since the weights are computed using the entire dataset, we cannot use Eq. (3). To this end, our suggested method for implementing the *mean* computation is simply to replace each point with a missing value with the $|Y_{possible}|$ points, each with a weight $\frac{1}{|Y_{possible}|}$, and run

weighted k-means on the new dataset. This method, in one hand, is simple to implement, but in the other hand, its runtime is high, since each point with, for example, a missing y value will be replaced with all $|Y_{possible}|$ points. As a result, the size of the dataset will be:

$$|D_{real}| + (|D| - |D_{real}|) \cdot |Att_{possible}|,$$

where D_{real} is the set of each data points that do not contain missing values. In order to reduce the runtime complexity, we turn to use Voronoi diagram. Based on Voronoi diagram, the data space is partitioned to k subspaces (as can be seen in **Figure 1(b)**). Each point is associated with the subspace of the cluster in which it lies.

The third possibility is to divide the y value space to several disjoint intervals. Where, each interval will be represented by its mean, and the weight of each interval will be the ratio between the number of points in the interval to the number of all possible points. This method we called ***k-mean-HistMD***$_E$. *k-mean-HistMD*$_E$ method approximates the two methods mentioned before that compute the weighted *mean*.

In conclusion, we have three methods:

- The naïve method which is equivalent to the MA method.
- k-means-MD_E
- k-mean-HistMD$_E$

These methods differ in their performance, efficiency, and the way they work.

5. Mean shift algorithm

In this section, we will describe another use case that integrates the derived distance function MD_E within the framework of mean shift clustering algorithm. Firstly, we will give a short overview of the mean shift algorithm, and then, we will describe how we use MD_E distance in this algorithm. Here, we only review some of the results described in [16, 17] which should be consulted for the details. Let $x_i \in \mathbb{R}^d, i = 1, \ldots, n$ is associated with a bandwidth value $h > 0$. The *sample point* density estimator at point x is

$$\widehat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{5}$$

Based on a symmetric kernel K with bounded support satisfying

$$K(x) = c_{k,d} k\left(\|x\|^2\right) \qquad \|x\| \leq 1 \tag{6}$$

is a nonparametric estimator of the density at x in the feature space. Where $k(x), 0 \leq x \leq 1$ is the *profile* of the kernel and the normalization constant $c_{k,d}$ assures that $K(x)$ integrates to one. As a result, the density estimator Eq. (5) can be rewritten as

$$\widehat{f}_{h,k}(x) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k\left(\left\| \frac{x - x_i}{h} \right\|^2 \right). \tag{7}$$

As a first step in the analysis is to find the modes of the density which are located among the zeros of the gradient $\nabla f(x) = 0$, of a feature space with the underlying density $f(x)$, and the mean shift procedure is a way to find these zeros without the need to estimate the density.

Therefore, the density gradient estimator is obtained as the gradient of the density estimator by capitalizing on the linearity of Eq. (7).

$$\nabla \widehat{f}_{h,K}(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x - x_i) k'\left(\left\| \frac{x - x_i}{h} \right\|^2 \right). \tag{8}$$

Define $g(x) = -k'(x)$, then the kernel $G(x)$ is defined as:

$$G(x) = c_{g,d} g\left(\|x\|^2 \right).$$

Introducing $g(x)$ into Eq. (8) yields

$$\begin{aligned} \nabla \widehat{f}_{h,K}(x) &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \right] \left[\frac{\sum_{i=1}^{n} x_i g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)} - x \right], \end{aligned} \tag{9}$$

where $\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)$ is assumed to be a positive number. Both terms of the product in Eq. (9) have special significance. The first term is proportional to the density estimate at x computed with the kernel G. The second term

$$m_G(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right)} - x \tag{10}$$

is called the *mean shift vector*. The mean shift vector thus points toward the direction of maximum increase in the density. The implication of the mean shift property is that the iterative procedure

$$y_{j+1} = \frac{\sum_{i=1}^{n} x_i g\left(\left\| \frac{y_j - x_i}{h} \right\| \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{y_j - x_i}{h} \right\| \right)} \quad j = 1, 2, \ldots \tag{11}$$

In real world, most often the convergence points of this iterative procedure are the local maxima (modes) of the density. All the points that share the same mode are clustered within the same cluster. Therefore, we get clusters as the number of modes.

5.1. Mean shift computing using the MD_E distance

This section describes the way to integrate the MD_E distance within the framework of the mean shift clustering algorithm. To achieve this mission, we will first compute the mean shift vector using the MD_E distance. And then, we will integrate the MD_E and the derived mean shift vector within the mean shift algorithm.

Using the derived MD_E distance the density estimator in Eq. (7) will be written as:

$$\widehat{f}_{h,k}(x) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\left\|\frac{x-x_i}{h}\right\|^2\right) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\frac{\sum_{j=1}^{d} MD_E\left(x^j, x_i^j\right)^2}{h^2}\right). \tag{12}$$

Since each point x_i may contain missing attributes, $\widehat{f}_{h,k}(x)$ will be:

$$\widehat{f}_{h,k}(x) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\underbrace{\frac{\sum_{j=1}^{kn_i} MD_E\left(x^j, x_i^j\right)^2}{h^2}}_{\text{each } x_i \text{ has } kn_i \text{ known attributes}} + \underbrace{\frac{\sum_{j=1}^{unkn_i} MD_E\left(x^j, x_i^j\right)^2}{h^2}}_{\text{each } x_i \text{ has } unkn_i \text{ missing attributes}}\right).$$

According to the definition of the MD_E distance, we obtain:

$$\widehat{f}_{h,k}(x) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\frac{\sum_{j=1}^{kn_i}\left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i}\left(x^j-\mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2}\right). \tag{13}$$

Now, we will compute the gradient of the density estimator in Eq. (13).

$$\begin{aligned}\nabla\widehat{f}_{h,k}(x) &= \frac{c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}\left[\sum_{j=1}^{kn_i}\left(x^j - x_i^j\right)^2 + \sum_{j=1}^{unkn_i}\left(x^j-\mu^j\right)^2 + \left(\sigma^j\right)^2\right]' \\ &\quad \cdot k'\left(\frac{\sum_{j=1}^{kn_i}\left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i}\left(x^j-\mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2}\right) \\ &= \frac{c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}\left[\sum_{j=1}^{kn_i}\left(x^j - x_i^j\right)^2\right]' \cdot k'\left(\frac{\sum_{j=1}^{kn_i}\left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i}\left(x^j-\mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2}\right) \\ &\quad + \left[\sum_{j=1}^{unkn_i}\left(x^j-\mu^j\right)^2 + \left(\sigma^j\right)^2\right]' \cdot k'\left(\frac{\sum_{j=1}^{kn_i}\left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i}\left(x^j-\mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2}\right).\end{aligned}$$

In our computation, we will first deal with one coordinate l, and then, we will generate the computation for all the other coordinates.

$$
\begin{aligned}
\Rightarrow f'_{x^l} &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n_l} \left(x^l - x_i^l\right) \cdot k' \left(\frac{\sum_{j=1}^{kn_i} \left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i} \left(x^j - \mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2} \right) \\
&+ \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{m_l} \left(x^l - \mu^l\right) \cdot k' \left(\frac{\sum_{j=1}^{kn_i} \left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i} \left(x^j - \mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2} \right) \\
&= \frac{2c_{k,d}}{nh^{d+2}} [x^l \cdot \sum_{i=1}^{n} k' \left(\frac{\sum_{j=1}^{kn_i} \left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i} \left(x^j - \mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2} \right) \\
&- \sum_{i=1}^{n_l} x_i^l \cdot k' \left(\frac{\sum_{j=1}^{kn_i} \left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i} \left(x^j - \mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2} \right) \\
&- \sum_{i=1}^{m_l} \mu^l \cdot k' \left(\frac{\sum_{j=1}^{kn_i} \left(x^j - x_i^j\right)^2}{h^2} + \frac{\sum_{j=1}^{unkn_i} \left(x^j - \mu^j\right)^2 + \left(\sigma^j\right)^2}{h^2} \right),
\end{aligned}
$$

where there are n_l points for which the x^l coordinate is known, and there are m_l points where it is missing.

$$
\begin{aligned}
f'_{x^l} &= \frac{2c_{k,d}}{nh^{d+2}} \cdot \left[\sum_{i=1}^{n} g \left(\sum_{j=1}^{d} MD_E \left(x^j, x_i^j\right)^2 \right) \right] \\
&\cdot \left[\frac{\sum_{i=1}^{n_l} x_i^l \cdot g \left(\sum_{j=1}^{d} MD_E \left(x^j, x_i^j\right)^2 \right) + \sum_{i=1}^{m_l} \mu^l \cdot g \left(\sum_{j=1}^{d} MD_E \left(x^j, x_i^j\right)^2 \right)}{\sum_{i=1}^{n} g \left(\sum_{j=1}^{d} MD_E \left(x^j, x_i^j\right)^2 \right)} - x^l \right].
\end{aligned}
$$

As a result, the mean shift vector using the MD_E distance is defined as:

$$
\begin{aligned}
&m_{MD_E,G}(x) = \\
&\frac{\sum_{i=1}^{n_l} x_i^l \cdot g \left(\sum_{j=1}^{d} MD_E \left(x^j, x_i^j\right)^2 \right) + \sum_{j=1}^{m_l} \mu^l \cdot g \left(\sum_{j=1}^{d} MD_E \left(x^j, x_i^j\right)^2 \right)}{\sum_{i=1}^{n} g \left(\sum_{j=1}^{d} MD_E \left(x^j, x_i^j\right)^2 \right)} - x^l.
\end{aligned} \tag{14}
$$

Now, we can use this equation to run the mean shift procedure over datasets with missing values.

6. Experiments on numerical datasets

In order to measure performance of the developed clustering algorithm (i.e., k-means and mean shift), we compare their performance on complete datasets to its performance on

Dataset	Dataset size	Clusters
Flame	240×2	2
Jain	373×2	2
Path-based	300×2	3
Spiral	312×2	3
Compound	399×2	6
Aggregation	788×2	7

Table 1. Speech and Image Processing Unit Dataset properties.

incomplete data using the suggested distance function and then again using the existing methods (MCA, MA, and MI) within the standard algorithms.

To measure the similarity between two data clusterings, we decide to use the Rand index [18]. We use it in order to compare the results of the original clustering algorithms to the results of the other derived algorithms for incomplete datasets.

Our experiments use six standard numerical datasets from the Speech and Image Processing Unit [13]; dataset characteristics are shown in **Table 1**.

We produced the missing data by drawing randomly a set consisting of 10–40% of the data from each dataset. These sets are used as samples of incomplete data, where one attribute from each point was randomly selected to be assigned as missing value. For each dataset, we average the results over 10 different runs.

6.1. k-Means experiments

In the k-means algorithm, we developed two versions, k-means-MD_E and k-means-$HistMD_E$; to cluster the incomplete datasets, we compare the performance of the k-means (k is fixed for each dataset) clustering algorithm on complete data (i.e., without missing values) to its performance on data with missing values, using the MD_E distance measure (k-means-MD_E and k-means-$HistMD_E$) and then again using k-means-(MCA, MA, and MI).

As can be seen in **Figure 2**, the new algorithms that is based on the MD_E distance outperformed the other existing algorithms on all the datasets. It occurred because in the MA MCA methods, the whole distribution of values is replaced by the mean or the mode of the distribution of known values, that is a fixed value. In our two developed algorithms, we use the distribution of the observed values in all the computation stages. This additional information, taking into account not only the mean of the attribute but also the variance, is probably the reason for the improved performance of our methods compared to the known heuristics.

6.2. Mean shift experiments

Mean shift clustering algorithm was tested using bandwidth $h = 4$ (because we saw that the standard mean shift worked well for this value).

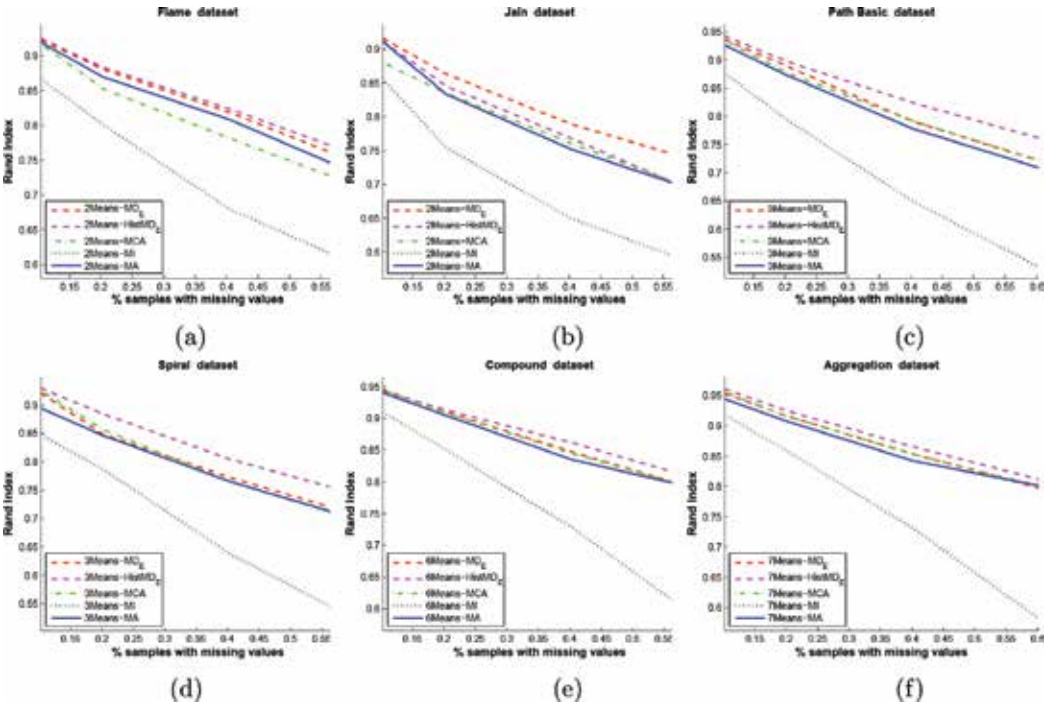

Figure 2. Results of k-means clustering algorithm using the different distance functions on the six datasets from the Speech and Image Processing Unit.

A resulting curve for the Rand index values was constructed for each dataset to evaluate how well the algorithm performed.

As can be seen in **Figure 3**, for all the datasets except the Jain dataset, the curves show that the new mean shift algorithm was superior and outperformed the other compared methods for all missing value percentages, while for the Jain dataset, its superiority became apparent only when the percent of the missing values was larger than 25%, as can be seen in **Figure 3(b)**. In addition, we can see that the $MS-MC$ method outperforms the $MS-MA$ method for the flame and path-based datasets, and the $MS-MC$ outperforms $MS-MA$ for the other datasets. As a result, we cannot decide unequivocally which algorithm is better. On the other hand, we obviously can state that the $MS-MD_E$ outperforms the other methods especially when the percentage of the missing values increases.

7. Conclusions

Missing values in data are common in real-world applications. They can be caused by human error, equipment failure, system-generated errors, and so on. Several methods were developed

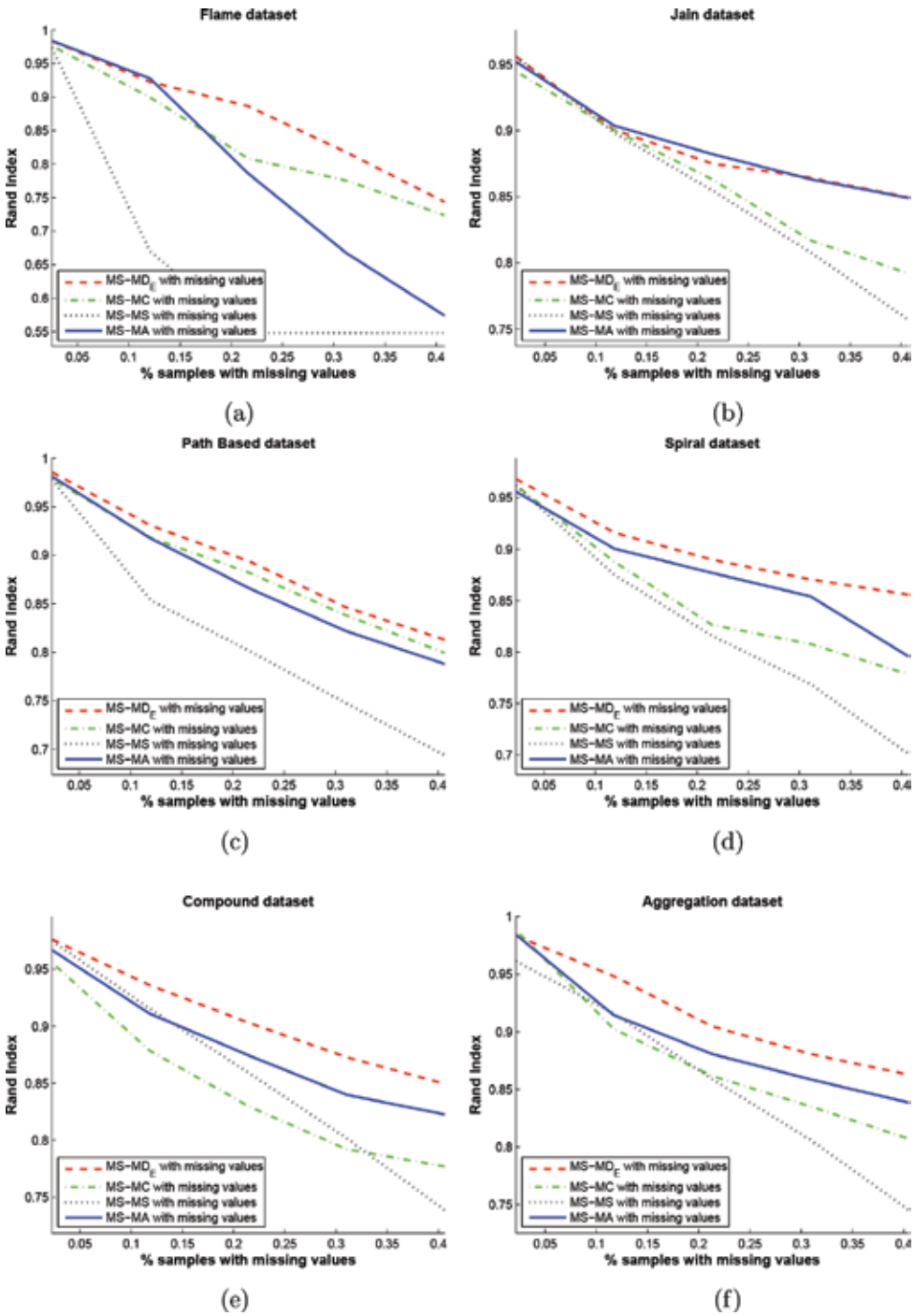

Figure 3. Results of mean shift clustering algorithm using the different distance functions on the six datasets from the Speech and Image Processing Unit.

to deal with this problem such as: filling the missing values with fixed values, ignoring sample with missing values, or dealing with the missing values by defining a distance function.

In this work, we have proposed a new mean shift clustering algorithm and two versions of the k-means clustering algorithm over incomplete datasets based on the developed MD_E distance that was presented in [1, 2, 12].

The computational complexities of all the developed algorithms were preserved and they are the same as that of the standard algorithms using the Euclidean distance. The distance was computed based only on the *mean* and *variance* of the data for each attribute.

We experimented on six standard numerical datasets from different fields. On these datasets, we simulated missing values and compared the performance of the developed algorithms using our distance and the suggested mean computations to other three basic methods.

From our experiments, we conclude that the developed methods are more appropriate for measuring the mean, mean shift vector, and weighted mean for objects with missing values, especially when the percent of missing values is large.

Author details

Loai AbdAllah[1]* and Ilan Shimshoni[2]

*Address all correspondence to: loai1984@gmail.com

1 Department of Information Systems, The Max Stern Yezreel Valley Academic College, Israel

2 Department of Information Systems, University of Haifa, Israel

References

[1] Abedallah L, Shimshoni I. K-means over incomplete datasets using mean Euclidean distance. In: Proceedings of 12th International Conference on Machine Learning and Data Mining; 2016

[2] Abedallah L, Shimshoni I. Mean shift clustering algorithm for data with missing values. In: Proceedings of 14th International Conference of DaWaK; 2014. pp. 426-438

[3] Donders ART, van der Heijden GJMG, Stijnen T, Moons KGM. Review: A gentle introduction to imputation of missing values. Journal of Clinical Epidemiology. 2006;**59**(10):1087-1091

[4] Ibrahim JG, Chen M-H, Lipsitz SR, Herring AH. Missing-data methods for generalized linear models: A comparative review. Journal of the American Statistical Association. 2005;**100**(469):332-346

[5] Little RJA. Missing-data adjustments in large surveys. Journal of Business & Economic Statistics. 1988;**6**(3):287-296

[6] Little RJA, Rubin DB. Statistical Analysis with Missing Data. Hoboken, New Jersey: John Wiley & Sons; 2014

[7] Zhang S, Qin Z, Ling CX, Sheng S. Missing is useful: Missing values in cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering. 2005;**17**(12):1689-1693

[8] Magnani M. Techniques for dealing with missing data in knowledge discovery tasks. Obtido. 2004;**15**(01):2007. http://magnanim.web.cs.unibo.it/index.html

[9] Jerzy Grzymala-Busse, Ming Hu. A comparison of several approaches to missing attribute values in data mining. In: Proceedings of Rough Sets and Current Trends in Computing; Springer; 2001. pp. 378-385

[10] Zhang S. Shell-neighbor method and its application in missing data imputation. Applied Intelligence. 2011;**35**(1):123-133

[11] Batista G, Monard MC. An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence. 2003;**17**(5–6):519-533

[12] AbdAllah L, Shimshoni I. A distance function for data with missing values and its applications on KNN and k-means algorithms. International Journal Advances in Data Analysis and Classification

[13] Speech University of Eastern Finland and Image Processing Unit. Clustering dataset, http://cs.joensuu.fi/sipu/datasets/; 2008

[14] Hunt L, Jorgensen M. Mixture model clustering for mixed data with missing information. Computational Statistics and Data Analysis. 2003;**41**(3):429-440

[15] Ghahramani Z, Jordan M. Learning from incomplete data. Technical Report, MIT AI Lab Memo, (1509), 1995

[16] Comaniciu D, Meer P. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002;**24**(5):603-619

[17] Georgescu B, Shimshoni I, Meer P. Mean shift based clustering in high dimensions: A texture classification example. In: Proceedings of the 9th International Conference on Computer Vision; 2003. pp. 456-463

[18] Rand WM. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association. 1971;**66**(336):846-850

Chapter 2

Partitional Clustering

Uğurhan Kutbay

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75836

Abstract

People are living in a world full of data. Humans are collecting data from many measurements and observations in their daily works. The sorting of these numerous data is important and necessary in terms of analyzing, reasoning, and decision-making processes. For this reason, clustering has been used in many areas and has become very important in recent years. Feature selection and classifying the data in subsets can be changed data to data. As a result of these feature selection methods, some clustering methods have been revealed. Hierarchical clustering, partitional clustering, artificial system clustering, kernel-based clustering, and sequential data clustering are determined for different clustering strategies. This chapter examines some popular partitional clustering techniques and algorithms. Partitional clustering assigns a set of data points into *k*-clusters by using iterative processes. The predefined criterion function (*J*) assigns the datum into k^{th} number set. As a result of this criterion function value in *k* sets (maximization and minimization calculation), clustering can be done. This chapter starts with criterion function for clustering process. In addition, some applications will be done for each algorithm in this chapter.

Keywords: partitional clustering, K-means, fuzzy clustering, C-FCM, genetic algorithm

1. Introduction

The choice of feature types and measurement levels depends on data type. For this reason, many clustering methods have been developed. According to clustering strategies, these methods can be classified as hierarchical clustering [1–3], partitional clustering [4, 5], artificial system clustering [6], kernel-based clustering and sequential data clustering. This chapter examines some popular partitional clustering techniques and algorithms.

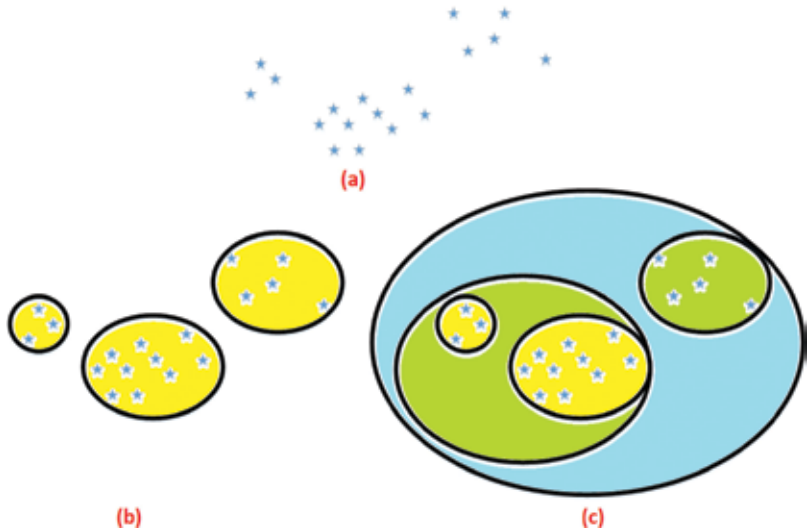

Figure 1. Clustering techniques: (a) data set; (b) partitional clustering; and (c) hierarchical clustering.

In contrast to hierarchical clustering methods, partitional clustering aims successive clusters using some iterative processes. Partitional clustering assigns a set of data points into k-clusters by using iterative processes. In these processes, n data are classified into k-clusters. The predefined criterion function J assigns the datum into k^{th} number set according to the maximization and minimization calculation in k sets.

Figure 1 represents the hierarchical clustering and partitional clustering. In addition, hierarchical clustering, all sub-clusters defined in another sub-cluster shown in **Figure 1**. **Figure 1a** represents the raw data, **Figure 1b** shows the partitional clustering and **Figure 1c** represents the hierarchical clustering. In hierarchical clustering, raw data are firstly clustered in some subgroups (three-clustered shape). After that procedure, subgroups hierarchically defined in two green clusters. Last procedure includes all these clusters which are defined in the union set.

This chapter starts with an introduction to clustering criteria and continues with K-Means algorithm, different fuzzy clustering techniques and genetic algorithm-based clustering.

2. Clustering criteria

Clustering aims to seek a partition of the data in the same homogenous clusters [7]. This homogeneity and finding the exact clustering are evaluated only by using criterion functions. One of the most popular techniques of the criterion function is the summing of squared-error (SSE) criterion and similar methods are used such as mean-square-error (MSE), normalized mean-square-error (NMSE), and so on. Sum of squared error criterion can be defined as:

$$J(\Gamma, \mathrm{M}) = \sum_{i=1}^{K} \sum_{j=1}^{N} \gamma_{ij} \left\| x_j - m_i \right\|^2 = \sum_{i=1}^{K} \sum_{j=1}^{N} \gamma_{ij} \left(x_j - m_i \right)^T \left(x_j - m_i \right) \tag{1}$$

where Γ is a partition matrix of γ_{ij} and defined as;

$$\gamma_{ij} = \begin{cases} 1 & if \;\; x_j \in cluster \;\; i \\ 0 & otherwise \end{cases} \tag{2}$$

M is the cluster prototype or centroid matrix and m_i defined as;

$m_i = \frac{1}{N_i} \sum_{i=1}^{N} \gamma_{ij} x_j$ is the sample mean for the i^{th} number cluster corresponding to N_i objects.

The partition minimizes the sum-square-error (SSE). When the SSE regarded is minimum, minimum variance partition will be achieved. As a result of this calculation, optimum cluster is determined.

3. K-means algorithm

The K-Means clusters were first developed by Mac Queen [8]. In the K-Means clusters, clusters are formed using Euclidean distance. In the K-Means algorithm, unsupervised learning is used and k classes are created which minimize the error function [9].

In K-Means clustering, k cluster centers are created from the selected data set. It is then placed at the nearest cluster using Euclidean distance. New cluster centers are formed according to the results of the clustering. From the calculations of the clustering, the cluster center is recalculated. The arithmetic average is used as the calculation method, and the new cluster center is determined. All samples are reclassified according to the new center. This process is repeated until it is determined that the samples in the set have not passed to another set.

The partitioning of the k pieces of data x is represented by the minimization of the J parameter as in (3).

$$J = \min \left(\sum_{k} \sum_{x \in c_k} w_x dist(x, o_k) \right) \tag{3}$$

If the data are classified in a cluster near the center of the nearest cluster, the J value will be the minimum. If the data x are classified in the k^{th} number cluster, the value can be optimized by changing the weighting value of w_x to obtain the minimum J value. $dist(x, o_k)$ is the notation which represents the distance function. In this formula, x represents the pixel data, o_k is the center of the cluster. k sets are shown as in (4).

K-Means Algorithm Flow Chart

1: while dist(.)<δ
2: o_k will be updated.
3: create k-clusters
4: calculate dist(.)
5: minimize $J = \min(\sum_{x} \sum_{x \in o_k} w_x dist(x, o_k))$
6: assign x data to the cluster with the minimum J
7: end
Note: δ is the given threshold value.

Figure 2. Flow chart of the K-Means algorithm.

$$c_k = (c_{k,1}, c_{k,2}, \ldots, c_{k,n}) \tag{4}$$

The distance function can be calculated as in Euclidean distance for k clusters in the next formula (5).

$$dist\left(x_j^i, o_k\right) = \sqrt{\left(x_{j,1}^i - o_{k,1}\right)^2 + \left(x_{j,1}^i - o_{k,2}\right)^2 + \ldots + \left(x_{j,n}^i - o_{k,n}\right)^2} \tag{5}$$

The iterative operations are repeated as in the flow diagram in **Figure 2**. This flowchart represents the iterative distance controlled operations to minimize the J parameter.

When this algorithm is applied for three-cluster for the original image shown in **Figure 3a**, this image can be represented in three clusters as in **Figure 3b–d**, respectively. Cluster-1 shows the

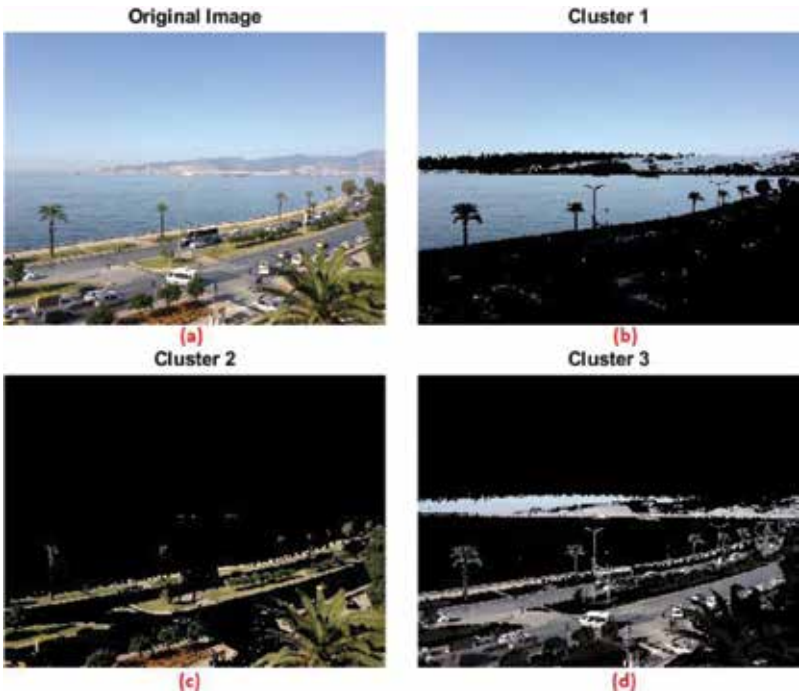

Figure 3. K-means algorithm results. (a) Original image; (b) K-Means Cluster No:1; (c) K-Means Cluster No:2; and (d) K-Means Cluster No:3.

sea for the given landscape, cluster-2 shows the the green areas in the landscape and cluster-3 shows the roads for the given landscape.

4. Fuzzy clustering

Fuzzy theory is firstly developed by Zadeh [10] for defining adjustable degrees of memberships. Fuzzy theory creates intermediate sets rather than classical sets. In classical sets, each data item is assigned into only one cluster. In contrast, data in fuzzy clusters can be represented in multiple clusters. This multiset assignment can belong to all the clusters with a certain degree of membership defined by Bezdek [11]. This one item in multiset representation can be useful for sharply separated cluster boundaries.

The fuzzy C-Mean algorithm (FCM) is frequently used because of its ease of operation and reliability in many applications [12–15]. Conventional Fuzzy C-Mean (FCM) works with the principle of minimizing the objective function [16] shown in the following formula (6):

$$J_{FCM} = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} d_{ik}^{2} \tag{6}$$

where u_{ik}^{m} is the membership function in the range [0,1]. This membership represents the membership degree of x_k for the k^{th} pixel. k is defined in the range of $k \in [1, n]$ for the i^{th} numbered cluster. Total cluster size (c) given in the range of $[1, c]$. In the formula (7), d_{ik}^{2} represents the distance between x_{k} and i^{th}-cluster center (v_{i}).

$$d_{ik}^{2} = \|x_k - v_i\| \tag{7}$$

The fuzzy set theory aims that the membership function is $\sum_{i=1}^{c} u_{ik} = 1$ for each pixel. Using the FCM membership function u_{ik}^{m} and cluster center v_i, FCM targets to reach local minimums by using the equivalence (8) and (9), respectively.

$$u_{ik}^{*} = \left\{ \sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}} \right)^{2/(m-1)} \right\}^{-1} \quad \forall i \in [1, c] \;\; and \;\; k \in [1, n] \tag{8}$$

$$v_i^{*} = \frac{\sum_{k=1}^{n} u_{ik}^{m} x_k}{\sum_{k=1}^{n} u_{ik}^{m}} \;\; \forall i \in [1, c] \tag{9}$$

FCM algorithm flow chart is shown in **Figure 4**. This algorithm will be done in the given iteration step specified in this traditional clustering method. The cluster centers are updated in each iteration step to calculate the membership function.

When the FCM algorithm is applied for the given original image shown in **Figure 5a** for three clusters, this image can be represented in three clusters as in **Figure 5b–d**, respectively. This algorithm can only be applied for gray-scaled images. There is no crisp clustering for the given

FCM Algorithm Flow Chart

Obligation: Fuzziness factor $m>1$, for each cluster C

1: Determine the cluster center randomly

2: while($d_{ik}^2<\delta$)

3: Calculate u_{ik}^m for each cluster center

4: $u_{ik}^* = \left\{ \sum_{j=1}^{c} \left(\frac{d(x_k c_i^{(m-1)})}{x_k c_j^{(m-1)}} \right) \right\}^{-1}$

5: Update cluster centers c_i^m by using $v_i^m = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m}$

6: end

Note: v_i **is the updated cluster center,** u_{ik} **is the updated membership degree**

Note: δ **is the given threshold value.**

Figure 4. Flow chart of the FCM algorithm.

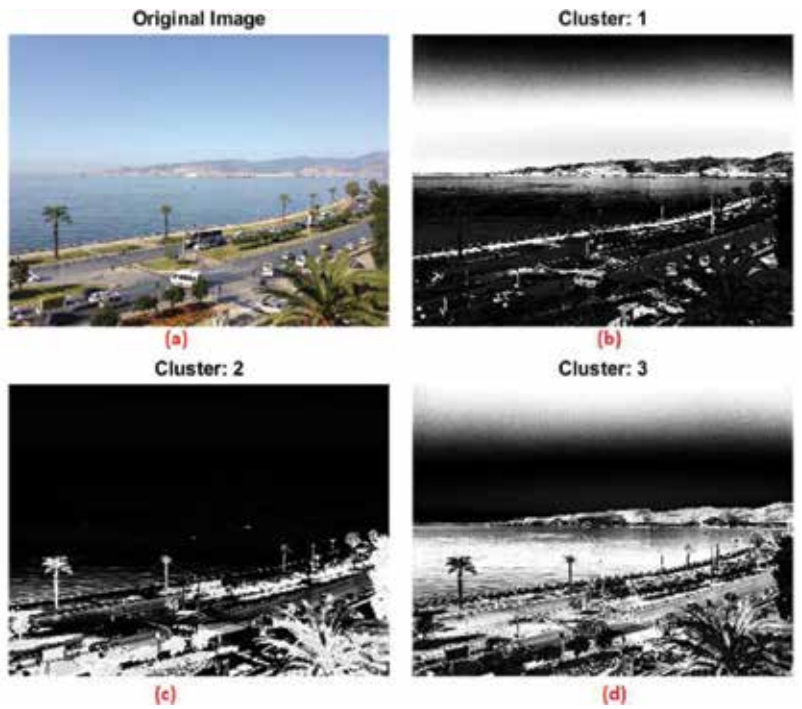

Figure 5. FCM algorithm results. (a) Original image; (b) FCM Cluster No:1; (c) FCM Cluster No:2; and (d) FCM Cluster No:3.

landscape image. This image includes many objects in different colors, and these colors are generally indented in these objects.

FCM algorithm includes iterative procecesses. For the given image at the end of the 55 iteration steps, clustering processes are completed for the given threshold value which is 10^{-5} for FCM distance change.

5. Colored FCM

Colored Image Fuzzy C-Mean (C-FCM) involves color-based clustering using fuzzy sets. This 3D method is firstly given by Kutbay and Hardalaç [17] as Robust Colored Image FCM (RCI-FCM), but this presented method is different from RCI-FCM. In RCI-FCM, distances are calculated for each R, G and B channels, but in this method, the mean distance is calculated for RGB color spaces. This method represents the RGB color formed images in FCM, which uses FCM-based algorithm in colored images. The membership function calculates the centroids of the clusters for each R, G and B color spaces. After calculation of Euclidian distance for each channel, the mean distance could be calculated shown in equivalence (10).

$$d^2_{ik(R,G,B)} = mean\left(\left\|x_{k(R)} - v_{i(R)}\right\|, \left\|x_{k(G)} - v_{i(G)}\right\|, \left\|x_{k(B)} - v_{i(B)}\right\|\right) \tag{10}$$

Membership for each cluster can be calculated in the following formula (11).

$$u^*_{ik(R,G,B)} = \left\{ \sum_{j=1}^{c} \left(\frac{d_{ik(R,G,B)}}{d_{jk(R,G,B)}} \right)^{2/(m-1)} \right\}^{-1} \quad \forall i \in [1,c] \;\; and \;\; k \in [1,n] \tag{11}$$

where $u^*_{ik(R,G,B)}$ represents the membership degree for mean distance for each color. In this representation for each item, cluster's membership can be calculated and statistically membership could be defined for each cluster.

New cluster center can be calculated in the following equivalence (12) for each cluster:

$$v^*_{i(R,G,B)} = \frac{\sum_{k=1}^{n} u^m_{ik(R,G,B)} x_{k(R,G,B)}}{\sum_{k=1}^{n} u^m_{ik(R,G,B)}} \quad \forall i \in [1,c] \tag{12}$$

where $u^*_{ik(R,G,B)}$ is the membership function of each RGB color pixel. This function calculates for each R, G and B colors, for the each RGB space representation. c denotes the clusters for each n-pixel, and $d_{ik(R,G,B)}$ explains the distance between pixel x_k and the centroid v_i for i^{th} number cluster for each color domain. $v^*_{i(R,G,B)}$ represents the centers of the clusters for each RGB pixel value into the c-cluster. C-FCM algorithm's flow chart is shown in **Figure 6**. The proposed method aims to create c^3 cluster for the given stopping criteria. New cluster centroids are calculated from the results of the membership function for all RGB colors.

C-FCM algorithm flow chart is shown in **Figure 6**. The flowchart shows the C-FCM algorithm for colored images. For each iteration step, RGB distances are calculated and cluster centers are calculated. After updating the cluster centers in the given threshold value, each item will be assigned into the given cluster.

Figure 7 represents the C-FCM algorithm, which is applied for the given original image shown in **Figure 7a**. For each color, pixels assigned into two cluster. For RGB color space 2^3 cluster are created for this image. These clusters can be given in eight clusters as in **Figure 4b–i**, respectively.

C-FCM Algorithm Flow Chart
Obligation: Fuzziness factor $m>1$, for each cluster C
1: Initialize the cluster center randomly for each color RGB
2: while($d_{ik}^2<\delta$)
3: Calculate $d_{ik(R,G,B)}^2$
3: Calculate u_{ik}^m for each cluster center
4: $u_{ik}^* = \left\{ \sum_{j=1}^{c} \left(\frac{d(x_k)}{x_k c_j^{(m-1)}} \right) \right\}^{-1}$
5: Update cluster centers c_i^m by using $v_i^m = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m}$
6: end
Note: v_i is the updated cluster center, u_{ik} is the updated membership degree
Note: δ is the given threshold value.

Figure 6. Flow chart of the C-FCM algorithm.

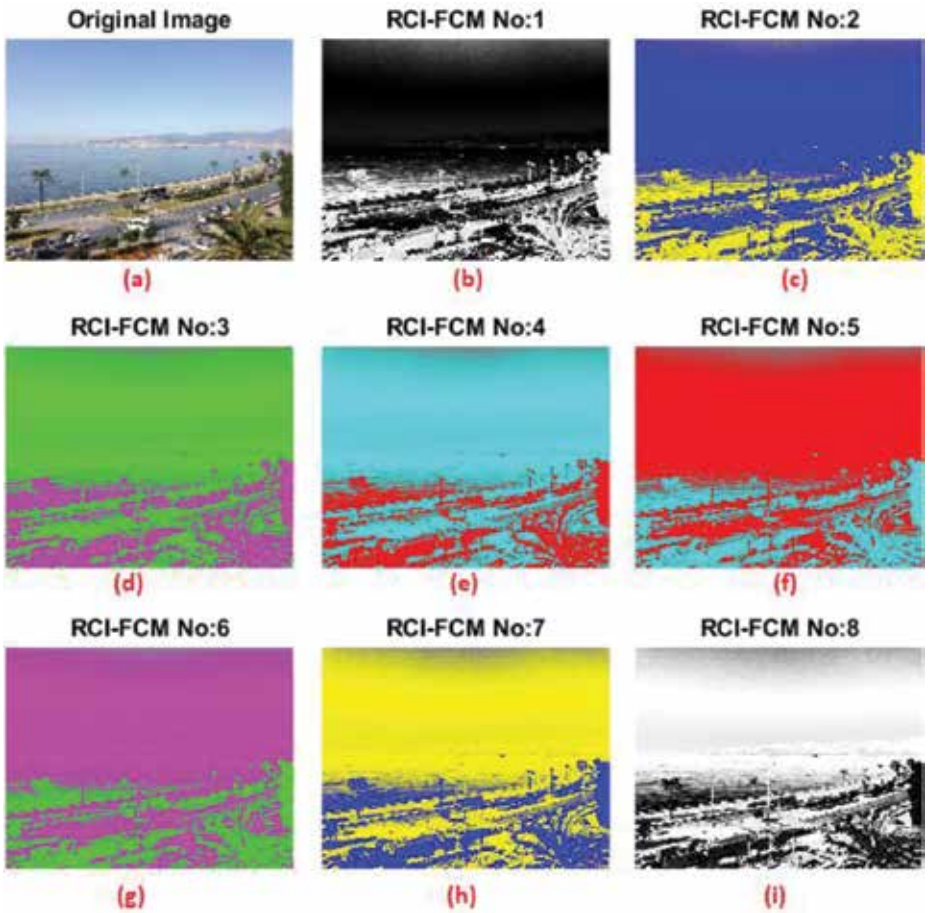

Figure 7. C-FCM algorithm results. (a)Original image; (b)–(i). C-FCM Cluster No:1–8.

C-FCM algorithm includes iterative procecesses. For the given image at the end of the 26 iteration steps, clustering processes are completed for the given threshold value which is 10^{-5} for C-FCM distance change.

6. Genetic clustering

Genetic algorithm is very popular method in evolutionary computation processes. This method is firstly developed by Holland in 1975 [18]. This algorithm includes natural evolutionary processes. This method optimizes a population of the structure by using a set of evolutionary operators.

This method maintains a population of structures and these structures consisting of individuals. Each individual is evaluated by a function named as fitness function. These processes includes selection, recombination and mutation processes.

In genetic algorithms (GAs), each individual represents a candidate solution in binary form. This individual called as chromosome. After an initial population is generated, randomly crossover and mutation processes are executed for each iteration step.

For genetic algorithm examination, the following terms are useful for describing the concept of genetic algorithms. These are gene, chromosome, population (mass), reproduction process, and conformity value.

Gen is a unit that carries partial information. By bringing together these units, the chromosomal sequence that forms the solution cluster comes into play; for this reason, the genome decides well how to code it.

Chromosomes are structures that contain the information of the problem solving. Population is formed by the combination of chromosomes. At the initiative of the designer, what information is to be found on the chromosome.

The population is called the heap of possible solutions. In the GA process, the population size remains constant, but the bad chromosomes separate from the stack. The size of the heap is a very important concept, which must be well established, as the overcrowded heap increases the time of possible heuristic approach, while the small heap may cause no possible solution at all.

The reproduction process is the process of selecting the sequences to be transferred from the current stack to the next stack. The sequences carried are genetically the most appropriate sequences. The requirement for transition is whether the level of conformity specified has been achieved.

The fitness value, in genetic algorithms, which specifies which index will transfer the next generation, which index will be lost. Conformity value reflects the purpose of the problem.

Bandyopadhyay and Maulik [19] attempted to use GA to automatically determine the number of clusters K in 2002. The GA clustering aims assigning the data into k^{th} cluster using genetic processes.

Showing the basic concept of the GA-based clustering, genetic guided algorithm [20] is used. Fitness value is represented in fuzzy clusters as shown in equivalent (13);

$$j(M) = \sum_{j=1}^{N} \left(\sum_{i=1}^{K} D_{ij}^{1/(1-m)} \right)^{1-m} \tag{13}$$

where $D_{ij} i \in [1, K]$ and $j \in [1, N]$. D_{ij} represent the distance between the i^{th} cluster prototype vector and the data object of the number of *j*. m represents the fuzzification coefficient. Calculation process of the genetic algorithm is shown in **Figure 8**.

In this genetic parameter optimized algorithm, firstly P individuals are initialized, and each individual represents *K x d* prototype matrix encoded as gray codes. After this prototype matrix representation, fitness values are calculated. After this calculated fitness value, tournament selection is used for parental member reproduction. For generating new parents, two-point crossover and bitwise mutation are done. For these new individuals, the highest fitness values are obtained using this fitness equation. All these processes are applied until termination condition ($D_{ij} < \delta$) is satisfied.

Chromosomes are randomly selected, and best parent are selected in tournament selection process. After tournament selection process, two-point crosover process and bitwise mutuation are done, respectively. These processes are shown in **Figures 9** and **10**, respectively.

Figure 9 represents crossover process. In two-point crossover process, bitwise crossover points are determined. After determination process, crossover process will be done.

There are some crossover processes are used in different works [21–23]. These types are 1-point crossover, K-point crossover, shuffle crosover, and so on [24].

Mutation process helps the genetically variations from parent chromosomes to child chromosomes. There are many mutation operators defined in the literature. These are bit-flip-mutation, swap-mutation and inversion-mutation [25]. In mutation process, mutation points are given. In this example, three mutation points are determined shown in **Figure 10**. After mutation process, mutation points will be changed. If the mutation rate is high, the main

Genetic Optimized Fuzzy Clustering Flow Chart
1: Initialize the population randomly for P individuals
2: *P*(i)=K x d prototype gray scaled matrix coded as binary
3:while(D_{ij}< δ)
3: Calculate fitness function $j(M) = \sum_{j=1}^{N} \left(\sum_{i=1}^{K} D_{ij}^{1/(1-m)} \right)^{1-m}$
3: Use a selection operator (roulette selection) to choose parental member
4: Two point crossover and bitwise mutation are done
5: Determine the next generation by using the highest fitness
6: end
Note: δ is the given threshold value.

Figure 8. Flow chart of the genetic clustering.

http://dx.doi.org/10.5772/intechopen.75836

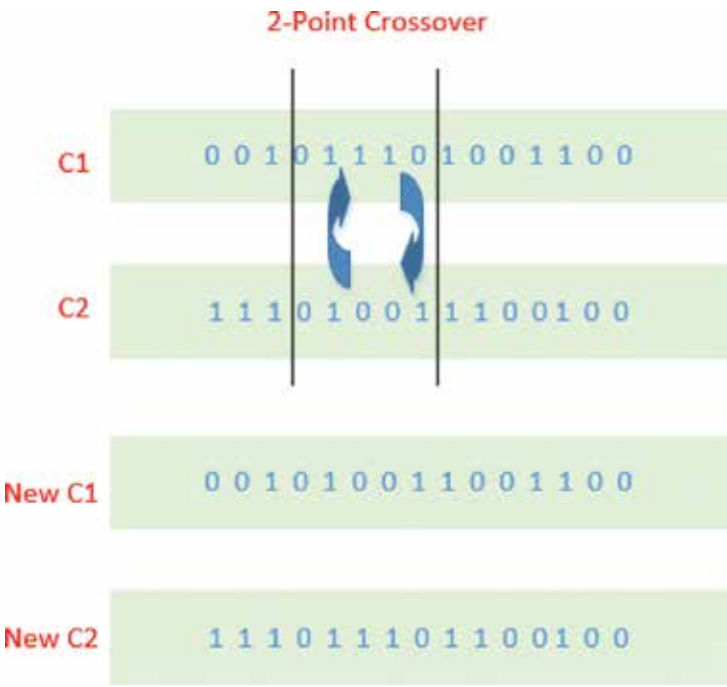

Figure 9. Two-point crossover process.

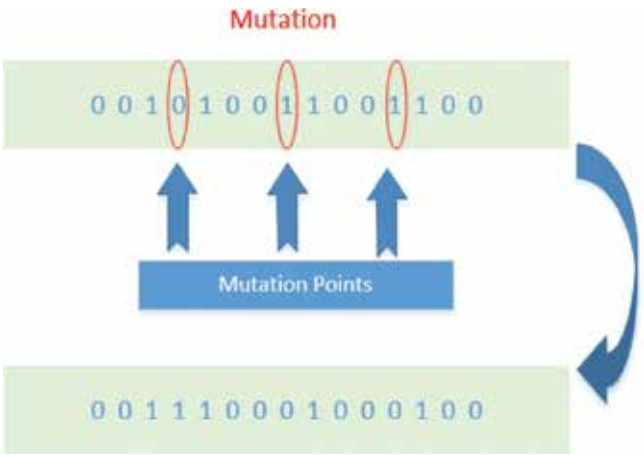

Figure 10. Mutation process.

generation may be lost. In general terms, the rate of mutation in GA is between given 0.05 and 0.15% of the parent.

GA-based fuzzy clustering process in four clusters shown in **Figure 11**. GA is applied for the given original image shown in **Figure 11a** for four clusters. This image can be represented in

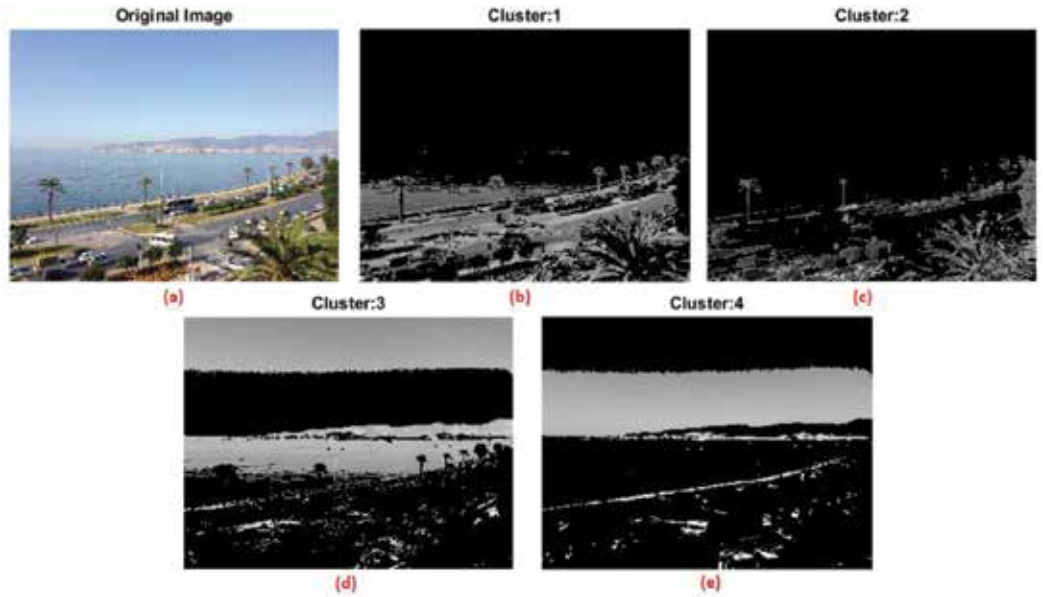

Figure 11. Genetic algorithm results. (a) Original image, (b) Cluster No:1, (c) Cluster No:2, (d) Cluster No:3, and (e) Cluster No:4.

four clusters as in **Figure 11b–e**, respectively. This algorithm can only be applied for gray-scaled images. Cluster-1 shows the roads for the given landscape, cluster-2 shows the green areas in the landscape, cluster-3 shows the sea for the given landscape and cluster-4 represents the air for the given landscape.

7. Conclusion

In this part, we discussed the fundamental partitional clustering algorithms and its applications. Partitional clustering produces a partition using *K* clusters but do not use hierarchical structures. This type of clustering process uses criterion function in a range of error such as mean square error, sum of squared error, and so on. For this reason, partitional clustering is an iterative process.

Some search techniques which include evolutionary process, provide seeking the global or local optima. Search techniques introduce more parameters than K-Means clustering process. There is no theoretical approval for selecting of the best approach. All the given methods can give the best results for different data.

The partitional clustering algorithms need high computational requirements. High computational requirements means that some complex algorithms need advanced operations. These requirements limit their applications in large data. Overcoming this situation, high speed and high-capacity memory-sized computers can be used for clustering processes.

Another method for overcoming high computational requirements is the genetic optimization. For large datasets, genetic optimization should be used with fuzzy clustering. In this technique, all individuals or pixels are genetically optimized so as to fuzzy sets.

Clusters cannot be always sharply separated in some datasets. For this type of data sets, fuzzy-based clustering overcomes this crisp clustering. Fuzzy clustering allows to find the nearest optimum cluster to assign into the clusters. This technique provides more information about the data structure.

The most important point of the search techniques of the partitional clustering is the optimum parameter selection. Parameter selection is an optimization problem. Overcoming this optimization problem, parameter selection can be done by using genetic algorithms. Genetic algorithms can be useful solution for very-large scale data sets.

In partitional clustering, determination of cluster size is important. This selection differs from the data sets to data sets. If data set includes more features to classify in a cluster, more clusters will be needed. This cluster number unfortunately is not known for many clustering problems. Generally experiences give the cluster number. Estimation of the cluster number is one of the major problems for validation.

As a result, by using partitional clustering techniques, image understanding can be done by using the given techniques. For many applications such as biomedical image understanding, robotic applications and security applications, these techniques can be useful for pre-processing of some algorithms.

Acknowledgements

This study was performed at Gazi University, Engineering Faculty, Electrical & Electronics Engineering Department. MATLAB® 2016b software platform was used for this study. The study was performed in a computer with 12 GB RAM, Intel i5 processor at 2.6 Ghz.

Competing interests

The author declares having no conflicts of interests.

Thanks

I would like to thank my wife Hande Kutbay who has given me a lot of support in my life. I thank my daughter, Gökçe Kutbay, who makes life difficult but makes sense. Special thanks to my mother Cemile KUTBAY and my father Hür Uğur KUTBAY for being in my life. In addition, I would like to thank Assoc. Prof. Dr. Fırat Hardalaç due to academic help.

Author details

Uğurhan Kutbay

Address all correspondence to: ukutbay@gazi.edu.tr

Electrical and Electronics Engineering Department, Gazi University Engineering, Ankara, Turkey

References

[1] Liu AA, Su YT, Nie WZ, Kankanhalli M. Hierarchical clustering multi-task learning for joint human action grouping and recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017;**39**(1):102-114

[2] Bouguettaya A, Yu Q, Liu X, Zhou X, Song A. Efficient agglomerative hierarchical clustering. Expert Systems with Applications. 2015;**42**(5):2785-2797

[3] Cao K, Jiao L, Liu Y, Liu H, Wang Y, Yuan H. Ultra-high capacity lithium-ion batteries with hierarchical CoO nanowire clusters as binder free electrodes. Advanced Functional Materials. 2015;**25**(7):1082-1089

[4] Nanda SJ, Panda G. A survey on nature inspired metaheuristic algorithms for partitional clustering. Swarm and Evolutionary Computation. 2014;**16**:1-18

[5] Menendez HD, Barrero DF, Camacho D. A genetic graph-based approach for partitional clustering. International Journal of Neural Systems. 2014;**24**(03):1430008

[6] Pandeeswari N, Kumar G. Anomaly detection system in cloud environment using fuzzy clustering based ANN. Mobile Networks and Applications; **21**(3):494-505

[7] Xu R, Wunsch D. Clustering 2010. Vol. 10. John Wiley & Sons; 2008. pp. 63-110

[8] Mac-Queen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability; Vol. 1: Statistics. California, USA; 1967. pp. 281-297

[9] Kutbay U, Ural AB, Hardalaç F. Underground electrical profile clustering using K-MEANS algorithm. In: 2015 23th IEEE Signal Processing and Communications Applications Conference (SIU); 2015. pp. 561-564

[10] Zadeh L. Fuzzy sets. Information and Control. 1965;**8**:338-353

[11] Bezdek J. Cluster validity with fuzzy sets. Journal of Cybernetics. 1974;**3**(3):58-72

[12] Abdulshahed AM, Longstaff AP, Fletcher S, Myers A. Thermal error modelling of machine tools based on ANFIS with fuzzy c-means clustering using a thermal imaging camera. Applied Mathematical Modelling. 2015;**39**(7):1837-1852

[13] Ji Z, Liu J, Cao G, Sun Q, Chen Q. Robust spatially constrained fuzzy c-means algorithm for brain MR image segmentation. Pattern Recognition. 2014;**47**(7):2454-2466

[14] Qiu C, Xiao J, Yu L, Han L, Iqbal MN. A modified interval type-2 fuzzy C-means algorithm with application in MR image segmentation. Pattern Recognition Letters. 2013;**34**(12): 1329-1338

[15] Kannan SR, Ramathilagam S, Devi R, Hines E. Strong fuzzy c-means in medical image data analysis. Journal of Systems and Software. 2012;**85**(11):2425-2438

[16] Ji Z, Xia Y, Chen Q, Sun Q, Xia D, Feng DD. Fuzzy c-means clustering with weighted image patch for image segmentation. Applied Soft Computing. 2012;**12**(6):1659-1667

[17] Kutbay U, Hardalaç F. Development of a multiprobe electrical resistivity tomography prototype system and robust underground clustering. Expert Systems. 2017;**34**(3):e12206

[18] Holland JH. Adaptation in Natural and Artificial Systems. An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence. Ann Arbor, MI: University of Michigan Press; 1975

[19] Bandyopadhyay S, Maulik U. Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition. 2002;**35**(6):1197-1208

[20] Hall LO, Ozyurt IB, Bezdek JC. Clustering with a genetically optimized approach. IEEE Transactions on Evolutionary Computation. 1999;**3**(2):103-112

[21] De Jong KA, Spears WM. A formal analysis of the role of multi-point crossover in genetic algorithms. Annals of Mathematics and Artificial Intelligence. 1992;**5**(1):1-26

[22] Whitley D. An executable model of a simple genetic algorithm. Foundations of Genetic Algorithms. 2014;**2**(1519):45-62

[23] Yu F, Xu X. A short-term load forecasting model of natural gas based on optimized genetic algorithm and improved BP neural network. Applied Energy. 2014;**134**:102-113

[24] Umbarkar AJ, Sheth PD. Crossover operators in genetic algorithms: A review. ICTACT Journal on Soft Computing. 2015;**6**(1):1083-1092

[25] Larranaga P, Kuijpers CM, Murga RH, Inza I, Dizdarevic S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review. 1999;**13**(2):129-170

Chapter 3

Incorporating Local Data and KL Membership Divergence into Hard C-Means Clustering for Fuzzy and Noise-Robust Data Segmentation

Reda R. Gharieb

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74514

Abstract

Hard C-means (HCM) and fuzzy C-means (FCM) algorithms are among the most popular ones for data clustering including image data. The HCM algorithm offers each data entity with a cluster membership of 0 or 1. This implies that the entity will be assigned to only one cluster. On the contrary, the FCM algorithm provides an entity with a membership value between 0 and 1, which means that the entity may belong to all clusters but with different membership values. The main disadvantage of both HCM and FCM algorithms is that they cluster an entity based on only its self-features and do not incorporate the influence of the entity's neighborhoods, which makes clustering prone to additive noise. In this chapter, Kullback-Leibler (KL) membership divergence is incorporated into the HCM for image data clustering. This HCM-KL-based clustering algorithm provides two-fold advantage. The first one is that it offers a fuzzification approach to the HCM clustering algorithm. The second one is that by incorporating a local spatial membership function into the HCM objective function, additive noise can be tolerated. Also spatial data is incorporated for more noise-robust clustering.

Keywords: data science, clustering, image clustering, hard and fuzzy C-means, membership function, Kullback-Leibler (KL) divergence

1. Introduction

Image segmentation is a principle process in many image, video, scene analysis and computer vision applications [1–3]. The objective of segmentation process is to divide an image into multiple separate regions or clusters which make it easier to recognize and distinguish different objects in image. Over the last few decades, several image segmentation methods have

been developed. However, there is still no satisfactory performance especially in noisy images. This makes development of segmentation algorithms that are capable of handling noisy images an active area of research. The current segmentation methods can be classified into thresholding, region-detection, edge-detection, probabilistic and artificial neural-network classification and clustering [1–3]. Among the widely used are the hard and fuzzy-based clustering methods since clustering needs no training examples [4–24]. Hard C-means (HCM) also called K-means clustering algorithm is an unsupervised approach in which data is basically partitioned based on locations and distances between various data points [4–6]. K-means partitions the data into C-clusters so that the distances between data within each cluster are as close as possible but as far as possible between data in different clusters. HCM clustering algorithm offers crisp segmentation in which each data point belongs to only one cluster. Thereby it does not take into consideration fine details of infrastructure of data such as hybridization or mixing. Compared with HCM algorithm, fuzzy C-means (FCM) algorithm is able to provide soft segmentation by incorporating membership of belonging described by a membership function [7, 8]. However, one disadvantage of the standard FCM is not incorporating any spatial or local information in image context, making it very sensitive to additive noise and other imaging artifacts. To handle this problem, different techniques have been developed [9–13]. These techniques have involved spatial or local data information for the enhancement and regularization of the performance of the standard FCM algorithm. Local membership information has also been employed to generate a parameter to weight or modify the membership function in order to give more weight to the pixel membership if the immediate neighborhood pixels are of the same cluster [14]. HCM algorithm has also been fuzzified by involving membership entropy optimization [15–17].

In this chapter, HCM clustering algorithm is modified by incorporating local spatial data and Kullback-Leibler (KL) membership divergence [18–22]. The local data information is incorporated via an additional weighted HCM function in which the smoothed image data is used for the distance computation. The KL membership divergence aims at minimizing the information distance between the membership function of each pixel and the locally smoothed one in the pixel vicinity. The KL membership divergence thus provides an approach for regularization and fuzzification. The chapter is organized as follows. In Section 2, clustering problem formulation is overviewed. In Section 3, HCM clustering algorithm is described. In Section 4, several FCM-related clustering algorithms are explained. In Sections 5 and 6, the proposed local membership KL divergence-based FCM (LMKLFCM) and Local Data and membership KL divergence-based FCM (LDMKLFCM) clustering algorithm are discussed. In Section 7, simulation results of clustering and segmentation of synthetic and real-world images are presented. Finally Section 8 presents the conclusion.

2. Problem formulation

The objective is to cluster a set of observed data $\{x_n; n = 1, 2, .., N\}$ where each data point is an $M-$ dimensional real-vector called the feature or the pattern vector, i.e., $x_n \in \mathrm{R}^{1\times M}$. For gray-scale image data, $\{x_n; n = 1, 2, .., N\}$ is a row-wise concatenation of a 2-D image

$\{X_{pq}; p = 1, 2, .., P; q = 1, 2, .., Q\}$. That is $n = (p-1)Q + q$ and the intensity-feature x_n is a single-dimensional real-value, i.e., $M = 1$. Clustering aims at partitioning theses N observations into $C < N$ divisions, $\{\mu_1, \mu_2, \ldots, \mu_C\}$ called C clusters or segments so as to make the entities or pixels in the same cluster as similar as possible and the ones in different clusters as dissimilar as possible. One approach to cluster these data is to minimize the within-clusters sum of squares of distances (WCSS) and to maximize the between-clusters sum of squares of distances (BCSS).

3. Hard C-means (HCM)

In hard C-means (HCM) algorithm also called the K-means one, the objective is to minimize the following function [4–6, 15].

$$J_{HCM} = \sum_{i=1}^{C} \sum_{n=1}^{N} u_{in} d_{in} \tag{1}$$

where $d_{in} = \|x_n - v_i\|^2$, is the square of the Euclidian distance between the nth pixel feature x_n of the image under segmentation and $v_i \in V = \{v_1, v_2, \ldots, v_C\}$ called the center of the ith cluster given by

$$v_i = \frac{\sum_{x_n \in \mu_i} x_n}{N_i}, \quad i = 1, 2, \ldots, C. \tag{2}$$

where μ_i is the ith cluster label and N_i is its number of patterns in cluster i. In (2), it is clear that the pattern x_n belongs to only one cluster which means that $u_{in} \in \{0, 1\}$ called the membership function is given by [15].

$$u_{kn} = \begin{cases} 1; \ k = \arg\min_i(d_{in}) \\ 0, \textit{Otherwise} \end{cases} \tag{3}$$

From (3), it is obvious that the HCM provides a crisp membership function $u_{in} \in \{0, 1\}$ or {False, True}. $u_{in} \in \{0, 1\}$. Thus HCM algorithm does not take into account fine details of infrastructure

Given x_n, $n = 1, 2, \ldots, N$.
Initialize v_i^0, $i = 1, 2, .., C; t = 0$;

1. For $n = 1, 2, \ldots, N$
Compute:

2. $d_{in} = \left\|x_n - v_i^t\right\|^2; i = 1, 2, \ldots, C.$

3. $k = \arg\min_i(d_{in})$; u_{kn}=1; $u_{in} = 0; i = 1, 2, .., C; i \neq k$. (HCM);
$u_{in} = \frac{1}{\sum_{j=1}^{C} \left(\frac{d_{in}}{d_{jn}}\right)^{\frac{1}{(m-1)}}}$ (FCM)

4. Update $t = t + 1; v_i^{t+1} = \frac{\sum_n u_{in} x_n}{\sum_n u_{in}}, \quad i = 1, 2, \ldots, C.$

5. Check if $\left\|V^t - V^{t+1}\right\|^2 > \varepsilon$ (negligible change); repeat 1–5 until convergence.

Table 1. Pseudo code of the HCM (FCM) algorithms.

such as hybridization or mixing of data which is important in data clustering and decision making. The algorithm is implemented by an iterative procedure as summarized in **Table 1**.

4. Related fuzzy clustering algorithms

4.1. Conventional FCM

The fuzzy C-means (FCM) algorithm seeks to minimize the following objective function [7].

$$J_{FCM} = \sum_{i=1}^{C} \sum_{n=1}^{N} u_{in}^{m} d_{in} \tag{4}$$

It is obvious that the difference between the FCM algorithm and HCM one is the incorporation of the exponent parameter m, called the fuzzification parameter, and if it is settled to 1, the FCM algorithm reduces to the HCM one. Thus, due to this exponent m, the membership of the nth pixel to the ith cluster, u_{in}, can take on an infinite set of values from 0 to 1. Thus each nth pixel may belong to all clusters with equal membership values of $1/C$ which in this case we obtain too fuzzy membership function. Then the exponent parameter $1 < m$ is incorporated to control the degrees of fuzzification; the bigger the m, the more the fuzzification. Finally, the fuzzy membership u_{in} should satisfy [7].

$$u_{in} \in U = \left\{ u_{in} \in [0,1]; \sum_{i=1}^{C} u_{in} = 1 \forall n; 0 < \sum_{n=1}^{N} u_{in} < N \forall i \right\}, \tag{5}$$

The membership u_{in} and the cluster-center v_i that minimize the FCM function in (4), subject to $\sum_{i=1}^{C} u_{in} = 1 \forall n$ are given by [7].

$$u_{in} = \frac{1}{\sum_{j=1}^{C} \left(\frac{d_{in}}{d_{jn}} \right)^{\frac{1}{(m-1)}}} \tag{6}$$

$$v_i = \frac{\sum_{n=1}^{N} u_{in} x_n}{\sum_{n=1}^{N} u_{in}} \tag{7}$$

4.2. Local spatial data-based FCM (LDFCM)

The neighboring pixels of an image are highly correlated and are thus highly expected to belong to the same cluster or object. To get benefit from this spatial data information, the standard FCM objective function in (4) has been modified by adding a weighted regularization function dependent on local image data information [10–12]. That is, the LDFCM objective function is given by

$$J_{LDFCM} = J_{FCM} + \alpha \sum_{i=1}^{C} \sum_{n=1}^{N} u_{in}^{m} \overline{d_{in}} \tag{8}$$

where α is a weight to be experimentally selected by the user, m is a fuzzification parameter, $\overline{d}_{in} = \|\overline{x}_n - v_i\|^2, \overline{x}_n \in \overline{X}$ is the nth pixel of the locally-smoothed image, $\overline{X}$, obtained in advance

from the original one by $\overline{X} = w^{(x)} ** X,$ where ** means two-dimensional convolution. The weights $w^{(x)}$ can be equal or not provided that its centerweight is zero and are summed to unity. From (8), it is clear that the LDFCM aims at minimizing the standard FCM objective function plus another weighted modified FCM function acting as a regularization function. In this regularization FCM function, the distances are generated from the locally-smoothed image data instead of the original image data. Therefore, this correlates the clustering pixel x_n with its immediate spatial neighboring pixels which biases the algorithm to provide clustered images with piecewise homogenous regions. The membership u_{in} and the cluster-center v_i functions of the LDFCM method are given by [10–12].

$$u_{in} = \frac{1}{\sum_{j=1}^{C} \left(\frac{d_{in} + \alpha \bar{d}_{in}}{d_{jn} + \alpha \bar{d}_{jn}} \right)^{\frac{1}{(m-1)}}} \tag{9}$$

$$v_i = \frac{\sum_{n=1}^{N} u_{in}(x_n + \alpha \overline{x}_n)}{(1+\alpha)\sum_{n=1}^{N} u_{in}} \tag{10}$$

It is obvious from (9) and (10) that when $\alpha = 0,$ the membership u_{in} and the cluster-center v_i become the ones provided by the standard FCM algorithm in (6) and (7). The advantage of the LDFCM method arises from involving the locally-smoothed data $\alpha \overline{x}_n$ in computing the membership u_{in} and the cluster-center v_i functions which indeed can handle additive noise.

4.3. Spatial-based fuzzy C-means (SFCM)

An approach to incorporating local spatial data information into the standard FCM has been presented in [13]. The objective function of the SFCM algorithm is given by

$$J_{SFCM} = \sum_{i=1}^{C} \sum_{n=1}^{N} u_{in}^{m} D_{in}, \tag{11}$$

where D_{in} is a modified or weighted distance between the nth pixel and the ith cluster-center. This modified distance is computed from the original or standard distance $d_{in} = \|x_n - v_i\|^2$ as follows

$$D_{in} = (1-\lambda) d_{in} f_{in} + \lambda d_{in} \tag{12}$$

where $\lambda \in [0,1]$ is an experimentally selected weight, and f_{in} is a spatial or local data function given by [13].

$$f_{in} = \frac{\sum_{k \in N_n} d_{ik}}{\min\left\{ \sum_{k \in N_n} d_{ik}; i = 1, 2, .., C \right\}} \tag{13}$$

It is obvious from (12) that with $\lambda = 1,$ the SFCM clustering method reduces to the standard FCM method. The spatial data function f_{in} is dependent on the original distances of the set of pixels N_n in the immediate neighborhood of the nth pixel. If all pixels in the neighbor set do

not belong to the ith cluster f_{in} is maximum since the denominator is minimum while the numerator is maximum. This implies that f_{in} causes D_{in} to increase when the pixels of the immediate neighborhood of the nth pixel do not belong to the ith cluster. This increase of D_{in} contributes to decreasing the membership u_{in} for achieving and preserving the minim of the SFCM function in (11).

The membership u_{in} and the cluster-center v_i associated with the SFCM method are given by [13].

$$u_{in} = \frac{1}{\sum_{j=1}^{C} \left(\frac{D_{in}}{D_{jn}}\right)^{\frac{1}{(m-1)}}} \tag{14}$$

$$v_i = \frac{\sum_{n=1}^{N} u_{in} x_n}{\sum_{n=1}^{N} u_{in}} \tag{15}$$

It is obvious from (14) that similar to the standard FCM, the membership u_{in} is inversely proportional to the weighted distance D_{in}, which again means that, increasing D_{in} when the immediate neighboring pixels to the nth pixel do not belong to the ith cluster, decreases the membership function u_{in}. From (15), however, it is clear that the SFCM algorithm computes the cluster-center v_i in a similar way as the standard FCM method does. Hence, additive noise can still reduce the accuracy of cluster center v_i obtained by the SFCM algorithm.

4.4. HCM incorporating membership entropy

The membership entropy has been incorporated into the HCM for fuzzification. The membership entropy-based FCM (MEFCM) algorithm has the following objective function [17].

$$J_{MEFCM} = J_{HCM} + \beta \sum_{i=1}^{C} \sum_{n=1}^{N} \left(u_{in} \log\left(u_{in}\right) + \left(1 - u_{in}\right) \log\left(1 - u_{in}\right)\right) \tag{16}$$

where $\beta > 0$ is a weight experimentally selected to control the fuzziness of the entropy term. We still need U to be constrained to satisfy (5). It can be shown that the membership and the cluster-center that minimize (16) are respectively given by [17]

$$u_{in} = \frac{1}{\sum_{j=1}^{C} \frac{\exp\left(d_{in}/\beta\right)+1}{\exp\left(d_{jn}/\beta\right)+1}} \tag{17}$$

$$v_i = \frac{\sum_{n=1}^{N} u_{in} x_n}{\sum_{n=1}^{N} u_{in}} \tag{18}$$

It is obvious so far that the membership function of the nth entities provided by FCM, HCM and MEFCM algorithms depends upon the inverse of the square of the Euclidean distance $d_{in} = \|x_n - v_i\|^2$ which is a function of only x_n with no data or membership information of the clustering entity's neighbors are involved. Hence, the FCM, HCM and MEFCM algorithms miss important spatial local data and membership information. Thus additive noise can degrade x_n, v_i and d_{in}, thereby biasing the membership of a degraded entity to a false cluster.

5. HCM incorporating local membership KL divergence

In [18], an approach to incorporating local spatial membership information into HCM algorithm has been presented. By adding Kullback-Leibler (KL) divergence between the membership function of an entity and the locally-smoothed membership in the immediate spatial neighborhood, the modified objective function, called the local membership KL divergence-based FCM (LMKLFCM), is given by [18–22].

$$J_{LMKLFCM} = J_{HCM} + \gamma \left(\sum_{i=1}^{C} \sum_{n=1}^{N} u_{in} \log \left(\frac{u_{in}}{\pi_{in}} \right) + \sum_{i=1}^{C} \sum_{n=1}^{N} \overline{u}_{in} \log \left(\frac{\overline{u}_{in}}{\overline{\pi}_{in}} \right) \right) \tag{19}$$

where γ is a weighting parameter experimentally selected to control the fuzziness induced by the second term in (19), $\overline{u}_{in} = 1 - u_{in}$ is the complement of the membership function u_{in}, π_{in} and $\overline{\pi}_{in}$ are the spatial local or moving averages of membership u_{in} and the complement membership $\overline{u}_{in}$, functions respectively. These local membership and membership complement averages are computed by [18–22].

$$\pi_{in} = \frac{1}{N_K} \sum_{k \in N_n;\, k \neq n} u_{ik} \tag{20}$$

$$\overline{\pi}_{in} = \frac{1}{N_K} \sum_{k \in N_n;\, k \neq n} (1 - u_{ik}) = 1 - \pi_{in} \tag{21}$$

where N_n is a set of entities/pixels falling in a square window around the nth pixel and N_K is its cardinality. It is obvious that all entities in the window are weighted equally by $w_{pq}^{(u)} = 1/N_K$. Other windows can be used such as Gaussian one provided that the weight of the window-center is 0 and the rest weights are summed to unity. The first term in (19) provides hard-cluster labeling. It pushes the membership function toward 0 or 1. The KL membership and membership-complement divergences, in addition to providing fuzzification approach to HCM clustering, measure the proximity between the membership of a pixel in a certain cluster and the local average of the membership over the immediate spatial neighborhood pixels in this cluster. So, they push the membership function to the locally smoothed membership function π_{in}. Therefore, this can smooth out additive noise and bias the solution to piecewise homogenous labels which leads to a segmented image with piecewise homogenous regions.

The minimization of the objective function $J_{LMKLFCM}$ in (19) yields u_{in} and v_i to be given, respectively, by [18].

$$u_{in} = \frac{1}{\sum_{j=1}^{C} \left(\frac{\pi_{jn}\left((1-\pi_{in}) \exp(d_{in}/\gamma) + \pi_{in} \right)}{\left(1-\pi_{jn}\right) \exp\left(d_{jn}/\gamma\right) + \pi_{jn}} \right)} \pi_{in} = \delta_{in} \pi_{in} \tag{22}$$

$$v_i = \frac{\sum_{n=1}^{N} u_{in} x_n}{\sum_{n=1}^{N} u_{in}} \tag{23}$$

It is obvious from (22) that u_{in} is proportional to π_{in} and the proportional parameter δ_{in} is inversely proportional to the entity's distance d_{in} and the maximum δ_{kn} occurs when $d_{kn} = 0$.

It is clear that if $\gamma \to \infty$, $u_{in} = \pi_{in} / \sum_{j=1}^{C} \pi_{jn}$. Therefore, the resultant membership is independent of the data to be clustered but dependent on the initial value of the membership matrix U^0 and on the smoothing fashion. If u_{in}^0 is generated from a random process greater than zero, then u_{in}^t versus the number of iteration t converges, because of recursive averaging and normalizing, to a normal distribution variable with mean equal to $\frac{1}{C} = E\{u_{in}^t\} = E\{\pi_{in}\} / \sum_{j=1}^{C} E\{\pi_{jn}\}$ which, in this case, means too much fuzzy membership function. This has been proved experimentally by using a synthetic image of 4 clusters and $\gamma = 10^{10}$. Finally, as shown by (23), the computation of the cluster-center v_i is still independent of the local original data.

6. HCM incorporating local data and membership KL divergence

To incorporate local spatial data into the LMKLFCM objective function in (19), the following objective function has been proposed in [18].

$$J_{LDMKLFCM} = \sum_{i=1}^{C} \sum_{n=1}^{N} u_{in}\left(d_{in} + \alpha \overline{d}_{in}\right) + \gamma\left(\sum_{i=1}^{C} \sum_{n=1}^{N} u_{in} \log\left(\frac{u_{in}}{\pi_{in}}\right) + \sum_{i=1}^{C} \sum_{n=1}^{N} \overline{u}_{in} \log\left(\frac{\overline{u}_{in}}{\overline{\pi}_{in}}\right)\right) \tag{24}$$

Therefore, similar to (22) and (23), the membership function u_{in} and the cluster-center v_i are, respectively, given by [18].

$$u_{in} = \frac{1}{\sum_{j=1}^{C} \left(\frac{\pi_{jn}\left((1-\pi_{in}) \exp\left(\left(d_{in} + \alpha \overline{d}_{in}\right)/\gamma \right) + \pi_{in} \right)}{\left((1-\pi_{jn}) \exp\left(\left(d_{in} + \alpha \overline{d}_{in}\right)/\gamma \right) + \pi_{jn} \right)} \right)} \pi_{in} \tag{25}$$

$$v_i = \frac{\sum_{n=1}^{N} u_{in}(x_n + \alpha \overline{x}_n)}{(1+\alpha) \sum_{n=1}^{N} u_{in}} \tag{26}$$

It is obvious that the LDMKLFCM algorithm in (24)–(26) provides a membership that depends upon the local spatial data and membership information while the cluster center is dependent upon the locally-smoothed data. Thus the algorithm has twofold approach to handle additive noise.

7. Simulation results

This simulation aims at examining the performance of the conventional FCM, the membership entropy-based FCM (MEFCM), the spatial distance weighted FCM (SFCM), the local membership KL divergence-based FCM (LMKLFCM) and the local data and membership KL

divergence-based FCM (LDMKLFCM) algorithms. It is to be noticed that all the algorithms can be implemented almost similar to the pseudo code in **Table 1** by replacing the steps 3 and 4 by the corresponding computation of the membership function and cluster centers of each algorithm.

7.1. Clustering validity

To measure the performance of the fuzzy clustering algorithms, several quantitative measures or indices have been adopted in [23, 25] and references therein. Few of these measures are the partition coefficient V_{PC} and the partition entropy V_{PE} index of Bezdek and Xie-Beni (XB index) V_{XB}, given respectively by

$$V_{PC} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} u_{in} \tag{27}$$

$$V_{PE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} u_{in}\log\left(u_{in}\right) \tag{28}$$

The closer of the V_{PC} to 1, the better the performance since the minimization is constrained by $\sum_{i=1}^{C} u_{in} = 1$. The closer the V_{PE} to 0, the better the performance since this means the less fuzziness of the membership and thus clusters are well-separated.

In synthetic images, in addition to the above clustering validity measures, several clustering validity and performance measures have also been used such as the accuracy, sensitivity and specificity given respectively by

$$Acc. = (TP + TN)/(TP + TN + FP + FN) \tag{29}$$

$$Sen. = TP/(TP + TN) \tag{30}$$

$$Spe. = TN/(TN + FN) \tag{31}$$

where *T, F, P,* and *N* are mean true, false, positive, and negative, respectively. The *TP, FP, TN,* and *FN* are computed as follows. While generating the synthetic image, the ground truth labels are formulated as the logical matrix given by [23].

$$L_{in} = \begin{cases} 1; \textit{if } x_n \in i \\ 0; \textit{otherwise} \end{cases} ;\ i = 1, 2, \ldots, C, n = 1, 2, .., N. \tag{32}$$

where x_n is the noise-free pixel in the synthetic image and 1 and 0 represent True and False, respectively. After the segmentation is done, the estimated labels are also formulated as logical matrices generated by [20].

$$\widehat{L}_{kn} = \begin{cases} 1; k = \arg\max_i(u_{in}) \\ 0; \textit{otherwise} \end{cases} ;\ i = 1, 2, \ldots, C, n = 1, 2, .., N. \tag{33}$$

Finally, the *TP*, *TN*, *FP,* and *FN* are given by [20].

$$TP = \sum_{i=1}^{C}\sum_{n=1}^{N} \widehat{L}_{in} L_{in}; \qquad TN = \sum_{i=1}^{C}\sum_{n=1}^{N} \overline{\widehat{L}}_{in} \overline{L}_{in}$$
$$FP = \sum_{i=1}^{C}\sum_{n=1}^{N} \widehat{L}_{in} \overline{L}_{in}; \qquad FN = \sum_{i=1}^{C}\sum_{n=1}^{N} \overline{\widehat{L}}_{in} L_{in} \tag{34}$$

where "__" means the logical complement.

7.2. Artificial image

In this simulation, the artificial or synthetic noise-free image shown in **Figure 1(a)** is degraded by adding zero-mean white Gaussian noise (WGN) with different variances. The noisy image

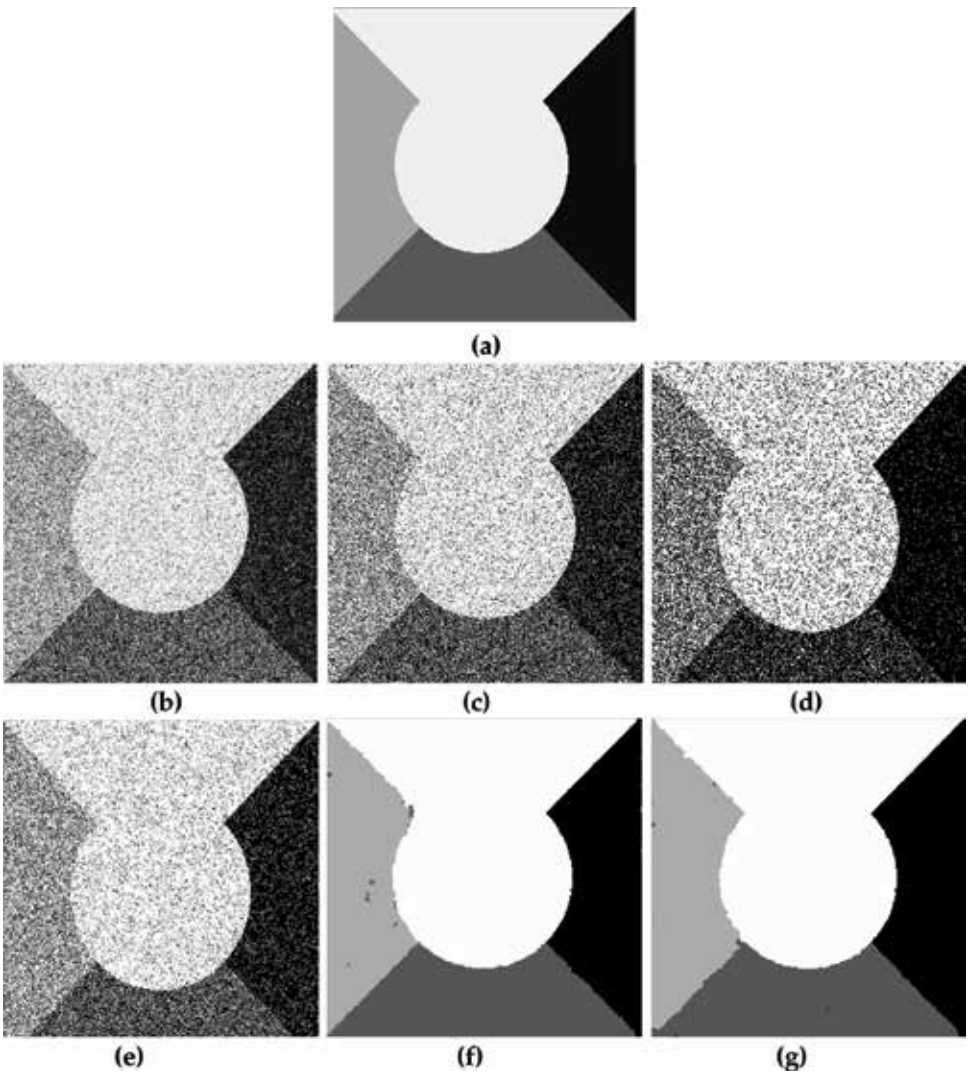

Figure 1. Clustering of the synthetic image: (a), noise free-image; (b), the noise-free image plus zero-mean and 0.08 variance WGN; (c) FCM; (d), MEFCM; (e), SFCM; (f), LMKLFCM; (g), LDMKLFCM. It is evident that the clustered images in (f) and (g) have lesser number of misclassified pixels which means that noisy pixels are rightly clustered. Clustering validation measures are summarized in **Table 2**.

shown in **Figure 1(b)** is for 0.08 noise variance. We have studied the performance of the five algorithms, namely, the standard FCM, the membership entropy-based FCM (MEFCM), the spatial distance weighted FCM (SFCM), the local membership KLFCM (LMKLFCM) and the local data and membership KLFCM (LDMKLFCM) algorithms in segmenting these noisy images with $m = 2$ and $C = 4$. The parameters for the algorithms have been elected via simulation as $\beta = 1000$ for MEFCM; $\lambda = 0.5$ for SFCM; $\gamma = 1000$ for LMKLFCM; and $\gamma = 1000$ and $\alpha = 0.5$ for LDMKLFCM. For the computation of the locally smoothed data $\bar{x}_n$, a neighboring window of size 3x3 has been used. Also, the same spatial window has been used for the computation of the locally-smoothed membership function π_{in}. The initial values of the membership functions U and the cluster-centers V are generated from a uniformly distributed random process with means 0.5 and equal to the image mean, respectively. We have collected results from 25 Monte Carlo runs of each algorithm. In each run, the initial values of U and V of the FCM are new random samples while the ones of the rest algorithms are generated by executing few number of iterations of the FCM algorithm. Simulation results, not included for space limitation, have shown that the algorithms provide further improvement with these initial values generated by the FCM algorithm than those randomly generated. Also, in each run, a new random sample of WGN is used in generating the noisy images. **Figure 1(c–g)** show the clustered images generated by the five algorithms in the case of 0.08 noise variance. These clustered images show that the LMKLFCM and the LDMKLEFCM algorithms provide the ones with lesser noise which means lesser number of misclassified pixels. Moreover, the LDMKLFCM algorithm offers the superior clustered image. **Table 2** summarizes the averages and standard deviations ($\mu \pm \sigma$) of the performance measures. The LMKLFCM and LDMKLFCM show the maximum V_{PC} and the minimum V_{PE}. The averages of the accuracy, sensitivity and the specificity performance measures of the five algorithms have been studied

Algorithm	Images	V_{PC}	V_{PE}
FCM	Synthetic	0.8105 ±0.0007	0.3517 ± 0.0012
	Simulated MR	0.7921 ±0.0011	0.3986 ± 0.0020
	Real MR	0.8930 ±0.0140	0.1998 ± 0.0240
	Lena	0.8286 ±0.0004	0.2824 ± 0.0006
SFCM	Synthetic	0.8370 ± 0.0010	0.3017 ± 0.0017
	Simulated MR	0.8674 ± 0.0009	0.2409 ± 0.0014
	Real MR	0.9204 ± 0.0006	0.1440 ± 0.0012
	Lena	0.8936 ± 0.0006	0.1786 ± 0.0009
MEFCM	Synthetic	0.8616 ± 0.0012	0.2271 ± 0.0019
	Simulated MR	0.8873 ± 0.0012	0.1841 ± 0.0018
	Real MR	0.9602 ± 0.0113	0.0650 ± 0.0183
	Lena	0.9268 ± 0.0004	0.1198 ± 0.0007
LMKLFCM	Synthetic	0.9853 ± 0.0011	0.0270 ± 0.0028
	Simulated MR	0.8958 ± 0.0088	0.1721 ± 0.0146
	Real MR	0.9625 ± 0.0087	0.0441 ± 0.0128
	Lena	0.9609 ± 0.0012	0.0643 ± 0.0020
LDMKLFCM	Synthetic	0.9874 ± 0.0011	0.0227 ± 0.0022
	Simulated MR	0.9234 ± 0.0030	0.1258 ± 0.0049
	Real MR	0.9519 ± 0.0016	0.0604 ± 0.0025
	Lena	0.9730 ± 0.0026	0.0446 ± 0.0026

Table 2. Clustering validation measures for synthetic and real-world images.

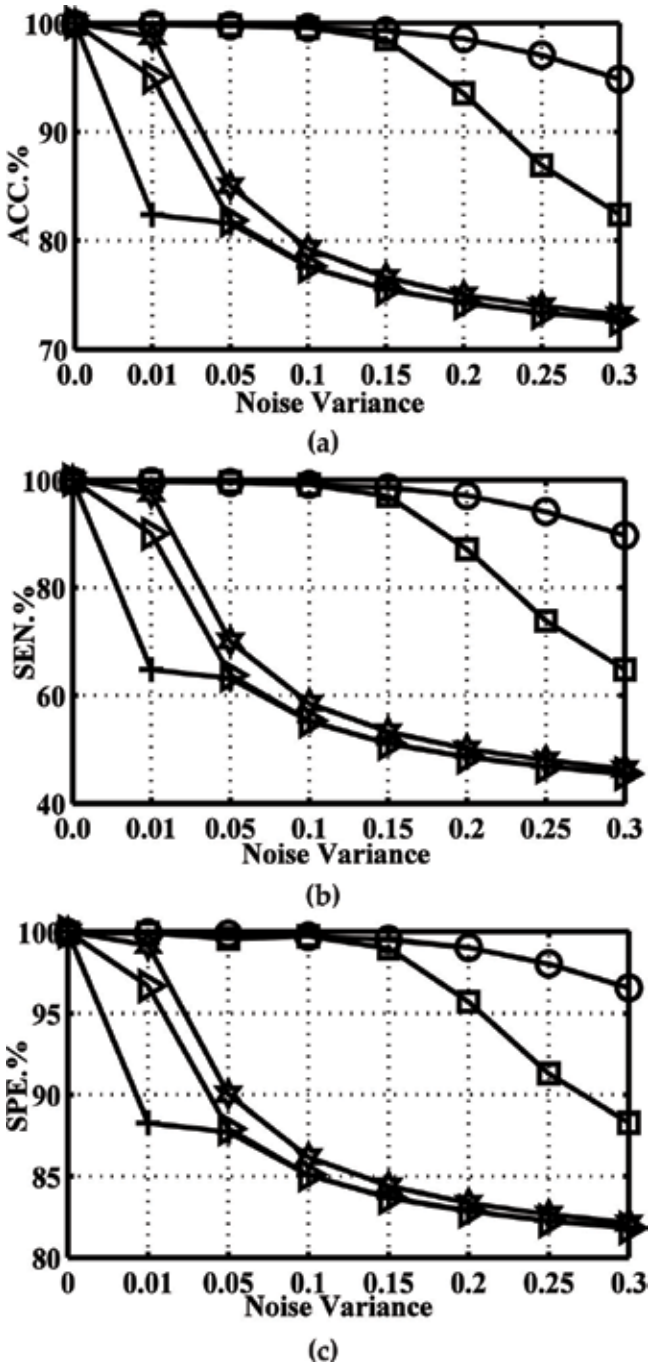

Figure 2. The average versus noise variance of accuracy, (a); sensitivity, (b); and specificity, (c); ▷, FCM; +, MEFCM; SFCM; LMKLFCM; LDMKLFCM. The proposed LMKLFCM and LDMKLFCM algorithms provide the superior performance among the five algorithms. The LDMKLFCM algorithm shows more noise-robust capability.

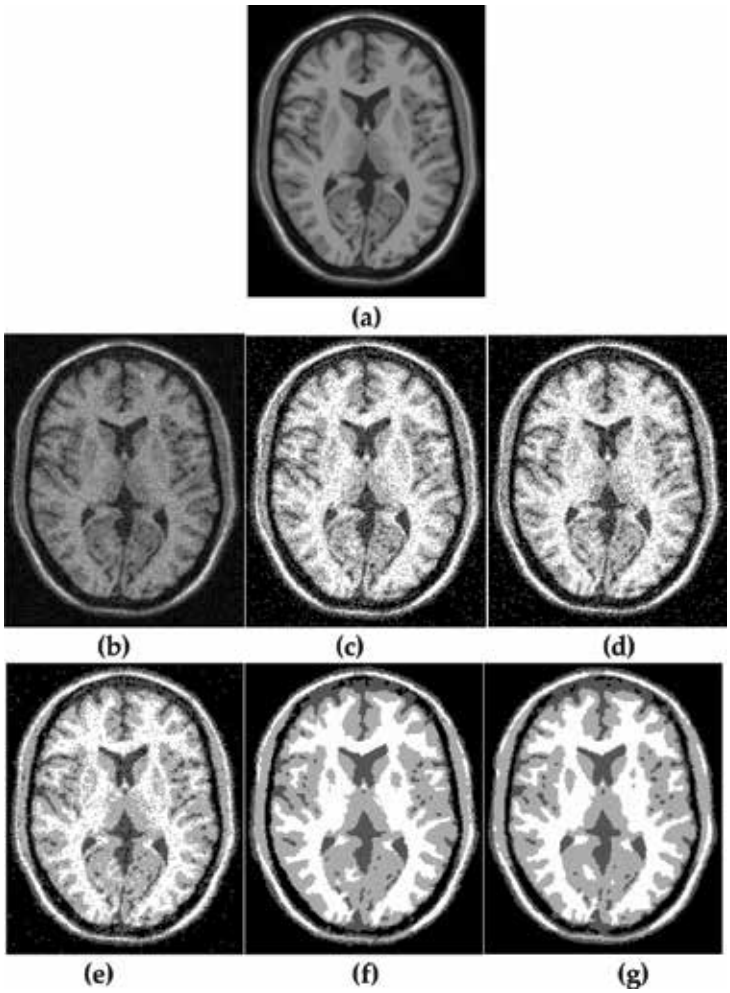

Figure 3. Clustering of simulated MRI: (a), noise-free MRI; (b), the MRI in (a) plus zero-mean WGN with 0.005 variance. Segmented images by: (c), FCM; (d), MEFCM; (e), SFCM; (f), LMKLFCM (g), LDMKLFCM. Obviously, the segmented images in (f) and (g) provided by the LMKLFCM and the LDMRKLCM algorithms, respectively, have lesser noise which means that the noisy pixels are correctly clustered. The clustering validation measures summarized in **Table 2** show that the LMRKlCM; and LDMKLFCM provide the maximum V_{PC} and the minimum V_{PE}.

against noise variance. **Figure 2** shows these measures versus noise variance. It is clear that both the LMKLFCM and the LDMKLFCM algorithms provide the superior performance among the five algorithms and the LDMKLFCM algorithm shows more noise-robustness.

7.3. Magnetic resonance image (MRI)

A simulated MRI of [26], illustrated by **Figure 3(a)**, has been used as a noise-free image. It has been degraded by adding white Gaussian noise (WGN) with zero-mean and 0.005 variance to

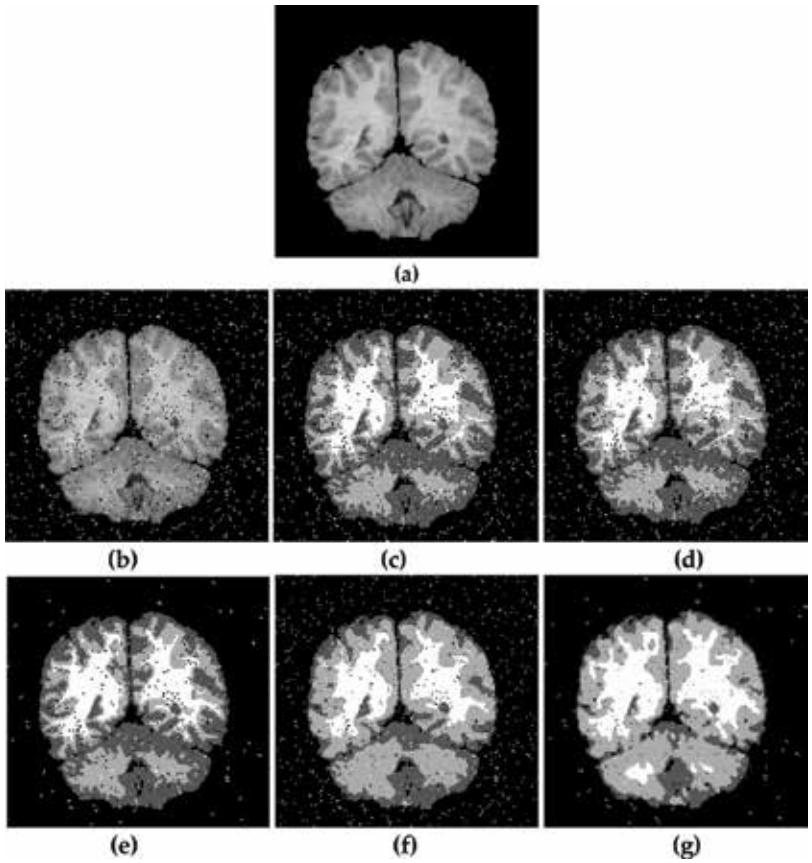

Figure 4. Clustering of real MRI example: (a), noise-free real MRI; (b), the image in (a) plus salt&pepper with 0.05 variance. Segmented images by: (c), FCM; (d), MEFCM; (e), SFCM; (f), LMKLFCM (g), LDMKLFCM. Clearly, the segmented images in (f) and (g) generated by the LMKLFCM and the LDMRKLCM algorithms, respectively, have lesser noise. The clustering validation coefficients summarized in **Table 2** show that the LMRKlCM; and LDMKLFCM provide the maximum V_{PC} and the minimum V_{PE}.

generate the noisy MRI illustrated by **Figure 3(b)**. This noisy MRI image has been clustered by the five algorithms. The parameters for all algorithms have been taken similar to the ones of the synthetic image simulation except, for the MEFCM algorithm, $\beta = 200$ and, for both LMKLFCM and LDMKLFCM algorithms, $\gamma = 1000$. We have also executed 25 runs of each algorithm. The initial values of u_{in} and v_i have been generated and adjusted as explained in the synthetic image simulation. **Figure 3(c–g)** shows the resulting clustered images provided by the five algorithms in a certain run. **Table 2** shows the averages and standard deviations ($\mu \pm \sigma$) of the performance measures V_{PC} and V_{PE} of the five algorithms. It obvious that the LMKLFCM and LDMKLFCM provide the segmented images with lesser noise or lesser number of misclassified pixels, the maximum V_{PC} and the minimum V_{PE}.

A real MRI from [27], shown in **Figure 4(a)**, has been considered as a noise-free image. To generate the noisy MRI shown in **Figure 4(b)**, salt & pepper noise with 0.050 variance have

http://dx.doi.org/10.5772/intechopen.74514

been added. The noisy MRI has been clustered by the FCM, SFCM, MEFCM, LMKLCM and the LDMKLFCM algorithms. The parameters for all algorithms have been taken similar to the ones of the synthetic image simulation except, for the MEFCM algorithm, $\beta = 300$ and, for both the LMKLFCM and LDMKLFCM algorithms, $\gamma = 800$. We have also obtained the results of 25 runs of each algorithm. The initial values of u_{in} and v_i have been generated and adjusted as

Figure 5. Segmentation of Lena image: (a), noise-free image; (b), the image in (a) plus WGN noise with zero-mean and 0.05 variance. It is obvious that the images in (f) and (g) have lesser number of misclassified pixels. The clustering validation coefficients summarized in **Table 2** which shows that the LMKLFCM and the LDMKLFCM algorithms provide the superior V_{PC} and V_{PE}.

mentioned in the synthetic image simulation. **Figure 4(c–g)** show the segmented images provided by the five algorithms in a certain run while **Table 2** summarizes the averages and standard deviations ($\mu \pm \sigma$) of the performance measures. It is obvious that the proposed LMKLFCM and LDMKLFCM algorithms provide the segmented images with lesser noise or lesser number of misclassified pixels, the maximum V_{PC} and the minimum V_{PE}.

7.4. Lena image

A popular Lena image shown in **Figure 5(a)** has been considered as a noise-free image example. The noisy Lena image shown in **Figure 5(b)** has been generated by adding WGN noise with zero-mean and 0.01 variance. The parameters of the five algorithms have been adjusted to the values similar to the ones used in the previous simulations except $C = 2$; $\beta = 1000$ for the MEFCM algorithm; $\gamma = 2000$ for the LMREFCM and $\gamma = 2000$ and $\alpha = 0.5$ for the LDMREFCM algorithms. We have also executed 25 Mont Carlo Runs of each algorithm as explained above. **Figure 5(c–g)** shows the resulting segmented images obtained by the five algorithms. Visually investigation of the segmented images shows that the LMKLFCM and LDMKLFCM algorithms provide the images with lesser number of misclassified pixels. **Table 2** shows the average and standard deviation ($\mu \pm \sigma$) of the performance measures of the five algorithms. It is also clear that the two algorithms provide the maximum V_{PC} and the minimum V_{PE}.

8. Conclusions

The hard C-means algorithm has been fuzzified by incorporating into the objective function spatial local information through two KL membership divergences. The first KL membership divergence measures the information proximity between the membership of each pixel and its local membership average in the pixel neighborhood. The second one measures the information proximity between the complement membership and its local membership average in the pixel neighborhood. For regularization, the local data information has been incorporated by an additional new weighted hard C-means function in which the noisy-image is replaced by a noise-reduced one. Such incorporation of both local data and local membership information facilitates biasing the algorithm to classify each pixel in correlation with its immediate neighboring pixels. Results of segmentation of synthetic, simulated medical and real-world images have shown that the proposed local membership KL divergence-based FCM (LMKLFCM) and the local data and membership KL divergence-based entropy FCM (LDMKLFCM) algorithms outperform several widely used FCM related algorithms. Moreover, the average runtimes of all algorithms have been measured via simulation. In all runs, all algorithms start from the same randomly generated initial conditions, as mentioned in the simulation section, and stopped at the same fixed point. The LDMKLFCM, LMKLFCM, standard FCM, MEFCM, and SFCM algorithms have provided average runtime of 1.5, 1.75, 1, 0.9 and 1 sec respectively. The simulation results have been done using Matlab R2013b under windows on a processor of Intel (R) core (TM) i3, CPU M370 2.4 GHZ, 4 GB RAM.

Acknowledgements

The author would like to thank for funding the open access publication of this Chapter. Also, the author would like to thank Prof. H. Selim, Dr. A. AbdelFattah and Eng. G. Gendy for their contribution to this work.

Conflict of interest

No potential conflicts of interest to report.

Author details

Reda R. Gharieb[1,2]*

*Address all correspondence to: rrgharieb@gmail.com

1 Faculty of Engineering, Assiut University, Assiut, Egypt

2 Higher Institute of Engineering, Thebes Academy for Sciences, Cairo, Egypt

References

[1] Pal NR, Pal SK. A review on image segmentation techniques. Pattern Recognition. 1993; **26**(9):1277-1294

[2] Ravindraiah R, Tejaswini KA. Survey of image segmentation algorithms based on expectation-maximization. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP). 2013;**2**:01-07

[3] Pham DL, Xu C, Prince JL. Current methods in medical image segmentation. Annual Review of Biomedical Engineering. 2000;**2**:315-337

[4] Jain AK. Data clustering: 50 years beyond K-means. Pattern Recognition Letters. 2010;**3**: 651-666

[5] MacQueen J. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1967;**14**:281-297

[6] Alsabti K, Ranka S, Singh V: An efficient k-means clustering algorithm. Electrical Engineering and Computer Science. Paper 43

[7] Bezdek JC. Pattern Recognition with Objective Fuzzy Algorithms. New York: Plenum Press; 1981

[8] Zadeh LA. Fuzzy sets. Information and Control. 1965;**8**:338-353

[9] Chuang KS, Tzeng HL, Chen S, Wu J, Chen TJ. Fuzzy c-means clustering with spatial information for image segmentation. Computerized Medical Imaging and Graphics. 2005;**30**:9-15

[10] Miyamoto S, Ichihashi H, Honda K. Algorithms for Fuzzy Clustering. Heidelberg: Springer; 2008

[11] Ahmed MN, Ymany S, Mohamed N, Farag A, Moriarty T. A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data. IEEE Transactions on Medical Imaging. 2002;**21**:193-199

[12] Krinidis CSV. A robust fuzzy local information and C-means clustering algorithm. IEEE Transactions on Image Processing. 2010;(5):1328-1337

[13] Guo Y, Liu K, Wu Q, Hong Q, Zhang H. A new spatial fuzzy c-means for spatial clustering. Wseas Transactions on Computer. 2015;**14**:369-381

[14] Cai W, Chen S, Zhang D. Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation. Pattern Recognition. 2005;**40**:825-838

[15] Honda K, Ichihashi H. A new approach to fuzzification of memberships in cluster analysis. Modeling Decisions for Artificial Intelligence. 2005:97-124

[16] Yao J, Dash M, Tan ST, Liu H. Entropy-based fuzzy clustering and fuzzy modeling. Fuzzy Sets and Systems. 2000;**113**:381-388

[17] Yasuda M. Fuzzy c-Means Clustering, Entropy Maximization, and Deterministic and Simulated Annealing. InTech Book of Simulated Annealing-Advances, Applications and Hybridizations. 2012. DOI: 10.5772/48659

[18] Gharieb RR, Gendy G, Abdelfattah A. C-means clustering fuzzified by two membership relative entropy functions approach incorporating local data information for noisy image segmentation. Signal, Image and Video Processing. 2017;**11**:541-548. https://doi.org/10.1007/s11760-016-0992-4

[19] Gharieb RR, Gendy G, Selim H. A hard C-means clustering algorithm incorporating membership KL divergence and local data information for noisy image segmentation. International Journal of Pattern Recognition and Artificial Intelligence. 2017:1-12. https://doi.org/10.1142/S021800141850012X

[20] Gharieb RR, Gendy G, Abdelfattah A, Selim H. Adaptive local data and membership based KL divergence incorporating C-means algorithm for fuzzy image segmentation. Applied Soft Computing. 2017. https://doi.org/10.1016/j.asoc.2017.05.055

[21] Gharieb RR. Data Science- Scientific and Statistical Computing. Germany: Noor Publishing; 2017

[22] Gharieb RR, Gendy G. Fuzzy C-means with a local membership KL distance for medical image segmentation. In: Proceedings of the IEEE International Conference on Biomedical Engineering Conference (CIBEC 14); 11–13 December 2014; Cairo; IEEE; p. 47-50

[23] Pal NR, Bezdek JC. On cluster validity for the fuzzy c-means model. IEEE Transactions on Fuzzy Systems. 1995;**3**:370-379

[24] Gharieb RR. Gendy fuzzy C-means with local membership based weighted pixel distance and KL divergence for image segmentation. Journal of Pattern Recognition Research. 2015;**1**:53-60

[25] Wang W, Zhang Y. On fuzzy cluster validity indices. Fuzzy Sets and Systems. 2007;**158**: 2095-2117

[26] Online simulated brain web. Available at: http://brainweb.bic.mni.mcgill.ca/brainweb/

[27] Internet brain segmentation repository (ibsr). Available at: https://www.nitrc.org/projects/ibsr

Chapter 4

Centroid-Based Lexical Clustering

Khaled Abdalgader

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75433

Abstract

Conventional lexical-clustering algorithms treat text fragments as a mixed collection of words, with a semantic similarity between them calculated based on the term of how many the particular word occurs within the compared fragments. Whereas this technique is appropriate for clustering large-sized textual collections, it operates poorly when clustering small-sized texts such as sentences. This is due to compared sentences that may be linguistically similar despite having no words in common. This chapter presents a new version of the original *k*-means method for sentence-level text clustering that is relay on the idea of use of the related synonyms in order to construct the rich semantic vectors. These vectors represent a sentence using linguistic information resulting from a lexical database founded to determine the actual sense to a word, based on the context in which it occurs. Therefore, while traditional *k*-means method application is relay on calculating the distance between patterns, the new proposed version operates by calculating the semantic similarity between sentences. This allows it to capture a higher degree of semantic or linguistic information existing within the clustered sentences. Experimental results illustrate that the proposed version of clustering algorithm performs favorably against other well-known clustering algorithms on several standard datasets.

Keywords: semantic similarity, sentence-level text clustering, word sense identification, lexical resource

1. Introduction

Although lexical clustering at the document-level text is well studied in the natural language processing (NLP), computational linguistic, and knowledge discovery literature, clustering at the sentence-text level is challenged by the fact that word frequency—possible frequent occurrence of words from textual collection—on which most text semantic similarity methods are

based, may be absent between two semantically similar text fragments. To solve this problem, several sentence-level text similarity methods have recently been established [1–17][1].

The sentence similarity measures proposed by Li et al. [1], Mihalcea et al. [2], and Wang et al. [18] have two major features in common. Firstly, rather than using all possible features from applied external textual collections to representing sentences in a vector space model [19], only the words appearing in the compared sentences are used, thus solving the issue of data sparseness (i.e., high dimensionally) resulting from a randomly processing of the words (i.e., bag of words representation). Secondly, they use the available semantic and linguistic information from the applied lexical sources to solve the issue of deficiency of word co-occurrence.

The measures of sentence-level text similarity such as presented by Abdalgader and Skabar [10] (the latter of which we use in this chapter and described later in Section 2) depend in a way of using the word-related synonyms to calculating the semantic similarity between words. Unlike existing measure of short text semantic similarity, which use the exact words that appear in the compared sentences, this similarity method creates an expansion word set for each sentence using related synonyms of the sense-disambiguated words in that sentence. This way lead to provide a richer and highly connected semantic context to estimate sentence similarity through better utilization of the possible semantic information from the available lexical resources such as WordNet [20, 21]. For each of the sentences being calculated for their similarity, a *word sense identification* step is first applied in order to determine the correct sense based on the surrounding context [22]. A synonym expansion step is then applied, resulting in a richer and fully connected semantic context from which to estimate semantic vectors. The similarity between these vectors can then be calculated using a standard vector space similarity measure (i.e., cosine measure).

Several text-clustering methods: however, have been existed in the study [18, 23–37, 38–40, 42], and a majority of them consider the matrix of semantic similarities between words as input only. The *k*-medoids [30, 31] is one of these methods, which is considered as a developed version of *k*-means method in which centroids are restricted to being data patterns (i.e., points). However, a problem with the *k*-medoid method is that it is highly sensitive to the random selection (i.e., initial) of centroids, and in empirical executions, it is often requiring to be executed many times with different initialization settings. To solve this issue with *k*-medoids, Frey and Dueck [35] proposed *Affinity Propagation*, a graph-based algorithm that concurrently does take all data points as possible centroids (i.e., exemplars). Processing each data point as a node in a graph, affinity propagation recursively transfers real-valued messages along the vertices of the graph until a required set of possible centroids are achieved.

Another graph-based clustering method that depends on matrix decomposition techniques from the linear algebra theories is a spectral-clustering algorithm [18, 36, 37, 39, 41]. Rather than clustering data patterns in the traditional vector space model, it associated data patterns together with the space resulted from eigen-vectors linked with the top eigen-values and then apply clustering in this new transformed space, usually applying a *k*-means method. One of

[1]This chapter adapts the journal version that appeared in the IAENG International Journal of Computer Science, 44:4, IJCS_44_4_12 [42].

the benefits of this method is that it has the ability to classify non-convex classes, which is challenging when clustering by using *k*-means method (i.e., typical feature space). Since spectral-clustering method requires only a matrix comprising pairwise similarity as input, it is easy to apply it to the sentence-level text-clustering task [18, 29].

Erkan and Radev [43], Mihalcea and Tarau [44], and Fang et al. [46] have applied a PageRank [45] as a centrality measure in the task of document summarization, in which the aim is to rank sentences regarding their role in the document being summarized. Importantly, Skabar and Abdalgader [29] proposed a new fuzzy sentence-level text-clustering method that also uses PageRank as a centrality measure, and it allows clustered sentences to belong to all classes with different degrees of similarity (i.e., membership). The nation of this fuzzy clustering is required in the case of document summarization, in which a sentence may be linguistically similar or related to more than one topic [14, 29, 47].

The contribution presented in this chapter is a new version of the original *k*-means method for sentence-level text clustering that is dependent on the idea of using the related synonym sets to create rich and highly connected semantic vectors [42]. These vectors characterize sentence using semantic information derived from a WordNet to determine the actual sense to a word, based on the surrounding context. Thus, while the original *k*-means method is relay on calculating the distance between patterns, the new version is operating by calculating the semantic similarity between sentences. This allows it to capture more semantic information accessible within the clustered sentences. The result is a centroid-based lexical-clustering method which can be used in any application in which the relationship between patterns is expressed in terms of pairwise semantic similarities. We apply the algorithm to several benchmark datasets and compare its performance with that of well-known clustering methods such as *spectral clustering* [36], *affinity propagation* [35], *k-medoids* [30, 31], *STC-LE* [39], and *k-means (TF-IDF)* [40]). We claim that the satisfactory performance of new proposed version of the centroid-based lexical-clustering method is due to its ability to better utilize and capture a higher degree of semantic information available in used lexical resource.

The remainder of this chapter is organized as follows. Section 2 presents a representation scheme for calculating sentence semantic similarity. Section 3 describes the proposed variation of original *k*-means clustering (centroid-based) method. Empirical results are shown in Section 4, and Section 5 concludes the chapter.

2. Semantic similarity representation scheme

By far, the most widely used text representation scheme in the natural language processing activities is the vector space model (VSM), in which a text or a document is represented as a point in a high-dimensional (N_i) input space. Each dimension in this input space (i.e., VSM) corresponds to a unique word [19]. That is, a document d_j is represented as a vector $x_j = (word_{1j}, word_{2j}, word_{3j}, \ldots)$, where $word_{ij}$ is a weight that represents in some way the importance or relatedness of word $word_i$ in document d_j and is dependent on the frequency of occurrence of $word_i$ in document d_j. The semantic similarity between the compared documents is then

measured using the corresponding vectors, and a usually applied measure is the cosine of the angle between the two vectors.

The VSM has been effective in information retrieval (IR) activities because it is able to sufficiently utilize much of the semantic information expressed in the larger-sized textual collection. This is due to a large textual collection or documents may contain many shared words with each other and thus be considered similar regarding to well-known vector space similarity measures such as the cosine measure. However, in the case of sentence-level text (text fragment), this is not the case, since two sentences may be carrying the same meaning (i.e., semantically similar) whereas comprising no similar words. For instance, consider the sentences "*Some places in the country are now in torrent crisis*" and "*The current flood disaster affects the particular states.*" Obviously, these two sentences have the same meaning, yet the only common word they have is *the*, which does not carry any semantic information (i.e., stop words). The reason why word co-occurrence may be rare or even absent in sentences is due to the flexibility of natural language that allows humans to express the same meanings using very different sentences in terms of structure and length [50]. Therefore, we need a sentence-level text representation scheme which is superiorly able to utilize and capture all the possible semantic information of sentences, thus enabling a more efficient similarity method to be used.

2.1. Measuring sentence-level text similarity

To calculate the semantic similarity between two sentences, we use sentence similarity method that uses the sets of synonym expansion appeared in the compared sentences [10]. To demonstrate how this measure work: however, suppose that $Sentence_1$ and $Sentence_2$ are the two sentences being compared to calculate their semantic similarity, W_1 and W_2 are the sets of sense-assigned words appeared in $Sentence_1$ and $Sentence_2$, respectively, $sentence_1$ and $sentence_2$ are the sets of synonym expansion appeared in W_1 and W_2, and $U = W_1 \cup W_2$. Then, a semantic vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ have been created, according to $sentence_1$ and $sentence_2$.

Let $word_j$ be the corresponding sense-assigned word from U and v_{ij} be the j^{th} element of v_i. In this case, there are two instances to take into the account, relaying on whether $word_j$ appears in $sentence_i$ or not:

Instance 1: If $word_j$ exists in $sentence_i$, then set v_{ij} equal to the value of 1, this is based on the semantic similarity of the same words in the WordNet.

Instance 2: If $word_j$ does not exist in $sentence_i$, then compute the semantic similarity between compared words by using one of the WordNet-based word-to-word similarity measures (i.e., J&C measure) [51]. The final similarity score to v_{ij} is the highest of these scores between $word_j$ and each $sentence_i$.

Once the vectors ($\mathbf{v}_1$ and $\mathbf{v}_2$) have been constructed, the semantic similarity between two sentences can be determined using a cosine similarity measure between two constructed vectors as

$$Similarity(Sentence_1, Sentence_2) = (\mathbf{v}_1.\mathbf{v}_2)/(|\mathbf{v}_1||\mathbf{v}_2|) \quad (1)$$

http://dx.doi.org/10.5772/intechopen.75433

2.2. Sentence-level clustering algorithm

In this section, we firstly describe the new proposed version of the original *k*-means clustering algorithm which we called it centroid-based lexical-clustering (CBLC) algorithm. Then, we describe how a cluster centroid can be constructed and defined. The remaining subsections discuss the issues of calculating the semantic similarity between sentences and clustering centroid, and other related technical issues such as empirical settings and space and time complexity.

2.3. Centroid-based lexical clustering

Algorithm 1. Centroid-Based Lexical Clustering (CBLC).

Input: Sentences to be clustered $S = \{S_i \mid i = 1 \text{ to } TN\}$

Classes # k

Output: Membership values of each cluster $\left\{\pi_i^j | i = 1..TN, j = 1..k\right\}$ where π_i^j is the membership value of sentences i to cluster j.

1. *//Randomly distribute the sentences into k classes*

2. **for** $i = 1$ to TN

3. **if** $i \leq k$

4. j + =1

5. $\pi_i^j = S_i$ *//Sentence*$_i$

6. **else**

7. $j = 1$

8. $\pi_i^j = S_i$ *//Sentence*$_i$

9. **end**

10. **repeat until there is no move (until convergence)**

11. *//Define or determine the centroid for each class (cluster)*

12. **for** $j = 1$ to k

13. M_j = *union-set {all possible synonym occurring in the cluster* $_j$*} //U set*

14. **end**

15. *//Compute the similarity between each sentence* (S_i) *to each cluster centroid*

16. **for** $j = 1$ to k

17. *similarity*(M_j, S_m) // S_m *is sentences related to cluster j, {m = 1.. n}.*

18. end

19. *//Re-locate each sentence to the corresponding cluster centroid to which it is similar to.*

20. re-*locate*(S_i, M_j)

21. End

Given a *k* set (i.e., clusters), partition all the data points (i.e., sentences) randomly in given sets (i.e., initialization), each with a determined centroid (*mean*) that demonstrates as representative of the cluster. There are iterations process that rearrange these means or centroid of the clusters, which is based on moving each sentence to the cluster corresponding to the centroid to which it is closest (i.e., semantically similar). Redetermine the cluster centroids based on the new located sentences belonging to them. Then, the following iteration is repeated until the centroids do not move (until convergence). The new proposed version of the original *k*-means clustering algorithm is as follows.

2.4. Determining a clustering centroid

In the standard vector space model, the text such as a document is processed as a vector (i.e., its elements are the *tf-idf* scores), a cluster centroid can be determined by taking into account the vector average over all text fragments related to that cluster. This is experienced very hard using the above-discussed text representation scheme, since the semantic vector for a sentence is not unique, but depends on the length of the compared sentence context. However, just as a context may be constructed by two sentences, it is direct to apply this nation to defining the context over a collection of sentences. While a cluster is just such text fragments, we can define the centroid of a cluster as the *union set* of all associated synonyms of disambiguated words existing in the sentences relating to that cluster. Thus, if $Sentence_1$, $Sentence_2$, … $Sentence_N$ are sentences belonging to some cluster, the centroid of the cluster, which we denote as M_j, is just the union set {$word_1$, $word_2$, .. $word_n$}, where *n* is the number of distinct synonym words ($sentence_i$) in$Sentence_1 \cup Sentence_2 \cup \ldots \cup Sentence_N$. **Figure 1** exemplifies the idea of determining a clustering centroid.

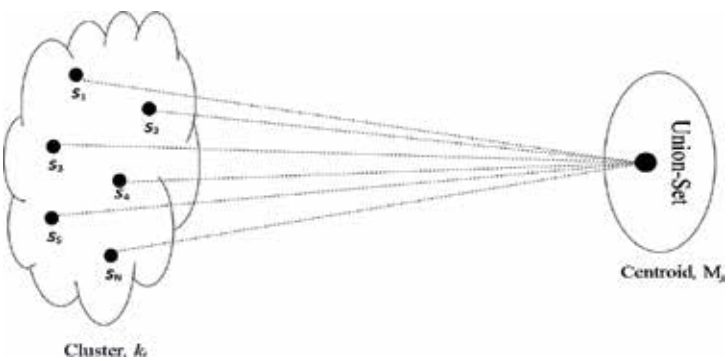

Figure 1. Clustering centroid, where $sentence_i$ (*si*) is a set of synonym words corresponding to *Sentencei* (S_i).

2.5. Calculating similarity between sentence and cluster centroid

When the CBLC algorithm calculates the semantic similarity between sentences, there are two cases to take into account. Firstly, if a sentence does belong to the cluster and secondly, if a sentence does not belong to the cluster. This case is straightforward to implement. Since the cluster centroids are represented in the same way as a union set (synonyms), the similarity between a sentence and a cluster centroid (i.e., two sentences) can be calculated by using sentence similarity measure, as described earlier. There is, however, a subtlety in the first case, which is not immediately apparent.

To demonstrate how this semantic similarity is calculated, assume that $\text{Sentence}_1 = \{word_1, word_2, word_3\}$ and $\text{Sentence}_2 = \{word_4, word_5\}$ are not semantically similar. Comparing these sentences (S_1 and S_2), we obtain the semantic vectors $\mathbf{v}_1 = \{1,1,1,0,0\}$ and $\mathbf{v}_2 = \{0,0,0,1,1\}$ which obviously have a cosine value of zero and is reliable with the fact that they are no semantic relation between them. Now suppose, however, that Sentence_1 (S_1) and Sentence_2 (S_2) are in the same cluster. If we create the cluster union set as mentioned earlier (i.e., by taking the union of all synonym words appearing in all sentences in that cluster), we obtain $M_j = \{word_1, word_2, word_3, word_4, word_5\}$. If we now calculate the semantic similarity between M_j and S_1 by using the cosine measure, we then obtain the vectors $\mathbf{v}_j = \{1,1,1,1,1\}$ and $\mathbf{v}_1 = \{1,1,1,0,0\}$, which have a similarity score equal to 0.77. An issue is clearly seen here, since S_1 and S_2 are not similar and their centroid would not carry any useful meaning. This issue in which we would not expect the similarity value like this has happened due to all of the words of S_1 already existing in the cluster centroid M_j. We can solve this problem by defining the centroid using all sentences in the cluster except the sentence with which the cluster centroid is being currently compared. Therefore, assuming that we have a cluster containing sentences $\text{Sentence}_1 \ldots \text{Sentence}_N$, and we want the similarity between this cluster and a sentence SG appearing in the cluster, we would determine the cluster centroid using only the words appearing in $Sentence_1 \cup Sentence_2 \cup \ldots \cup Sentence_{G-1} \cup_{G+1} \cup \ldots \cup Sentence_N$; that is, we omit SG in calculating the cluster centroid.

2.6. Space and time complexity of CBLC algorithm

It has been founded that the proposed algorithm is no more expansive comparing with the basic *k*-means [52] and spectral-clustering [18, 37] algorithms regarding the space complexity (i.e., the three algorithms require the storage of the same similarity scores). The time (i.e., computation) complexity of a new version of the standard *k*-means: however, far exceeds that of basic *k*-means; and spectral-clustering algorithms. Furthermore, the computation complexities appeared in the stage of calculating the similarity between each sentence and corresponding centroid; this is due to representation of the text in the sentence similarity measure we have been applied within this clustering algorithm. To demonstrate this complexity, suppose that operation time unit for calculating semantic similarity between each sentence and cluster centroid is *SentSim*, the operation time unit for recalculating cluster centroids is *ReTime*, the total number of sentences in the used dataset is *tn*, the number of clusters is *k*, and the iteration loop of the proposed algorithm is *LoopI*. Therefore, essentially, the two following computations are required for each and every clustering iteration: (i) *tn.k* times sentence to cluster centroid similarity calculation; (ii) *k* times for

relocate cluster centroid. As a result, the time complexity of proposed version can be defined as $O_{CBLC} = (SentSim.\ tn.\ k + ReTime.\ k).\ LoopI$.

Since $SentSim >> ReTime$ and $tn >> k$, the overall time complexity of CBLC algorithm is found $O(tn)$, which means that computational complexity is relative to the size of the dataset that needs to be clustered.

3. Experiments and results

This section presents the performance of the CBLC algorithm to seven benchmark datasets, and the results are compared with that of other well-known clustering algorithms; spectral clustering [18, 36], affinity propagation [35], *k*-medoids algorithm [30, 31], STC-LE [39], and *k*-means (TF-IDF) [40]. We first describe the seven benchmark datasets, discuss cluster evaluation criteria, and we then report the experimental results (**Figure 2**).

3.1. Benchmark datasets

While CBLC algorithm is obviously appropriate to tasks involving sentence clustering, the algorithm is applied to generic in nature standard datasets such as *Reuters-21,578* dataset [29], *Aural Sonar* dataset [29, 53], *Protein* dataset [29, 54], *Voting* dataset [29, 55], *SearchSnippets* [38, 56], *StackOverflow* [38], and *Biomedical* [38].

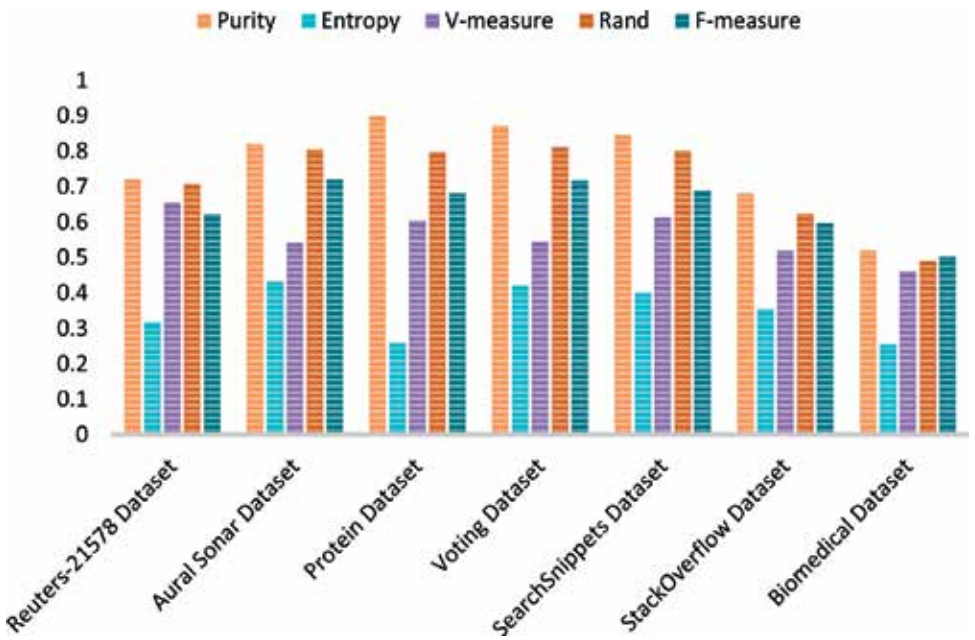

Figure 2. CBLC algorithm performance on seven benchmark datasets.

The Reuters-21,578 is the commonly used dataset for text classification task. It contains more than 20,000 documents from over 600 classes. The experimental results presented in this chapter only use a subset containing only 1833 text fragments, each of them are labeled as relating to one of 10 distinguished classes. The total number of the text fragments in each of the 10 classes is 354, 333, 258, 210, 155, 134, 113, 100, 90, and 70, respectively.

In the Aural Sonar dataset [53], two randomly selected people were asked to assign a similarity score between 1 and 5 to all pairs of signals returned from a broadband active sonar system. The two obtained scores from participated people were added to produce a 100×100 similarity matrix with values ranging from 2 to 10.

The Protein dataset [54, 57] consists of dissimilarity values for 226 samples over nine classes. We use the reduced set [57] of 213 proteins from four classes that result from removing classes with fewer than seven samples.

The Voting dataset is a two-class classification task with around 435 samples (text fragments). Similarity scores in the form of a matrix table were computed from the data in the categorical domain.

The SearchSnippets dataset consists of eight different predefined domains (i.e., classes), which was generated from the web-search-transaction result activity.

The StackOverflow dataset consists of 3,370,528 samples collected through the period of July 31, 2012, to August 14, 2012 (https//:www.kaggle.com). In this chapter, we randomly select 20,000 question titles from 20 different classes.

The Biomedical is a challenge dataset published in BioASQ's official website, and we randomly select 20,000 paper titles from 20 different MeSH major classes.

3.2. Clustering evaluation criteria

Since *complete* cluster (i.e., all objects from a single class are assigned to a single cluster) and *homogeneous* cluster (i.e., each cluster contains only objects from a single class) are hardly achieved, we aim to reach a satisfactory balance between these two approaches. Therefore, we apply five well-known clustering criteria in order to evaluate the performance of the proposed algorithm, which are *Purity, Entropy, V-measure, Rand Index,* and *F-measure*.

Entropy and *Purity* [58]. Entropy measure is used to show how the clusters of sentences are partitioned within each cluster, and it is known as the average of weighted values in each cluster entropy over all clusters $C = \{c_1, c_2, c_3, \ldots c_n\}$:

$$Entropy = \sum_{j=1}^{|L|} \frac{|w_j|}{N} \left(-\frac{1}{\log|C|} \sum_{i=1}^{|C|} \frac{|w_j \cap c_i|}{|w_j|} \log \frac{|w_j \cap c_i|}{|w_j|} \right) \tag{2}$$

The purity of a cluster is the fraction of the cluster size that the largest class of sentences assigned to that cluster represents, that is,

$$P_j = \frac{1}{|w_j|}\max_i \left(|w_j \cap c_i|\right) \tag{3}$$

Overall purity is the weighted sum of the individual cluster purities and is given by

$$Purity = \frac{1}{N}\sum_{j=1}^{|L|}\left(|w_j| \times P_j\right) \tag{4}$$

While purity and entropy are useful for comparing clusterings with the same number of clusters, they are not reliable when comparing clusterings with different numbers of clusters. This is because entropy and purity perform on how the sets of sentences are partitioned within each cluster, and this will lead to homogeneity case. Highest scores however, of purity and lowest scores of entropy are usually obtained when the total number of clusters is too big, where this step will lead to being lowest in the completeness. The next measure we have used considers both completeness and homogeneity approaches.

V-measure [59]. This is a measure that is known as the homogeneity and completeness harmonic mean; that is, *V* = *homogeneity* * *completeness* / (*homogeneity* + *completeness*), where *homogeneity* and *completeness* are defined as *homogeneity* = 1 – *H*(*C*|*L*)/*H*(*C*) and *completeness* = 1 – *H*(*L*|*C*)/*H*(*L*).

Eq. (5) can be written as follows, where

$$\begin{aligned} &H(C) = -\sum_{i=1}^{|C|}\frac{|c_i|}{N}\log\frac{|c_i|}{N}, \quad H(L) = -\sum_{j=1}^{|L|}\frac{|w_j|}{N}\log\frac{|w_j|}{N} \\ &H(C|L) = -\sum_{j=1}^{|L|}\sum_{i=1}^{|C|}\frac{|w_j \cap c_i|}{N}\log\frac{|w_j \cap c_i|}{|w_j|}, \quad \text{and} \quad H(L|C) = -\sum_{i=1}^{|C|}\sum_{j=1}^{|L|}\frac{|w_j \cap c_i|}{N}\log\frac{|w_j \cap c_i|}{|c_i|} \end{aligned} \tag{5}$$

Rand Index and *F-measure*. These measures depend on a combinatorial approach which considers each possible pair of sentences. It is defined as *Rand Index* = (*TP* + *FP*)/(*TP* + *FP* + *FN* + *TN*), where TP is a true positive (sentences corresponded to both same class and cluster), FP is a false positive (sentences corresponded to the different classes but same cluster), FP is a false positive (sentences corresponded to the different clusters but same class), and FN is a false negative (sentences must correspond to both different clusters and classes).

The *F*-measure is another method widely applied in the information retrieval domain and is defined as the harmonic mean of Precision (P) and Recall (R), that is, *F*-measure = 2**P***R*/(*P* + *R*), where *P* = *TP*/(*TP* + *FP*) and *R* = *TP*/(*TP* + *FN*).

3.3. Results

Since CBLC algorithm is generic in nature and can in principal be applied to any lexical semantic clustering domain, **Figure 3** shows the results of applying it to the *Reuters-21,578*, *Aural Sonar*, *Protein, Voting, SearchSnippets, StackOverflow,* and *Biomedical* datasets, respectively, by using the

http://dx.doi.org/10.5772/intechopen.75433

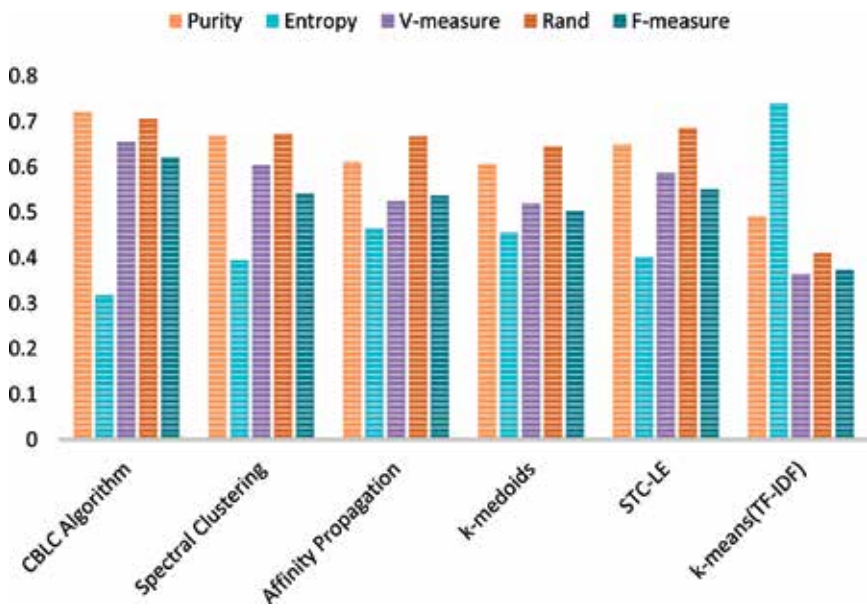

Figure 3. CBLC algorithm and other compared algorithms performance on Reuters-21,578 dataset.

Purity, Entropy, V-measure, Rand Index, and F-measure evaluation measures. CBLC algorithm however, requires an initial number of clusters in which we specified before the algorithm start. This number was varied from 7 to 12 for Reuters-21,578, Aural Sonar, Protein, Voting, and SearchSnippets datasets, and from 17 to 23 for StackOverflow and Biomedical datasets. This is because we found a proper clustering performance. Note that the values in the figure are averaged over 100 trials, and the best performance according to each measure is only presented.

Figures 3–9 show the clustering performance of CBLC algorithm comparing with that of spectral clustering, affinity propagation, *k*-medoids, STC-LE, and *k*-means (TF-IDF), respectively, on seven mentioned benchmark datasets using the five cluster evaluation criteria described earlier. For the baselined (i.e., compared) methods, the total values of the used evaluation measures (i.e., purity, entropy, V-measure, Rand Index, and F-measure) were in each measure obtained by discovering a range of numbers starting from 7 to 23 clusters and then considering that which performance is the best in overall clustering quality. The figured empirical results for our proposed new version of standard *k*-means clustering and other compared algorithms correspond to the best performance resulted from 200 time runs.

The empirical results demonstrate that CBLC algorithm significantly outperforms the other baselined algorithms on all used datasets. In this experiment however, we knew *a priori* what the real number of clusters was. Generally, we wish that the clustering algorithm could automatically determine an actual number of clusters, since we would not have this information. Even when run with a high initial number of clusters, CBLC algorithm was able to converge to a solution containing not more than seven clusters (e.g., in case of Reuters-21,578 dataset), and from the figures, it can be again seen that the evaluation of these clusterings is superior than that for the other baselined clustering algorithms.

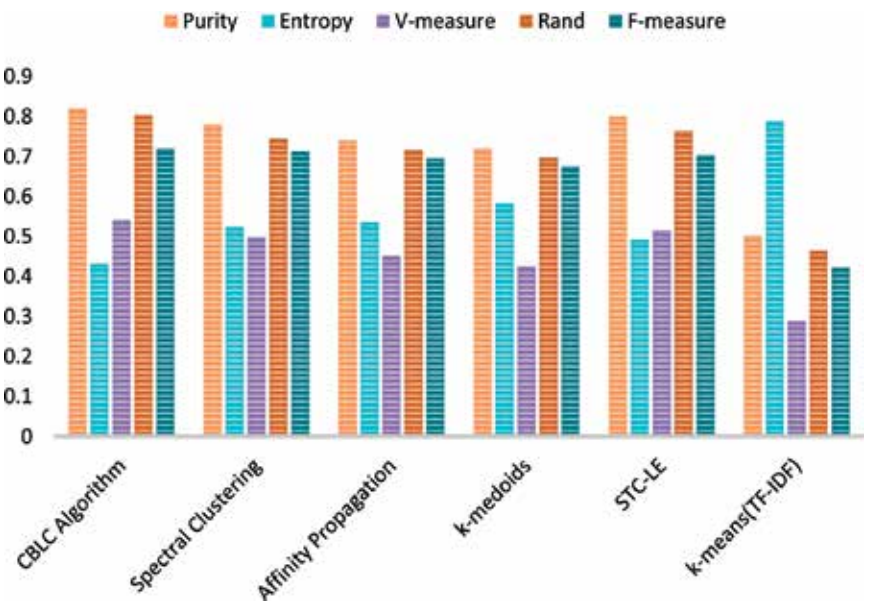

Figure 4. CBLC algorithm and other compared algorithms performance on aural sonar dataset.

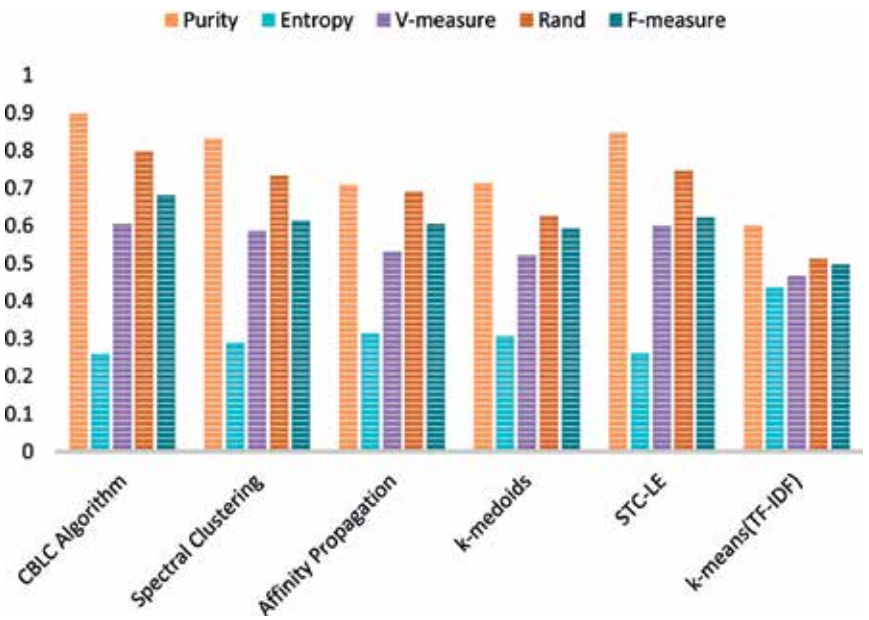

Figure 5. CBLC algorithm and other compared algorithms performance on protein dataset.

http://dx.doi.org/10.5772/intechopen.75433

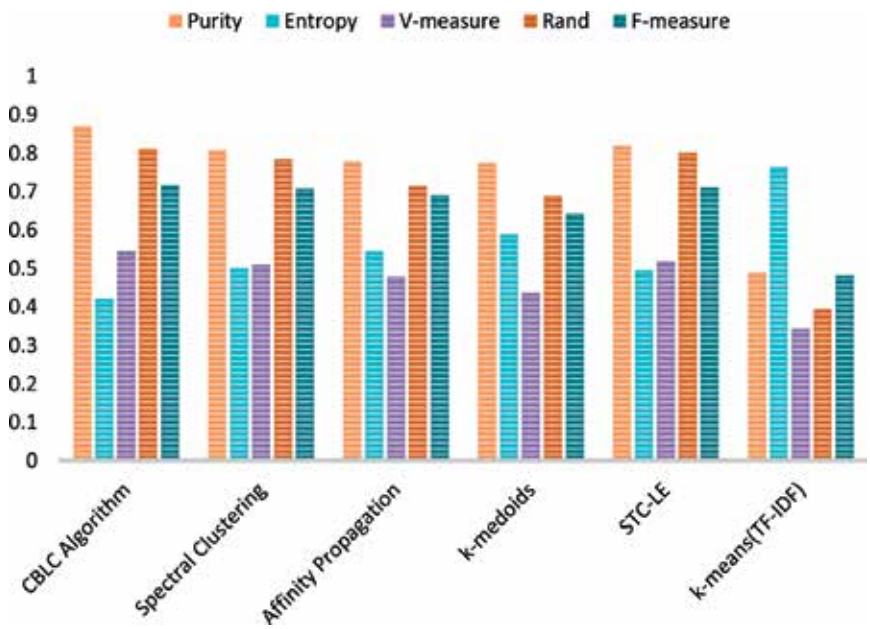

Figure 6. CBLC algorithm and other compared algorithms performance on voting dataset.

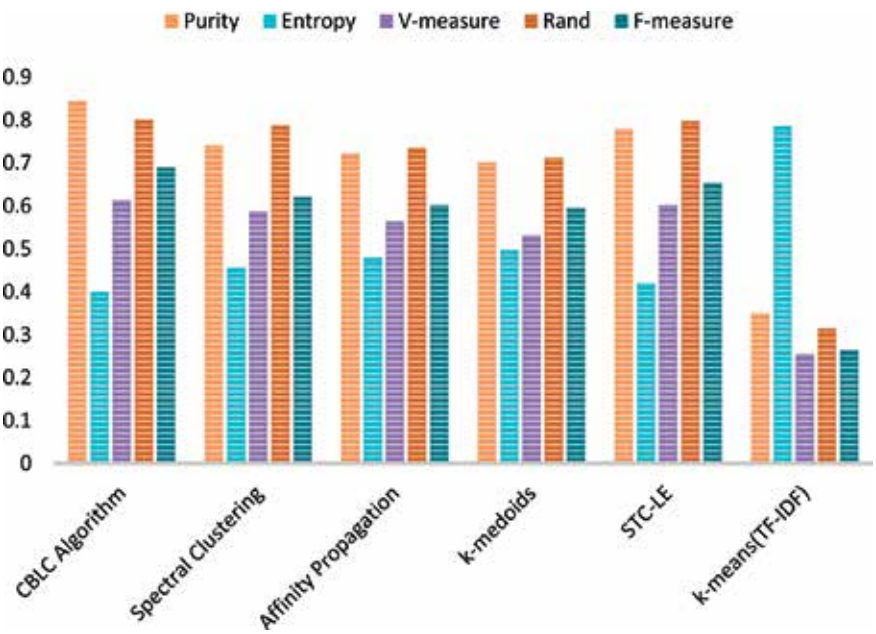

Figure 7. CBLC algorithm and other compared algorithms performance on SearchSnippets dataset.

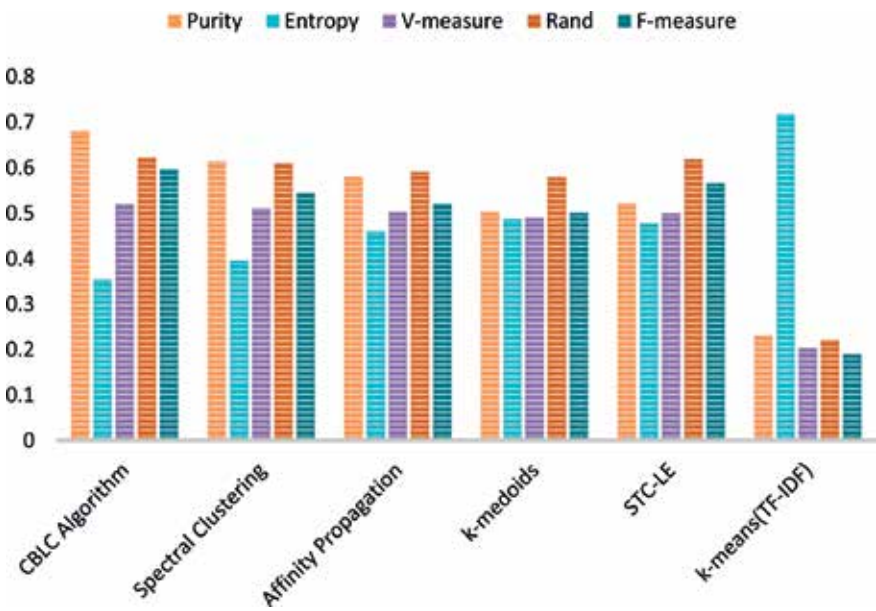

Figure 8. CBLC algorithm and other compared algorithms performance on StackOverflow dataset.

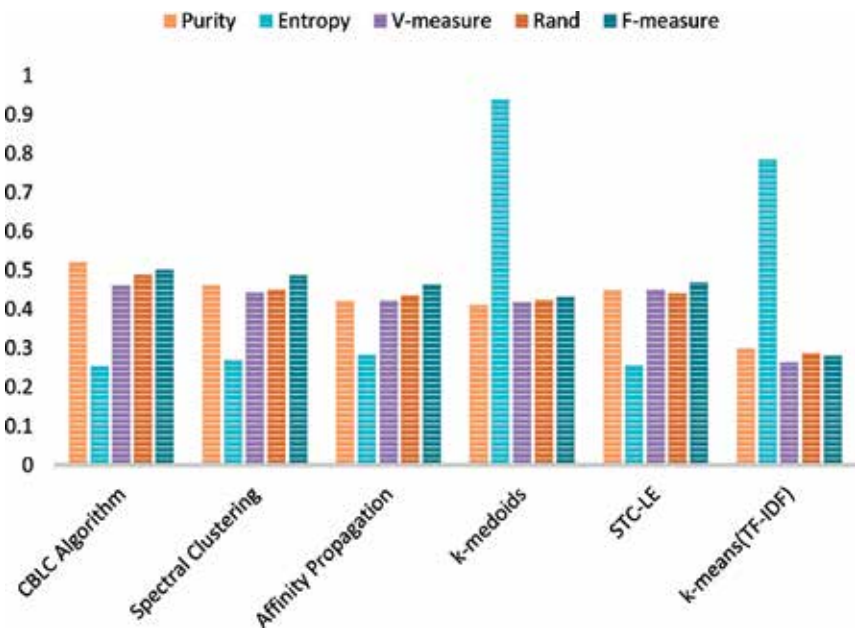

Figure 9. CBLC algorithm and other compared algorithms performance on biomedical dataset.

4. Concluding remarks

This chapter has shown a new version of the *k*-means clustering method that is able to cluster small-sized text fragments. This new variation measures the semantic similarity between patterns (i.e., sentences) based on the idea of creating a synonym expansion set to be used in the compared semantic vectors. The sentences are represented in these vectors by using semantic information derived from a WordNet that is created for the purpose of identifying the actual sense to a word, based on the surrounding context. The experimental results have demonstrated the method to achieve a satisfactory performance against the compared algorithms such as spectral clustering affinity propagation, *k*-medoids, STC-LE, and *k*-means (TF-IDF), as evaluated on several standard datasets.

A clear domain of applying the algorithm is to text-mining processing; however, the algorithm can also be used within more general text-processing settings such as text summarization. Like any clustering algorithm, the performance of CBLC will eventually be based on the text similarity values, and these values can be improved by defining the sentence-level text similarity measure that can utilize much more possible semantic information expressed with the compared sentences. Any such improvements are surly effected by the overall sentences clustering performance.

Sentence-level text clustering is an exciting area of research within the knowledge discovery and computational linguistic activities, and this chapter has proposed a new variation of *k*-means clustering which are capable to cluster sentences based on available semantic information written in these sentences. We are interested in some of the new research directions that we have experienced in this area; however, what we are most excited about is applying our proposed cluster technique to operate on the text-mining activities. This is because the concepts existing in human-written documents usually have buried knowledge and information, whereas the technique we have developed in this work is only applied on the clusters text-fragments domain. Therefore, one of the possible future works is to apply these ideas of sentence clustering to the development of complete techniques for sentiment analysis of the people's opinion.

Author details

Khaled Abdalgader

Address all correspondence to: komar@soharuni.edu.om

Sohar University, Sohar, Oman

References

[1] Li Y, McLean D, Bandar ZA, O'Shea JD, Crockett K. Sentence similarity based on semantic nets and Corpus statistics. IEEE Transactions on Knowledge and Data Engineering. 2006;**18**(8):1138-1150

[2] Mihalcea R, Corley C, Strapparava C. Corpus-based and knowledge-based measures of text semantic similarity. In: Proceedings of the 21st National Conference on Artificial Intelligence; 2006. pp. 775-780

[3] Sowmya V, Vishnu Vardhan B, Bhadri Raju MSVS. Influence of token similarity measures for semantic textual similarity. In: Proceedings of 2016 IEEE 6th International Conference on Advance Computing (IACC2016); 2016. pp. 27-28

[4] Metzler D, Dumais S, Meek C. Similarity measures for short segments of text. In: Proceedings of the 29th European Conference on Information Retrieval. 4425, Springer, Heidelberg; 2007. 16-27

[5] Islam A, Inkpen D. Semantic text similarity using corpus-based word similarity and string similarity. ACM Transactions on Knowledge Discovery from Data (TKDD). 2008;**2**(2):1-25

[6] Feng J, Zhou Y-M, Martin T. Sentence similarity based on relevance. In: Proceedings of the IPMU'08; 2008. 832-839

[7] Ramage D, Rafferty A, Manning C. Random walks for text semantic similarity. In: Proceedings of ACL-IJCNLP 2009; 2009. 23-31

[8] Achananuparp P, Hu X, Yang C. Addressing the variability of natural language expression in sentence similarity with semantic structure of the sentences. In: Proceedings of PAKDD 2009. Bangkok; 2009. 548-555

[9] Ho C, Murad MAA, Kadir RA, Doraisamy SC. Word sense disambiguation-based sentence similarity. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, COLING '10. Stroudsburg, PA, USA. Association for Computational Linguistics; 2010. pp. 418-426

[10] Abdalgader K, Skabar A. Short-text similarity measurement using word sense disambiguation and synonym expansion. In: Proceedings of the 23rd Australasian Joint Conference on Artificial Intelligence. (AI2010, Adelaide, Australia). vol. LNAI 6464; 2011. pp. 435-444

[11] Liu H, Wang P. Assessing sentence similarity using WordNet based word similarity. Journal of Software. June 2013;**8**(6)

[12] Zhu TT, Lan M. ECNUCS: Measuring short text semantic equivalence using multiple similarity measurements. Second Joint Conference on Lexical and Computational Semantics (SEM), Volume 1: Proceedings of the Main Conference and the Shared Task, Atlanta, Georgia, June 13-14, 2013. pp. 124-131

[13] Kenter T, Rijke DM. Short text similarity with word embeddings. In: Proceedings of the 24th ACM international conference on information and knowledge management. In CIKM '15. ACM; 2015

[14] Abdalgader K. Text-fragment similarity measurement using word sense identification. International Journal of Applied Engineering Research. 2016;**11**(24):11755-11762

[15] Abdalgader K. Computational Linguistic Techniques for Sentence-Level Text Processing". PhD Dissertation. Department of Computer Engineering and Computer Science, La Trobe University; 2011

[16] Skabar A, Abdalgader K. Improving sentence similarity measurement by incorporating sentential word importance. In: Proceedings of the 23rd Australasian joint conference on artificial intelligence. (AI2010, Adelaide, Australia). Vol LNAI 6464. 2011. pp. 466-475

[17] Abdalgader K. Word sense identification improves the measurement of short-text similarity. In: Proceedings of the International Conference on Computing Technology and Information Management (ICCTIM2014), Dubai, UAE, Digital Library of SDIWC, ISBN: 978–0–9891305-5-4. 2014. pp. 233-243

[18] Wang D, Li T, Zhu S, Ding C. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In: proceedings of the 31st annual international ACM SIGIR conference on Research and Development in Information Retrieval. pp. 307-314; 2008

[19] Salton G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley: Reading, Mass; 1989

[20] Fellbaum C, editor. "WordNet: An Electronic Lexical Database". Cambridge, MA: MIT Press; 1998

[21] Navigli R, Ponzetto S. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial intelligence, 193, Elsevier. 2012. pp. 217-250

[22] Abdalgader K, Skabar A. Unsupervised similarity-based word sense disambiguation using context vectors and sentential word importance. ACM Transactions on Speech and Language Processing (TSLP). 2012;**9**(2)

[23] Chen F, Han K, Chen G. An approach to sentence selection based text summarization. In: Proceedings of IEEE TENCON02; 2008. pp. 489-493

[24] Kyoomarsi F, Khosravi H, Eslami E, Dehkordy PK, Tajoddin A. Optimizing text summarization based on fuzzy logic. Seventh IEEE/ACIS International Conference on Computer and Information Science, IEEE Computer Society. 2008. pp. 347-352

[25] Radev DR, Jing H, Stys M, Tam D. Centroid-based summarization of multiple documents. Information Processing and Management: AN International Journal. 2004;**40**:919-938

[26] Aliguyev RM. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. Expert Systems with Applications. 2009;**36**:7764-7772

[27] Hotho A, Nürnberger A, Paaß G. A brief survey of text mining. GLDV-Journal for Computational Linguistics and Language Technology. 2005;**20**:19-62

[28] Kosala R, Blockeel H. Web mining research: A survey. ACM SIGKDD Explorations Newsletter. 2000;**2**(1):1-15

[29] Skabar A, Abdalgader K. Clustering sentence-level text using a novel fuzzy relational clustering algorithm. IEEE Transactions on Knowledge and Data Engineering (TKDE) IEEE Computer Society Digital Library. 2013;**25**(1):62-75

[30] Kaufman L, Rousseeuw PJ. Clustering by means of medoids. In: Godge Y, editor. Statistical Analysis Based on the L_1 Norm. Amsterdam: North Holland/Elsevier; 1987. pp. 405-416

[31] Kaufman L, Rousseeuw PJ. Finding Groups in Data. Wiley; 1990

[32] Krishnapuram R, Joshi A, Liyu Y. A fuzzy relative of the k-Medoids algorithm with application to web document and snippet clustering". In: Proceedings of the IEEE Fuzzy Systems Conference; 1999. pp. 1281-1286

[33] Geweniger T, Zühlke D, Hammer B, Villmann T. Fuzzy variant of affinity propagation in comparison to median fuzzy c-means. In: Proceedings of the 7th International Workshop on Advances in Self-Organizing Maps. Springer-Verlag, Berlin, Heidelberg; 2009. pp. 72-79

[34] Geweniger T, Zühlke D, Hammer B, Villmann T. Median fuzzy C-means for clustering dissimilarity data. Neurocomputing. 2010;**73**(7–9):1109-1116

[35] Frey BJ, Dueck D. Clustering by passing messages between data points. Science. 2007;**315**: 972-976

[36] Ng AY, Jordan MI, Weiss Y. On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems. 2001:849-856

[37] Luxburg UV. A tutorial on spectral clustering. Statistics and Computing. 2007;**17**(4):395-416

[38] Xu J, Xu B, Wang P, Zheng S, Tian G, Zhao J. Self-taught convolutional neural networks for short text clustering. Neural Networks. 2017;**30**(2):117-131

[39] Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in Neural Information Processing Systems. 2001;**14**:585-591

[40] Wagsta K, Cardie C, Rogers S, Schrodl S, et al. Constrained k-means clustering with background knowledge. In: ICML. Vol. 1. 2001. pp. 577-584

[41] Ganesan KA, Zhai CX, Han J. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10). 2010

[42] Abdalgader K. Clustering short text using a centroid-based lexical clustering algorithm. IAENG International Journal of Computer Science. 2017;**44**(4):523-536

[43] Erkan G, Radev DR. LexRank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research. 2004;**22**:457-479

[44] Mihalcea R, Tarau P. TextRank: Bringing order into texts. In: Proceedings of EMNLP. 2004. pp. 404-411

[45] Brin S, Page L. The anatomy of a large-scale Hypertextual web search engine. Computer Networks and ISDN Systems. 1998;**30**:107-117

[46] Fang C, Mu D, Deng Z, Wu Z. Word-sentence co-ranking for automatic extractive text summarization. Expert Systems with Applications. 2017;**72**:189-195

[47] Namburu SM, Tu H, Luo J, Pattipati KR. Experiments on supervised learning algorithms for text categorization. IEEE Aerospace Conference. Big Sky, MT. 2005

[48] Hatzivassiloglou V, Klavans JL, Holcombe ML, Barzilay R, Kan M-Y, McKeown KR. SIMFINDER: A flexible clustering tool for summarization. In: NAACL Workshop on Automatic Summarization. Association for Computational Linguistics, 2001. 41-49

[49] Vidhya KA, Aghila GG. Text mining process, techniques and tools: An overview. International Journal of Information Technology and Knowledge Management. 2010;**2**(2):613-622

[50] Bates M. Subject access in online catalogue: A design model. Journal of the American Society for Information Science. 1986;**37**(6):357-376

[51] Jiang JJ, Conrath DW. Semantic similarity based on Corpus statistics and lexical taxonomy. In: Proceedings of the 10th International Conference on Research in Computational Linguistics. 1997. pp. 19-33

[52] MacQueen JB. Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. 1967. pp. 281-297

[53] Philips S, Pitton J, Atlas L. Perceptual feature identification for active sonar echoes. Proceedings of IEEE OCEANS Conference. 2006

[54] Hofmann T, Buhmann JM. Pairwise data clustering by deterministic annealing. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1997;**19**(1):1-14

[55] Asuncion A, Newman DJ, UCI machine learning repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science

[56] Phan X-H, Nguyen L-M, Horiguchi S. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In: Proceedings of the 17th International Conference on World Wide Web, ACM. 2008. pp. 91-100

[57] Chen Y, Garcia EK, Gupta MR, Rahimi A, Cazzanti L. "Similarity-based classification: Concepts and algorithms". Journal of Machine Learning Research, vol. 10, pp. 747–776, 2009

[58] Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press; 2008

[59] Rosenberg A, Hirschberg J. V-Measure: A conditional entropy-based external cluster evaluation measure. In: Proceedings of the EMNLP. 2007. pp. 410-420

Chapter 5

Point Cloud Clustering Using Panoramic Layered Range Image

Masafumi Nakagawa

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.76407

Abstract

Point-cloud clustering is an essential technique for modeling massive point clouds acquired with a laser scanner. There are three clustering approaches in point-cloud clustering, namely model-based clustering, edge-based clustering, and region-based clustering. In geoinformatics, edge-based and region-based clustering are often applied for the modeling of buildings and roads. These approaches use low-resolution point-cloud data that consist of tens of points or several hundred points per m^2, such as aerial laser scanning data and vehicle-borne mobile mapping system data. These approaches also focus on geometrical knowledge and restrictions. We focused on region-based point-cloud clustering to improve 3D visualization and modeling using massive point clouds. We proposed a point-cloud clustering methodology and point-cloud filtering on a multilayered panoramic range image. A point-based rendering approach was applied for the range image generation using a massive point cloud. Moreover, we conducted three experiments to verify our methodology.

Keywords: point-cloud clustering, point-based rendering, terrestrial laser scanning, surface extraction, 3D edge extraction

1. Introduction

Massive point-cloud acquisition is an effective approach for 3D modeling of unknown objects in various fields, such as urban mapping, indoor mapping, plant management, factory management, heritage documentation, and infrastructure asset inspection and management. In construction fields, base maps and 3D data are required for managing processes of construction, maintenance, rehabilitation, and replacement. Online maps, such as Google Maps and

OpenStreetMap, are useful for approximate construction surveys in urban areas. However, online maps are often insufficient for infrastructure inspection to recognize the details of natural features. Thus, base maps and 3D data should be prepared before inspection. Massive point-cloud data can be acquired with a terrestrial laser scanner, mobile mapping systems (MMSs), handheld laser scanners, and cameras using structure from motion (SfM) methodology. SfM is a methodology for reconstructing a scene using multiple cameras simultaneously from all available relative motions through key point detection, feature matching, motion estimation, triangulation, and bundle adjustment. In an open sky environment, aerial photogrammetry and SfM using an unmanned aerial vehicle (UAV) are more effective than ground-based scanning. On the other hand, when environments include natural obstacles, such as trees, a terrestrial laser scanner is more effective than a UAV or MMSs. In indoor navigation and building information modeling (BIM), floor maps and 3D data are also required. We expected terrestrial laser scanners and indoor MMSs to be adequate for colored point-cloud acquisition in an indoor environment.

Moreover, point-cloud clustering is an essential technique for modeling massive point clouds. **Figure 1** shows an example of point-cloud clustering using a terrestrial laser scanner data acquired in an indoor environment.

There are three clustering approaches in point-cloud clustering, namely model-based clustering [1], edge-based clustering [2], and region-based clustering [3]. Model-based clustering is a 3D model preparation approach. The model-based clustering requires 3D models such as CAD models to estimate simple objects or point clusters from the point cloud. In 3D industrial modeling, standardized objects, such as pipes, boxes, and parts, are prepared as CAD models in advance. Although the model-based clustering is suitable for modeling known objects such as the standardized objects, the model-based clustering is unsuitable for modeling unknown objects such as complex and natural objects. On the other hand, in modeling unknown objects, such as buildings and roads in geoinformatics and civil engineering

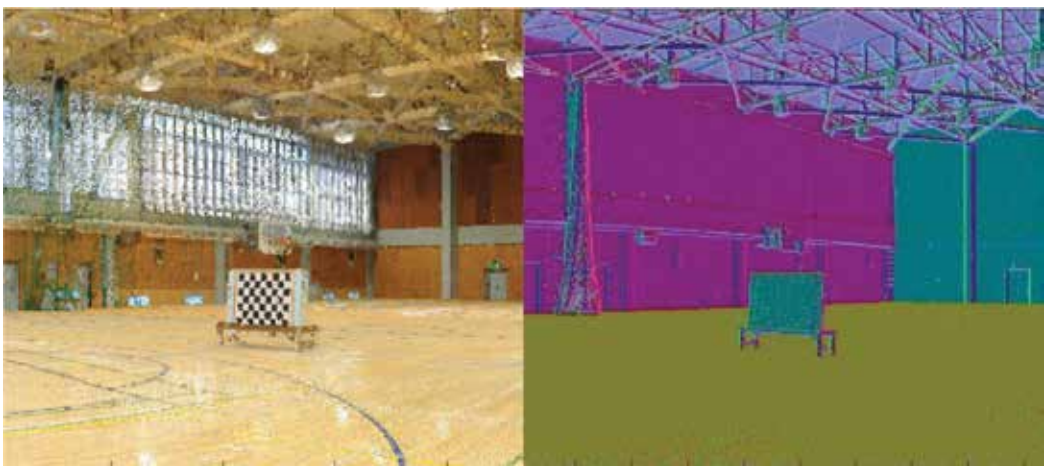

Figure 1. Point-cloud clustering: colored point cloud (left image) and clustered point cloud (right image).

fields, edge-based and region-based clustering are often applied [4]. These approaches use low-resolution point-cloud data that consist of tens of points or several hundred points per m^2, such as aerial laser and vehicle-borne MMSs data. These approaches also focus on geometrical knowledge [5] and 2D geometrical restrictions, such as the depth from a platform [6] and discontinuous point extraction on each scanning plane from the MMSs [7] to extract features. In urban areas and indoor environments, although there are simple features consisting of lines and planes, there are many complex features consisting of curved lines and surfaces with unclear boundaries. Moreover, point-cloud data are generally acquired with a terrestrial and mobile laser scanner from many viewpoints and view angles for 3D modeling. Like the conventional approaches, range image processing is proposed to apply 2D restrictions with an interactive procedure in 3D plant modeling. However, viewpoints for range image rendering are limited to data acquisition points.

Thus, our aim was to improve region-based point-cloud clustering in modeling after point-cloud integration. We also focused on region-based point clustering to extract a polygon from a massive point cloud, because it is not easy to estimate accurate edges from point clouds acquired with a laser scanner. In region-based clustering, random sample consensus (RANSAC) [8] is a suitable approach for surface detection and estimation. However, local work space should be selected to improve performance in a surface estimation from a massive point cloud. Moreover, it is hard to determine whether a point lies inside or outside a surface with conventional RANSAC.

In this chapter, we first proposed a point-cloud clustering methodology on a panoramic layered range image generated with point-based rendering from a massive point cloud. Next, we conducted three experiments to verify our methodology. The first experiment was a 3D edge and surface extraction for indoor modeling using an indoor MMS. The second experiment was a 3D edge and surface extraction for 3D bridge modeling using a terrestrial laser scanner. The third experiment was a 3D edge and surface extraction for ground surface and feature extraction using a terrestrial laser scanner. Even though the acquired data had low homogeneity of spatial point density, these experiments confirmed that a terrestrial laser scanner could cover complex surfaces, including flat surfaces, slopes, and steps. We also confirmed that our proposed methodology could achieve point-cloud clustering to extract these features from complex environments.

2. Methodology

Our proposed processing flow for point-cloud clustering is shown in **Figure 2**. First, we register and integrate point-cloud data acquired from a viewpoint. Next, the point-cloud data are projected into the image space with translation, view angle, and resolution parameters in "Panoramic multilayered range image generation with point cloud rendering" to generate several range images. Then, normal vectors around each projected point are estimated using the 3D coordinate values of the point cloud in "Normal vector estimation in panoramic multilayered range image." Next, edges are extracted from depth images generated in the panoramic

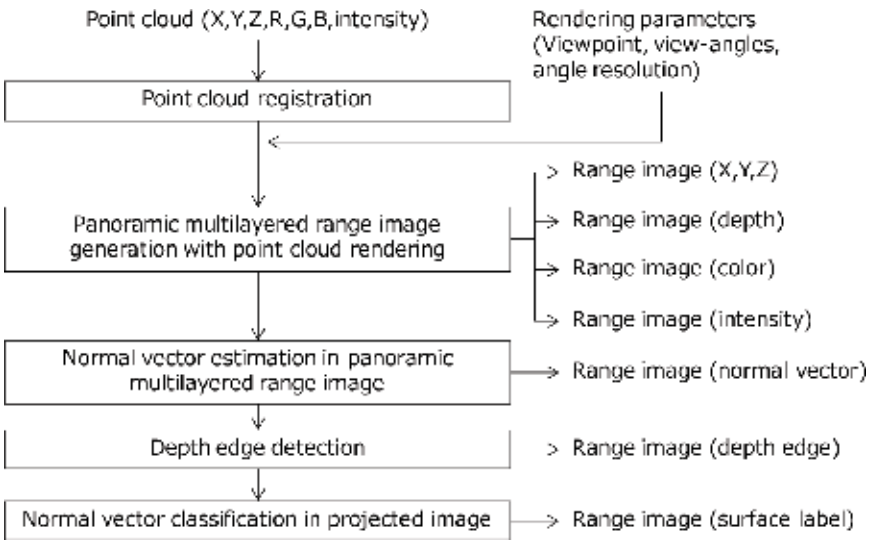

Figure 2. Processing flow for point-cloud clustering.

multilayered range image generation in ”Depth edge detection.” Finally, groups that have similar direction in the point cloud are extracted after normal vector classification in the projected image in “Normal vector classification in projected image.” Generated range images are managed in a multilayered range image.

2.1. Panoramic multilayered range image generation with point-cloud rendering

An advantage of 3D point-cloud data is that they allow accurate display from an arbitrary viewpoint and 3D modeling. Additionally, point-cloud data have the potential for applications such as panoramic image geo-referencing and distance value-added panoramic image processing for 3D geographical information system (GIS) visualization [9, 10]. However, point-cloud visualization has two technical issues. The first issue is the near-far effect caused by distance differences from the viewpoint to scanned points. The second issue is the transparency effect caused by rendered hidden points. These effects degrade the quality of point-cloud visualization. Thus, we focus on methodologies to improve the quality of point-cloud visualization. The Splat-based ray tracing [11] can generate a photorealistic curved surface from point-cloud data; surface generation requires the long period in the 3D work space. Moreover, the curved-surface description is unsuitable for representing urban objects such as CAD and GIS data. Therefore, we have applied a point-based rendering and filtering, which we call a layered image-based depth arrangement refiner for versatile rendering (LIDAR VR) [12] for point-cloud rendering.

The processing flow of LIDAR VR is described as follows. First, the sensors acquire a point cloud with additional color data such as RGB data. The sensor position is defined as the origin point in a 3D work space. Second, a multilayered range image from the simulated viewpoint is generated using the point cloud. Finally, the generated multilayered range image is filtered

http://dx.doi.org/10.5772/intechopen.76407

to generate missing points in the rendered result using distance values between the viewpoint and objects. The colored point cloud is projected from a 3D space to a panorama space. This transformation simplifies viewpoint translation, filtering, and point-cloud browsing. The LIDAR VR data consist of a projection model and multilayered range image, as shown in **Figure 3**. The panorama model can be selected from a spherical, cylindrical, plane, hemispherical, or cubic model. In this chapter, the spherical model is described. First, the measured point data are projected onto a spherical surface as a range of data. Next, the measured point data are projected onto a spherical surface to manage X, Y, Z, R, G, B, and the intensity values in the multilayered panorama space. Then, azimuth and elevation angles are calculated using 3D vectors generated from the viewpoint and the measured points. The azimuth and elevation angles are converted to panorama image coordinate values with adequate spatial angle resolution in the range data. Finally, a spherical panorama image is generated from the measured point cloud.

Based on this panorama projection, the multilayered range data with a translated viewpoint are generated using the point cloud, as shown in **Figure 4**. When a panorama space is generated using points from P_1 to P_{10} from a viewpoint X_o the points from P_1 to P_{10} are continuously

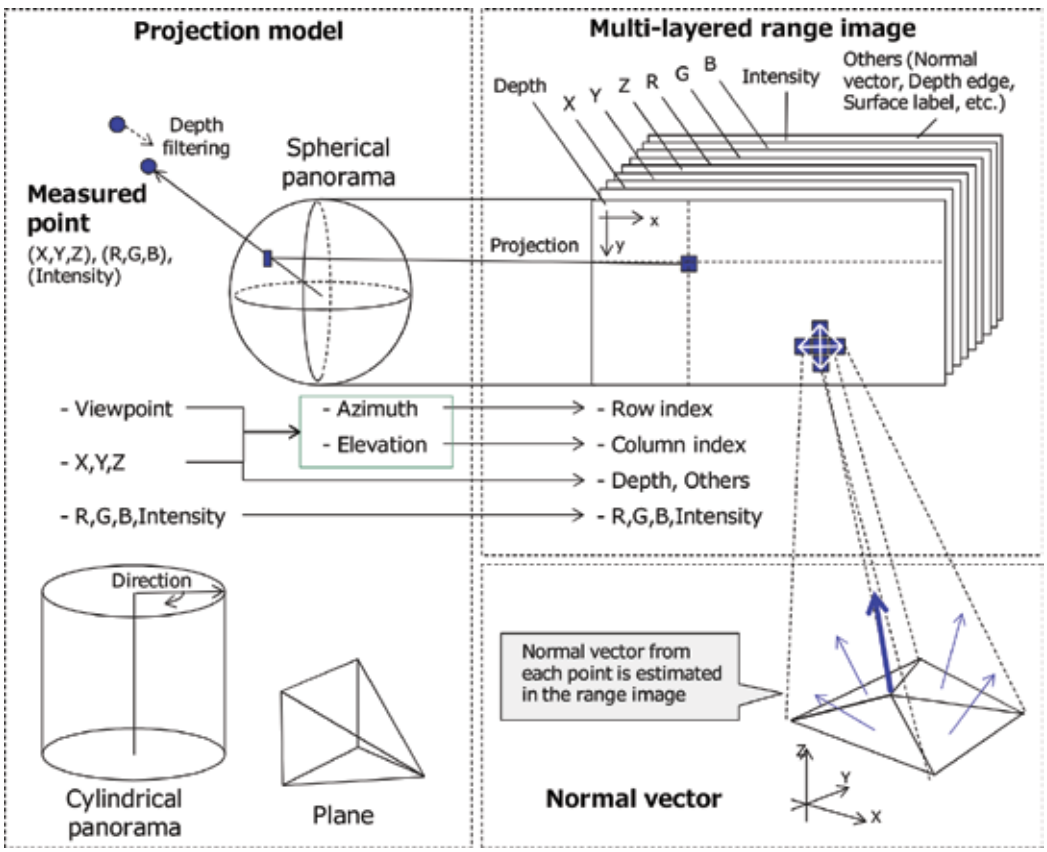

Figure 3. LIDAR VR data model.

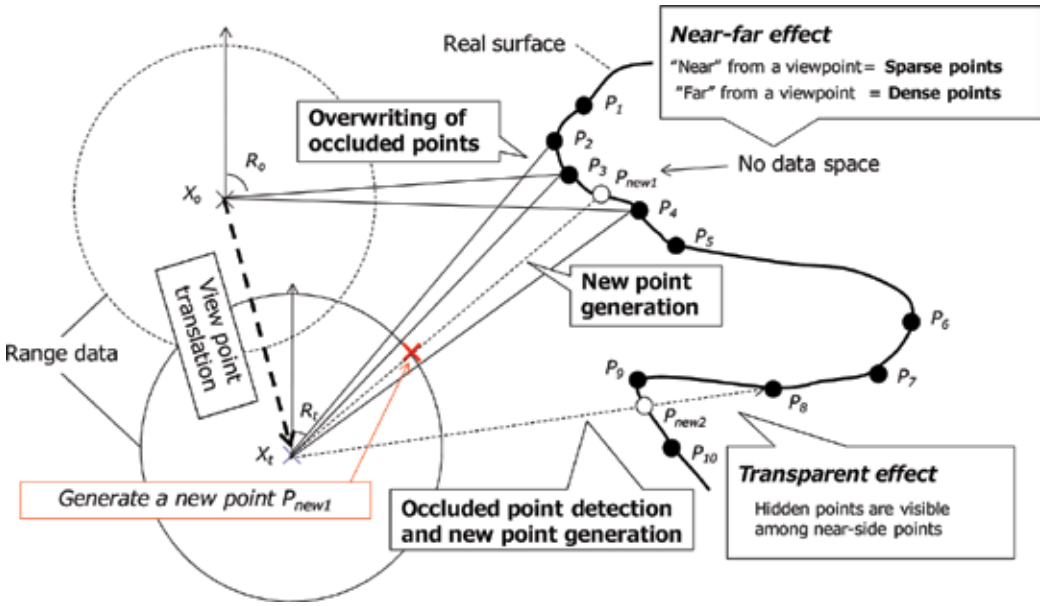

Figure 4. Point distribution calculated by viewpoint translation in range data, occlusion detection using the point cloud, and point-cloud interpolation with distance information.

arranged in the range data. An azimuth or elevation angle from a viewpoint X_o to a measured point P_1 is denoted as R_o. On the other hand, when a panorama space is generated using the points from P_1 to P_{10} from a different viewpoint X_t, the angle from the viewpoint X_t to the measured point P_1 is denoted as R_t. Thus, the change in angle from R_o to R_t affects the arrangement of the projected points in the range data.

After the viewpoint translation, three types of filtering are applied to point-cloud rendering, as shown in **Figure 4**. The first filtering is the occluded point overwriting. **Figure 4** shows an example to overwrite the projected point P_1 by the projected point P_2. After the viewpoint translation from X_o to X_t, the projected point P_1 becomes an occluded point behind P. Thus, **Figure 4** shows that P_1 is overwritten by P_2. The second filtering is the new point generation in the no-data space. **Figure 4** also shows an example to generate a new point P_{new1}. After the viewpoint translation from X_o to X_t, a no-data space occurs between the projected points P_3 and P_4. Therefore, **Figure 4** shows that P_{new1} is generated between the projected points P_3 and P_4. The third filtering is the occluded point replacement. **Figure 4** shows an example to replace an occluded point P_8 with a new point P_{new2}. After the viewpoint translation from X_o to X_t, the point P_8 is visible between points P_9 and P_{10}. However, the point P_8 exists behind the real surface. Thus, the occluded point P_8 should be given a new distance value as P_{new2}. The new distance value is calculated the distance values of points P_9 and P_{10} through the pixel-selectable averaging filter developed in this study, which we now describe.

In general image processing, each pixel value in an image is resampled by using pixel values around it when the image is transformed. A similar technique is applied to the pixel-selectable

averaging filter to improve the quality of the range data generated from point-cloud data with the view-point translation. However, general image resampling techniques, such as the nearest interpolation, linear interpolation, and cubic interpolation, degrade the quality of the range data because the resampling blends various data, such as valid points, occluded points, measurement noises, and missing data. Therefore, the pixel-selectable averaging filter is applied to this technical issue. The pixel-selectable averaging filter extracts valid points around a point for the resampling, as shown in **Figure 5**. This processing consists of a detection of valid data extraction and rejection of occluded points, noises, and missing data, and missing-point regeneration.

The detailed flow of the pixel-selectable averaging filter is described as follows. First, a three-by-three block of pixels is prepared in the range data projected from point clouds. The center point in the block is the focus point in the range data. Second, the block is checked to see whether valid points exist. When there are more than two valid pixels, the processing moves to the next step. If there is only the focus point, it is deleted as spike noise. When the focus point is a missing part, a new pixel value such as color and intensity value is given to the focus point using the other valid points around the center point in the block. Third, after these point extraction steps, an average value of valid points in the block within the search range is calculated to overwrite the focus point value. The average value is a distance from the viewpoint to the valid points. This processing is applied to each channel in the RGB and intensity image. However, when the center point in the block has a distance value within the search range, the overwriting processing is not performed. Because, when the point can be defined approximately as the nearest surface, the overwriting processing has a possibility of degrading geometrical accuracy and image quality in this case. This processing sequence is applied to all points.

In the valid point extraction, a range of search distances should be given. The distance from the viewpoint to the nearest point found among the extracted points is defined as the start

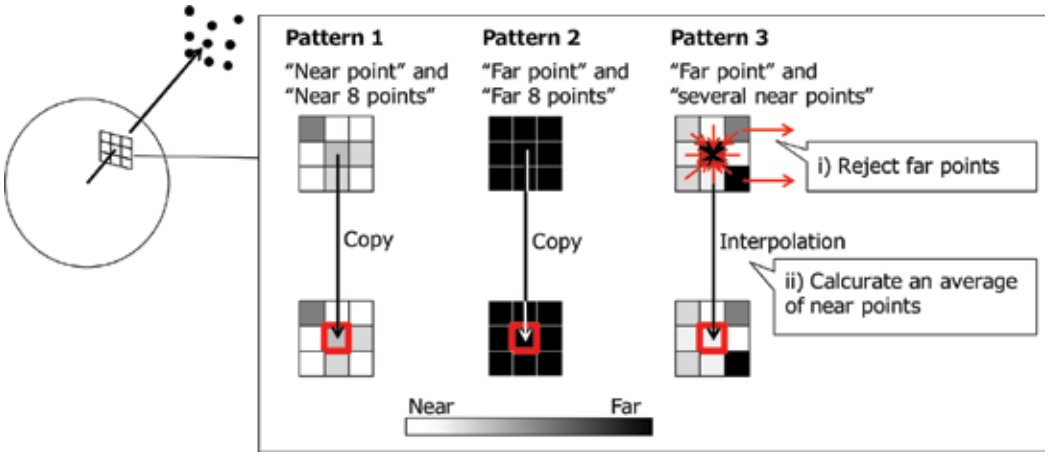

Figure 5. Pixel-selectable averaging filter.

value of the search range. Moreover, the start value plus a defined distance parameter is assumed as the end value. The defined distance parameter is determined with the continuity of the points in the point cloud. For example, the defined parameter would be between 10 cm and 1 m from experience, when trees and building walls are measured.

Thus, the pixel-selectable averaging filter uses valid points in the range data to achieve an interpolation without reducing geometrical accuracy by a uniform smoothing effect. **Figure 6** shows an example of processing result.

2.2. Normal vector estimation in a panoramic multilayered range image

A normal vector can be estimated using three points in point cloud with a triangle patch or mesh generation processing. In 2D image processing, the Delaunay division is a popular algorithm. The Delaunay division can also be applied for 3D point-cloud processing with millions of points [13]. However, using the Delaunay division, it is hard to generate triangle patches for more than hundreds of millions of points without a high-speed computing environment [14, 15]. Thus, we focused on point-cloud rendering that restricts visible point-cloud data as a 2D image. A closed point detection and topology assignment can be processed as 2D image processing.

In our normal vector estimation, four faces in a range image are generated to estimate normal vectors of a point in point cloud, as shown in **Figure 7**. First, a point in the projected point cloud on a panoramic layered range image is defined as point C. Next, the projected points P_1, P_2, P_3,

Figure 6. LiDAR VR processing result: input point cloud (upper image) and output point cloud (bottom image).

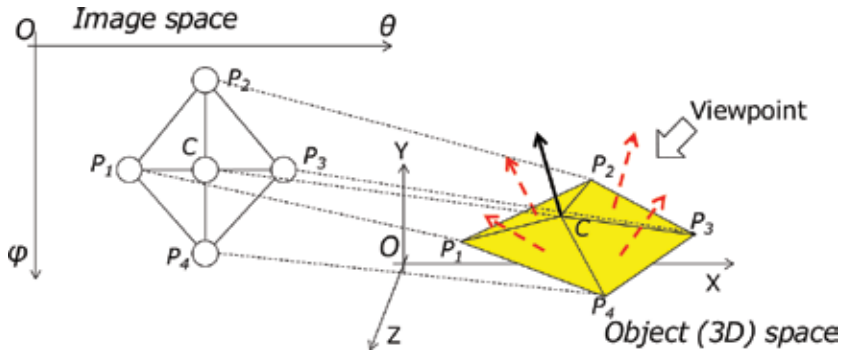

Figure 7. Normal vector estimation in panoramic multilayered range image.

and P_4 in the range image are set from point C with d_1, d_2, d_3, and d_4 pixels in vertical and horizontal directions. Triangulation is applied to these points as vertexes C-P_1-P_2, C-P_2-P_3, C-P_3-P_4, and C-P_4-P_1 with a clockwise topology in the image space. Moreover, parameters d_1, d_2, d_3, and d_4 in this procedure depend on the accuracy and resolution of the measurement data taken from the laser scanner or stereo camera. When the accuracy and resolution are high enough, these parameters are set as one pixel. These parameters are set to more than one pixel for low accuracy and resolution measurement data to keep a smooth condition of normal vectors on a flat surface. This procedure, which is based on 2D image processing, can provide a higher topology attachment to the point cloud.

Additionally, the normal vector on each triangle is estimated using the 3D coordinate values of each point. When five points consisting of a center and four vertex points exist on the same plane in 3D space, each normal vector has the same direction. When point C exists on the edge of the 3D space, two clusters can be classified by two directions. Moreover, when point C exists on the corner of the 3D space, each triangle has a different direction. Surfaces, edges, and corners in the 3D space were estimated in point-cloud data using these clues. In this research, we used the point cloud taken from a laser scanner that presents difficulties for measuring edges and corners clearly. Thus, the average value of each normal vector is used as a normal vector of point C. These procedures were iterated to estimate the normal vectors of all points in point cloud.

2.3. Normal vector-based point clustering

Point clusters are generated from a classification result of normal vectors. The accuracy of point-cloud classification can be improved with several approaches such as the Mincut, Markov network, and fuzzy-based algorithms [16–18]. However, in this study, we focused on verifying the practicality of our point-based rendering for point-cloud clustering. Thus, we applied multilevel slicing as a simple classification algorithm to classify normal vectors, as shown in **Figure 8**.

This classification detected boundaries of point clusters with the same normal vectors. Moreover, clustered normal vectors were compared with normal vectors of neighboring

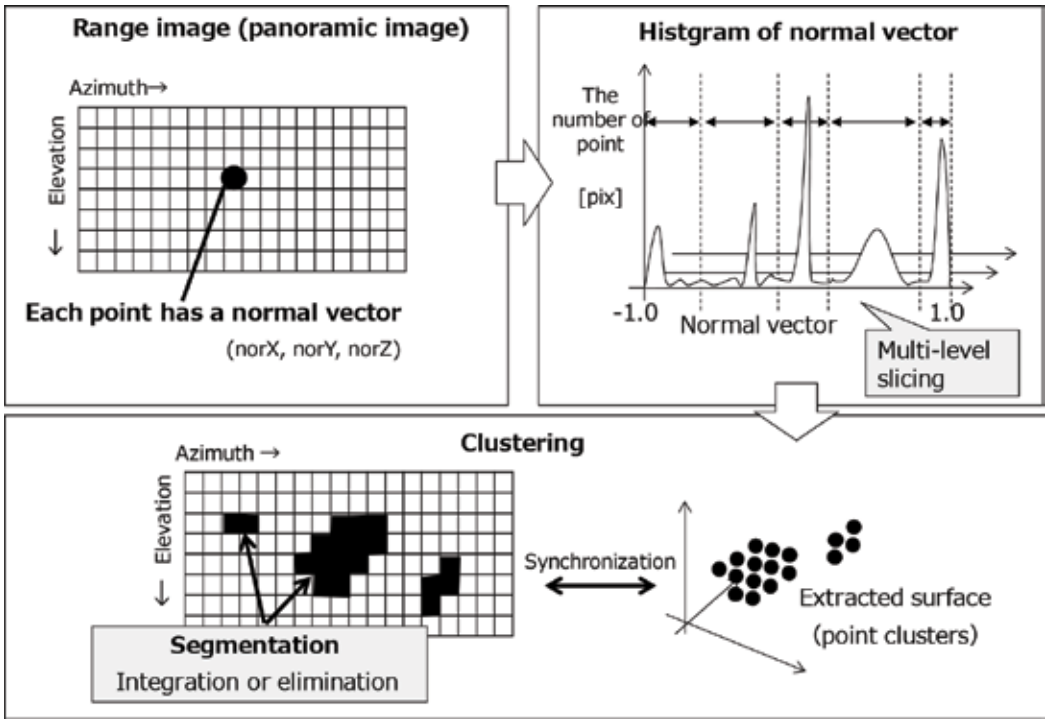

Figure 8. Normal vector-based point clustering.

planes to be integrated into a larger plane or deleted as a small segment. When a specified plane is extracted, the direction of a normal vector and the cluster number are available as initial value inputs. The point-cloud clustering methodology for extracting the intersection of planes as ridge lines requires appropriate initial values such as curvature, fitting accuracy and distances to closed points [19]. However, our approach can extract boundaries from a point cloud without these parameters.

3. Experiments

We conducted three experiments using point-cloud data acquired in indoor and outdoor environments. Here we present the point-cloud clustering results of these experiments.

3.1. Experiment 1: indoor MMSs data

We selected a floor in our university as the indoor environment. We prepared a point cloud taken from an indoor MMSs (TIMMS, Nikon-Trimble), which consisted of a laser scanner, an omni-directional camera, inertial measurement units (IMU), and a wheel encoder, as shown in **Figure 9**. Acquired point-cloud data, point-cloud rendering results,

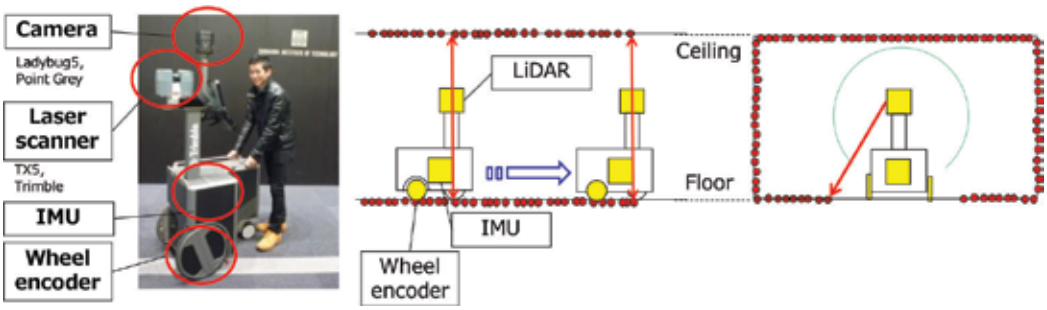

Figure 9. Indoor MMSs.

and point-cloud clustering results are shown in **Figure 10**. The results show that building features such as ceilings, beams, window shades, pillars, benches, and floors are classified clearly.

3.2. Experiment 2: terrestrial laser scanner data (1)

We selected a bridge as a study area in the outdoor environment. We acquired 25.9 million points using a terrestrial laser scanner (GLS-2000, TOPCON) from four viewpoints. Rendered point cloud, depth range image, and point-cloud clustering results are shown in **Figure 11**. The results show that vertical planes, horizontal planes, and natural features are classified clearly. The processing time for the clustering was 6.1 s (Intel Core i7-6567U 3.30 GHz, MATLAB).

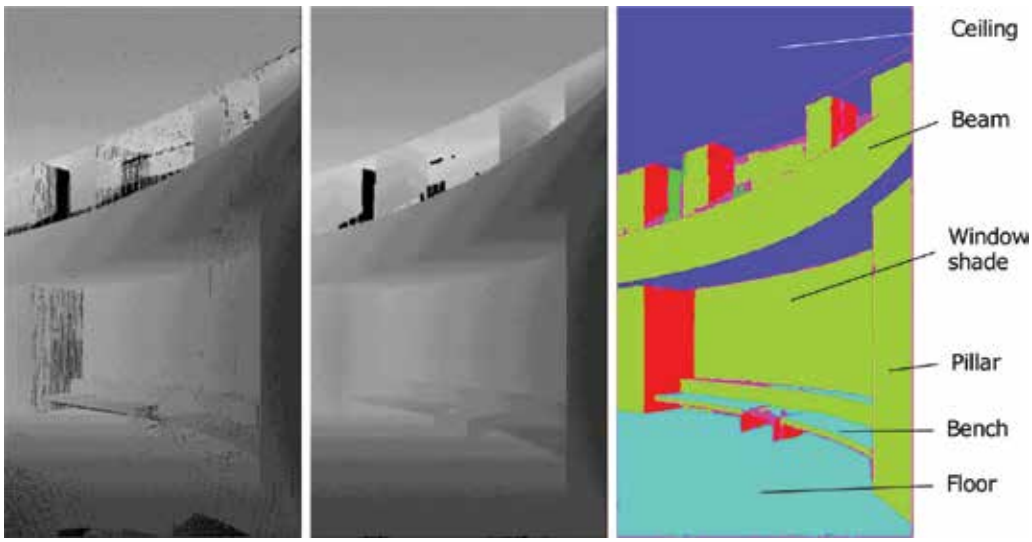

Figure 10. Point-cloud clustering result: rendered point cloud (left image), filtered point cloud (center image), and clustered point cloud (right image).

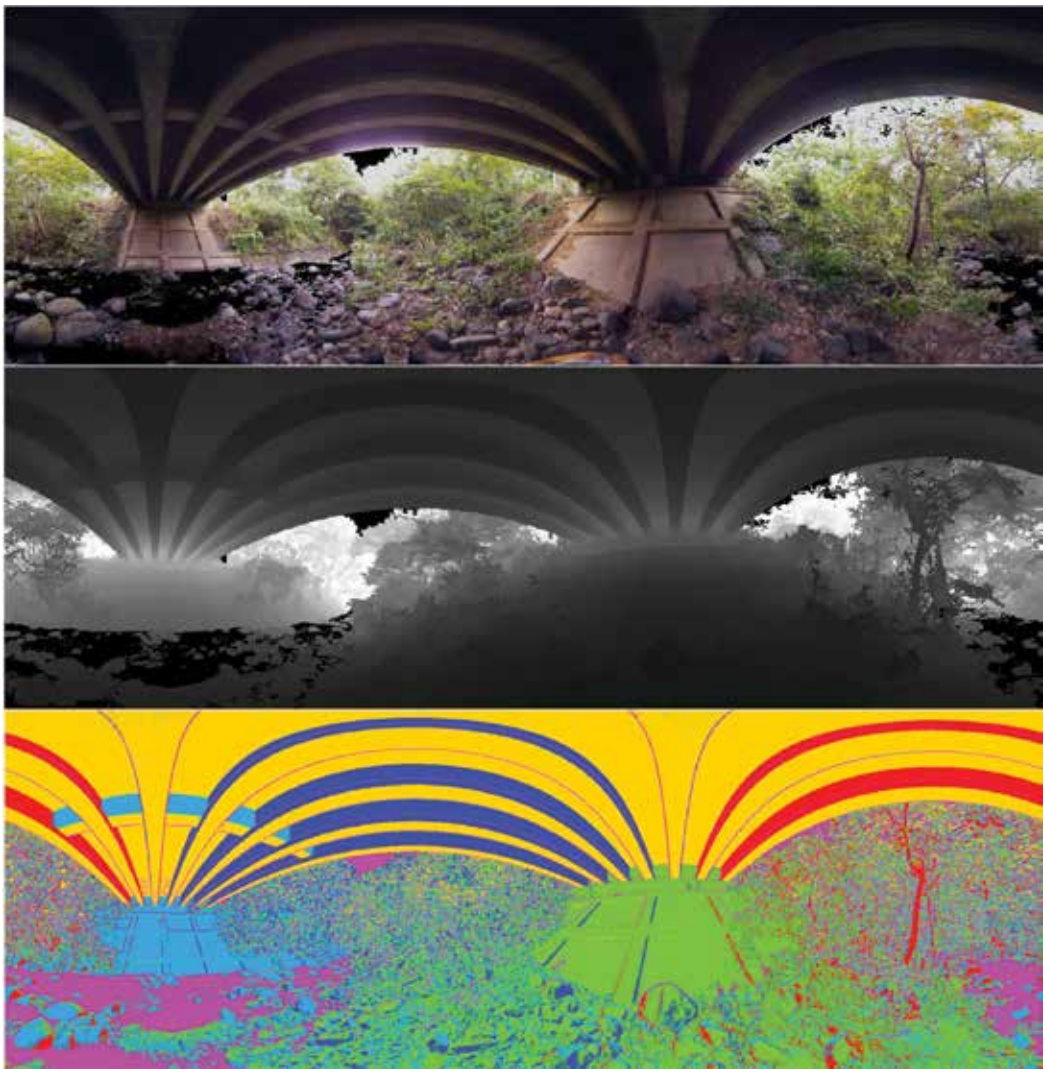

Figure 11. Point-cloud clustering result: colored point cloud (upper image), depth image (center image), and clustered point cloud (bottom image).

3.3. Experiment 3: terrestrial laser scanner data (2)

We selected a long narrow slope including slopes and stone steps as our study field. We prepared a point cloud acquired from 18 viewpoints over a wide area using a terrestrial laser scanner (VZ-400, RIEGL). Acquired point cloud and point-cloud clustering results are shown in **Figure 12**. The results show that features, such as steps, slopes, rock walls, and trees, are classified clearly. The processing time for the clustering was 11.4 s (Intel Core i7 2.80 GHz, MATLAB).

http://dx.doi.org/10.5772/intechopen.76407

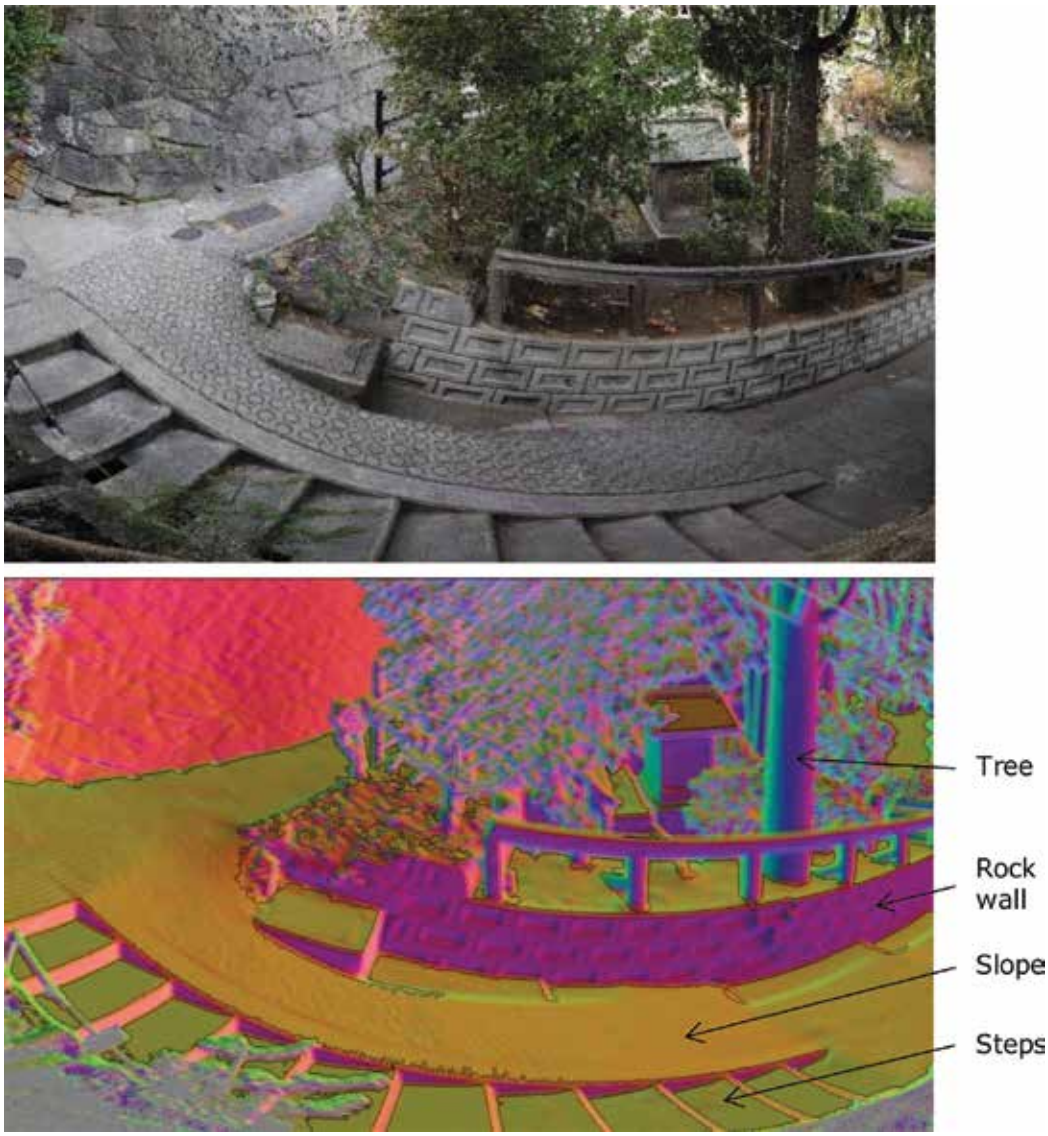

Figure 12. Point-cloud clustering result: colored point cloud (upper image) and clustered point cloud (bottom image).

4. Discussion

Our described processing flow in **Figure 2** can be extended from point-cloud clustering to polygon extraction [20], as shown in **Figure 13**.

After the "Normal vector classification in projected image," when a small region has a similar direction with the neighboring region, the small region is merged into the neighboring

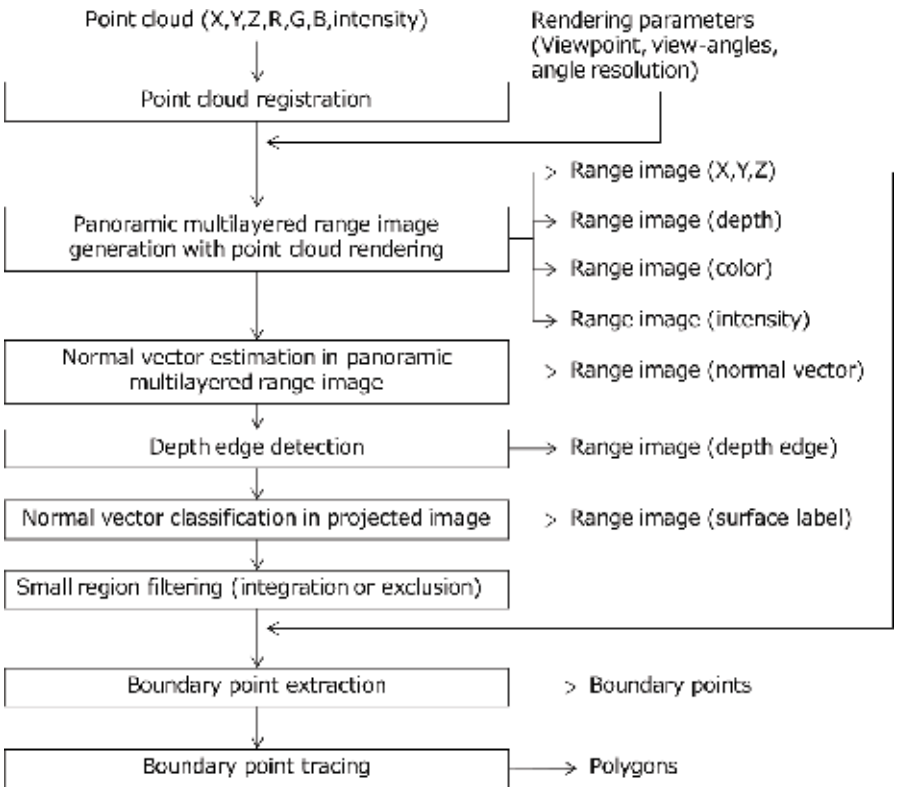

Figure 13. Overall processing flow.

region. Otherwise, the small region is deleted from the clustering result. Boundary points are extracted through "Boundary point extraction" from the clustering result. Moreover, polygons are extracted through "Boundary point tracing." Boundaries of features can be extracted from the refined surfaces in a range image. Moreover, 3D polygons can be extracted with topology estimation using these extracted boundaries in the range image. In this procedure, point tracing connects points in the 3D space along the boundary, as shown in **Figure 14**. Although least squares fitting and polynomial fitting are generally applied for straight and curved line extraction from points, these fitting approaches require a straight-line recognition or curved-line recognition. When the point clouds include noises, RANSAC is a suitable approach for feature estimation. However, the RANSAC also requires the fitting procedure. Thus, tracing based on the region growing is applied to complex geometry extraction, as follows. First, a topology of points is estimated in a range image. Continuous 3D points can be extracted when a polyline or polygon is drawn in a range image. Next, a seed point is selected from the continuous 3D points for point tracing. Then, a possible next point is searched within a candidate area. The candidate area is determined using a 3D vector from the seed point. When a point exists within the candidate area, it is connected to the seed point. Otherwise, the point is assumed to be an outlier. A position of the outlier is corrected to a suitable position using the 3D vector from the seed point. Then, the connected point is assumed as the next

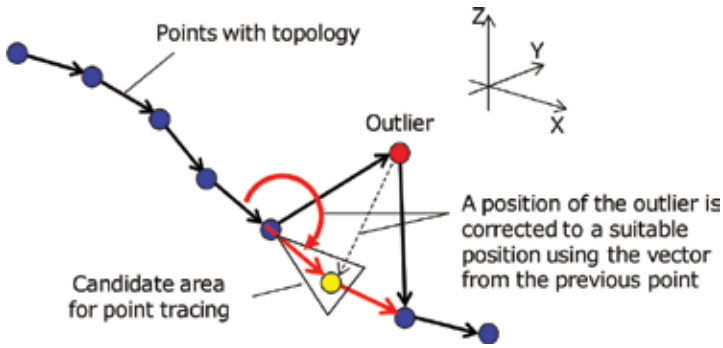

Figure 14. Point tracing.

seed point. These steps are iterated to close the geometry for the 3D smooth polygon generation. These procedures are applied to each rendered point cloud from arbitrary viewpoints.

In indoor navigation and BIM, terrestrial laser scanners and indoor MMSs are used for colored point-cloud acquisition to generate floor maps and 3D data in an indoor environment. **Figure 15** shows the result of polygon extraction from the point cloud used in the first experiment.

However, missing areas and a non-uniform density of the point cloud would exist in point-cloud acquisition. This issue causes transparent and near-far effects in point-cloud visualization. To avoid these effects, we have developed a spatial interpolation based on point-based rendering in the point-cloud visualization and modeling. Nevertheless, when large missing and occluded areas exist, the spatial interpolation approach is inefficient and ineffectual. Therefore, we focused on a randomized algorithm to quickly find approximate nearest-neighbor matches between image patches for image inpainting [21]. The image inpainting aims to improve image quality with deletion works of scratches and unnecessary objects in an image and reconstruction works of the natural image. That is, scratches and unnecessary objects are replaced by other textures in the image [22], as shown in **Figure 16**. In manual works, these objects are replaced using image retouching software such as Adobe Photoshop. The inpainting approach is an automated procedure for image retouches.

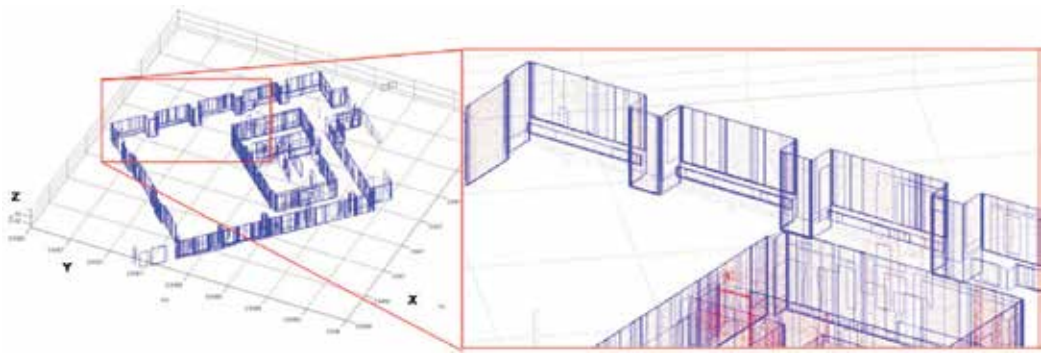

Figure 15. Polygon extraction result.

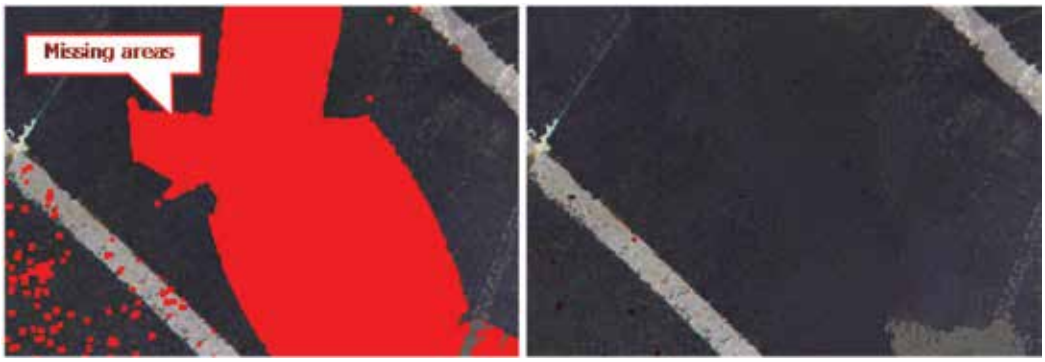

Figure 16. Inpainted result: rendered point cloud (left image) and inpainted point cloud (right image).

5. Conclusion

We have focused on improving region-based point-cloud clustering in 3D modeling after point-cloud integration. We have also focused on region-based point clustering to extract a polygon from a massive point cloud, because it is difficult to estimate accurate edges from point clouds acquired with a laser scanner. First, we proposed a point-cloud clustering methodology on a panoramic layered range image generated from a massive point cloud with point-based rendering. Next, we conducted three experiments using laser scanning data to verify our methodology. The first experiment was 3D edge and surface extraction for indoor modeling using an indoor MMS. The second experiment was 3D edge and surface extraction for 3D bridge modeling using a terrestrial laser scanner. The third experiment was 3D edge and surface extraction for ground-surface and feature extraction using a terrestrial laser scanner. The results of these experiments confirm that our proposed methodology can achieve point-cloud clustering to extract features such as flat surfaces, slopes, and steps from complex environments in practical processing times.

Acknowledgements

This work is supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Japan. This work is also supported by RIEGL Japan Co. Ltd., Nikon-Trimble Co. Ltd., and Sumire survey Co., Ltd. for data acquisition.

Author details

Masafumi Nakagawa

Address all correspondence to: mnaka@shibaura-it.ac.jp

Shibaura Institute of Technology, Tokyo, Japan

References

[1] Boyko A, Funkhouser T. Extracting roads from dense point clouds in large scale urban environment. ISPRS Journal of Photogrammetry and Remote Sensing. 2011;**66**(2011):S2-S12

[2] Jiang X, Bunke H. Edge detection in range images based on scan line approximation. Computer Vision and Image Understanding. 1999;**73**(2):183-199

[3] Vosselman G, Gorte BGH, Sithole G, Rabbani T. Recognising structure in laser scanning point clouds. In: ISPRS 2004: Proceedings of the ISPRS Working Group VIII/2: Laser Scanning for Forest and Landscape Assessment; 2004. pp. 33-38

[4] Tsai A, Hsu C, Hong I, Liu W. Plane and boundary extraction from LiDAR data using clustering and convex hull projection. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2010;**XXXVIII**(Part 3A):175-179

[5] Pu S, Vosselman G. Knowledge based reconstruction of building models from terrestrial laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing. 2009;**64**(6):575-584

[6] Zhou Q, Neumann U. Fast and extensible building modeling from airborne LiDAR data. In: ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS). 2008. p. 8

[7] Denis E, Burck R, Baillard C. Towards road modelling from terrestrial laser points. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2010;**XXXVIII**(Part 3A):293-298

[8] Schnabel R, Wahl R, Klein R. Efficient RANSAC for point-cloud shape detection. Computer Graphics Forum. 2007;**26**(2):214-226

[9] Shade J, Gortler S, He L, Szeliski R. Layered depth images. In: SIGGRAPH '98. 1998. pp. 231-242

[10] Verbree E, Zlatanova S, Dijkman S. Distance-Value-Added Panoramic Images as the Base Data Model for 3D-GIS. Panoramic Photogrammetry Workshop. 2005

[11] Linsen L, Müller K, Rosenthal P. Splat-based ray tracing of point clouds. Journal of WSCG. 2007;**15**(1-3):51-58

[12] Nakagawa M. Point cloud clustering for 3D modeling assistance using a panoramic layered range image. Journal of Remote Sensing Technology. 2013;**1**(3):10

[13] Chevallier N, Maillot Y. Boundary of a non-uniform point cloud for reconstruction. In: SoCG '11 Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry. 2011. pp. 510-518

[14] Fabio R, From point cloud to surface: The modeling and visualization problem. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2003;**XXXIV-5/W10**

[15] Böhm J, Pateraki M. From point samples to surfaces, on meshing and alternatives. ISPRS Image Engineering and Vision Metrology. 2006;**XXXVI**:50-55

[16] Golovinskiy A, Funkhouser T. Min-cut based segmentation of point clouds. In: IEEE Workshop on Search in 3D and Video (S3DV) at ICCV; 2009. p. 6

[17] Shapovalov R, Velizhev A. Cutting-plane training of non-associative Markov network for 3D point cloud segmentation. In: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT); 2011. p. 8

[18] Biosca M, Luis Lerma J. Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. ISPRS Journal of Photogrammetry and Remote Sensing. 2008;**63**(1):84-98

[19] Kitamura K, D'Apuzzo N, Kochi N, Kaneko S. Automated extraction of break lines in tls data of real environment. International Archives of Photogrammetry and Remote Sensing. 2010;**38**(5):331-336

[20] Nakagawa M, Kataoka K, Yamamoto T, Shiozaki M, Ohhashi T. Panoramic rendering-based polygon extraction from indoor mobile LiDAR data. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2014;**XL**(4):181-186

[21] Barnes C, Shechtman E, Finkelstein A, Dan Goldman B. PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (Proc. SIGGRAPH). 2009;**28**(3)

[22] Nakagawa M, Tanaka S, Yamamoto T. Colorerd point cloud reconstruction based on image inpainting. In: The 36th Asian Conference on Remote Sensing 2015 (ACRS2015); 2015. p. 7

Chapter 6

`CoClust`: An R Package for Copula-Based Cluster Analysis

Francesca Marta Lilja Di Lascio

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74865

Abstract

The aim of this chapter is to present and describe the R package `CoClust`, which enables implementing a clustering algorithm based on the copula function. The copula-based clustering algorithm, called CoClust, was introduced by Di Lascio and Giannerini in 2012 (Journal of Classification, 29(1):50–75), improved in 2016 (Statistical Papers, p.1–17, DOI 10.1007/s00362-016-0822-3), and is able to find clusters according to the complex multivariate dependence structure of the data-generating process. Hence, among other advantages, the CoClust overcomes the limitations of classic approaches that only deal with linear bivariate relationships. The first part of the chapter briefly describes the clustering algorithm. The second part illustrates the clustering procedure through the R package `CoClust` and presents numerical examples showing how the main R commands can be used to perform a fully developed clustering of multivariate dependent data.

Keywords: clustering algorithm, CoClust, copula function, multivariate dependence structure, R package

1. Introduction

Cluster analysis is an unsupervised classification method that aims to detect a structure within data by assigning a set of objects (observations or variables) into groups, called clusters, whereby objects in the same cluster are in some sense more closely related to each other than objects assigned to different clusters.

Literature on clustering methods is very extensive and different criteria are taken into account to organize and present these methods. One such criterion is the mathematical object of the clustering methods: a distance or dissimilarity measure versus a probability

model. In this chapter, we consider the model-based clustering methods that assume the data matrix is generated according to a specific data generating process (henceforth DGP). The classic model-based clustering method in [1, 2] is based on a mixture of multivariate probability distributions, such as the multivariate normal. However, this approach only accounts for the linear dependence between objects so that it inherits all the limitations of the linear correlation coefficient. Hence, we here focus on clustering methods that assume the data matrix is generated by a K-dimensional copula [3] such that each of the K clusters is represented by a (continuous) univariate density function and the complex multivariate relationship among clusters is expressed by the copula and its dependence parameter. Specifically, this chapter aims to describe the copula-based clustering algorithm first introduced by [4] and improved by [5], presenting in detail its implementation in the `R` package called `CoClust`. The CoClust approach inspired the work of [6], while different copula-based clustering approaches can be found in [7–9], [10–13], [14], and in [15, 16]. To the best of our knowledge, none of these methods have been implemented in software available to the scientific community.

Most clustering algorithms take as input some parameters, such as number of clusters, the distance or density of clusters, or the number of points in a cluster, and a starting classification. Some important benefits of the `R` function `CoClust`, which implements the copula-based clustering algorithm in [5], are that (i) the user can simultaneously test a multiple number of clusters in a single function call, (ii) there is no need for a starting classification, and (iii) the algorithm can nonparametrically estimate the density of the clusters, which are distributional free. The package is available from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/web/packages/CoClust/index.html.

The chapter is organized as follows. Section 2 presents the theoretical tools of copula theory essential to understanding the copula-based clustering algorithm introduced and described in Section 3. Section 4 describes in detail the R implementation of the CoClust algorithm and illustrates its use on simulated DGPs. In Section 5, an application to a real dataset is presented, while a brief conclusion follows in Section 6.

2. Copula theory

The copula function [17–20] was born in the probabilistic metric space with Sklar's theorem [3] stating that every joint distribution function $F(\cdot)$ can be expressed in terms of K marginal distribution function F_k and the copula distribution function C as follows:

$$F(x_1, \ldots, x_k, \ldots, x_K) = C(F_1(x_1), \ldots, F_k(x_k), \ldots, F_K(x_K)) \tag{1}$$

for all $(x_1, \ldots, x_k, \ldots, x_K) \in \overline{\mathrm{R}}^K$ (where $\overline{\mathrm{R}}$ denotes the extended real line). According to this theorem, any joint probability function $f(\cdot)$ can be split into the margins $f_k(\cdot)$ and a copula $c(\cdot)$, so that the latter represents the association among variables, that is, the multivariate dependence structure of a joint density function [17–20]:

$$f(x_1, \ldots, x_k, \ldots, x_K) = c(F_1(x_1), \ldots, F_k(x_k), \ldots, F_K(x_K)) \prod_{k=1}^{K} f_k(x_k). \tag{2}$$

Such separation determines the modeling flexibility of copulas, since it enables (i) freely choosing the distribution of the margins and, separately, that of the copula, (ii) decomposing the estimation problem into two steps: in the first step, the margins are estimated, in the second step, the copula model is estimated, and (iii) combining different estimation methods or approaches.

The log-likelihood function of $f(\cdot)$ is composed of two positive terms as follows:

$$l(\Theta) = \sum_{i=1}^{n} \log c\{F_1(X_{1i}; \beta_1), \ldots, F_k(X_{ki}; \beta_k), \ldots, F_K(X_{Ki}; \beta_K); \theta\} + \sum_{i=1}^{n} \sum_{k=1}^{K} \log f_k(X_{ki}; \beta_k) \tag{3}$$

where the first term involves the copula density $c(\cdot)$ and its parameter θ, and the second involves marginal densities $f_k(\cdot)$ and their parameters β_k, and the whole set of parameters to be estimated is $\Theta = (\beta_1, \ldots, \beta_k, \ldots, \beta_K, \theta)$. Thus, it is possible to estimate $f(\cdot)$ by exploiting the decomposition into two terms of Eq. (2) [20, Chapter 4] using a sequential two-step maximum likelihood method, called inference for margins (henceforth IFM) [21]. This method estimates the marginal parameters in the first step and uses them to estimate the parameter of the copula function in the second step, in either a full or semi-parametric approach.

A full parametric approach for the IFM method is based on the estimation of the marginal parameters $(\beta_1, \ldots, \beta_k, \ldots, \beta_K)$ in the first step by the maximum likelihood estimation for each margin:

$$\widehat{\beta}_k = \arg\max_{\beta_k} \sum_{i=1}^{n} \log f_k(X_{ki}; \beta_k) \tag{4}$$

where each marginal distribution f_k has its own parameters β_k. In the second step, the dependence parameter θ given $\widehat{\beta}_k$ for $k = 1, \ldots, K$ is estimated by:

$$\widehat{\theta} = \arg \max_{\theta} \sum_{i=1}^{n} \log c\left[F_1\left(X_{1i}; \widehat{\beta}_1\right), \ldots, F_k\left(X_{ki}; \widehat{\beta}_k\right), \ldots, F_K\left(X_{Ki}; \widehat{\beta}_K\right); \theta\right]. \tag{5}$$

using the maximum likelihood estimation method.

The IFM method can also be used in a semi-parametric approach [22] where the margins are modeled without assumptions on their parametric form, that is, through the following empirical cumulative distribution function $\widehat{F}_k(X_{ki})$:

$$\widehat{U}_{ki} = \frac{n\widehat{F}_k(X_{ki})}{(n+1)} \tag{6}$$

where $\widehat{F}_k(X_{ki})$ is computed from $(X_{k1}, \ldots, X_{ki}, \ldots, X_{kn})$ with $k = 1, 2, \ldots, K$ and n is the sample size, while the copula parameter θ is estimated by using the following maximum log-likelihood function:

$$\widehat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log c\left[\widehat{U}_{1i}, \ldots, \widehat{U}_{ki}, \ldots, \widehat{U}_{Ki}; \theta\right]. \tag{7}$$

Note that the scaling factor $n/(n+1)$ in Eq. (6) is typically introduced in the nonparametric computation of the margins to avoid numerical problems at the boundary of $[0,1]^K$.

2.1. Copula models

While many different copula models are available in literature (see [18, 19] for details), the Elliptical and Archimedean families are shown to be the most useful in empirical modeling. The Elliptical family includes the Gaussian copula and the *t*-copula: both are symmetric, exhibit the strongest dependence in the middle of the distribution, and can take into account both positive and negative dependence, since $-1 \le \theta \le 1$. The Archimedean family enables describing both left and right asymmetry as well as weak symmetry among the margins using the Clayton, Gumbel, and Frank models, respectively. Clayton's copula has the parameter $\theta \in (0, \infty)$ and as θ approaches zero, the margins become independent. The dependence parameter θ of a Gumbel model is restricted to the interval $[1, +\infty)$ where the value 1 means independence. Finally, the dependence parameter θ of a Frank copula may assume any real value and as θ approaches zero, the marginal distributions become independent. According to the type of copula model, the value of θ has a specific meaning and the magnitudes of the dependence parameter are not comparable across copulas. It is always true that the greater the value of the dependence parameter, the stronger the association among the margins, but since the relationship between θ and the concordance measures is well known, it is standard to convert θ to these, for example, to the Kendall's τ correlation coefficient. The families of copula models considered here are described in **Table 1** and shown in **Figure 1** in their bivariate version. Note that here only single parameter copula models are considered.

The copula model selection task is still an open research field. Although various statistical tests enable evaluating whether a specific model is plausible or not, no tool has thus far been recognized as the best. In the copula-based clustering context, this issue can be overcome,

Copula	$C(u_1, u_2; \theta)$	**Parameter range**	**Kendall's** τ
Gaussian	$\Phi_G\left[\Phi^{-1}(u_1), \Phi^{-1}(u_2)\right]$	$\theta \in (-1, 1)$	$\frac{2}{\pi}\arcsin(\theta)$
Student-*t*	$t_{2,\nu}\left(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2); \theta\right)$	$\theta \in [-1, 1]$, $\nu \in (2, \infty)$	$\frac{2}{\pi}\arcsin(\theta)$
Clayton	$\left[u_1^{-\theta} + u_2^{-\theta} - 1\right]^{-\frac{1}{\theta}}$	$\theta \in (0, \infty)$	$\frac{\theta}{\theta+2}$
Frank	$-\frac{1}{\theta}\ln\left\{1 + \frac{(e^{-\theta u_1}-1)(e^{-\theta u_2}-1)}{(e^{-\theta}-1)}\right\}$	$\theta \in (-\infty, \infty)$	$1 - \frac{4}{\theta}[1 - D_1(\theta)]$
Gumbel	$e^{\left[-(-\log u_1 - \log u_2)^{1/\theta}\right]}$	$\theta \in [1, \infty)$	$1 - \frac{1}{\theta}$

Table 1. Some standard single parameter bivariate copulas with the range of the dependence parameter θ and its relation with Kendall's τ. u_k with $k = 1, 2$ are uniformly distributed variates so that $x_k = F^{-1}(u_k) \sim F_k$. Φ is the cumulative distribution function (cdf) of the standard normal distribution, $\Phi_G(u_1, u_2)$ is the standard bivariate normal distribution, $t_{2,\nu}(\cdot, \cdot; \theta)$ denotes the standard bivariate student-*t* distribution with ν degrees of freedom, and t_ν^{-1} the inverse univariate student-*t* distribution function. $D_1(x)$ denotes the "Debye" function $1/x \int_0^x t/(\exp^t - 1)dt$.

http://dx.doi.org/10.5772/intechopen.74865

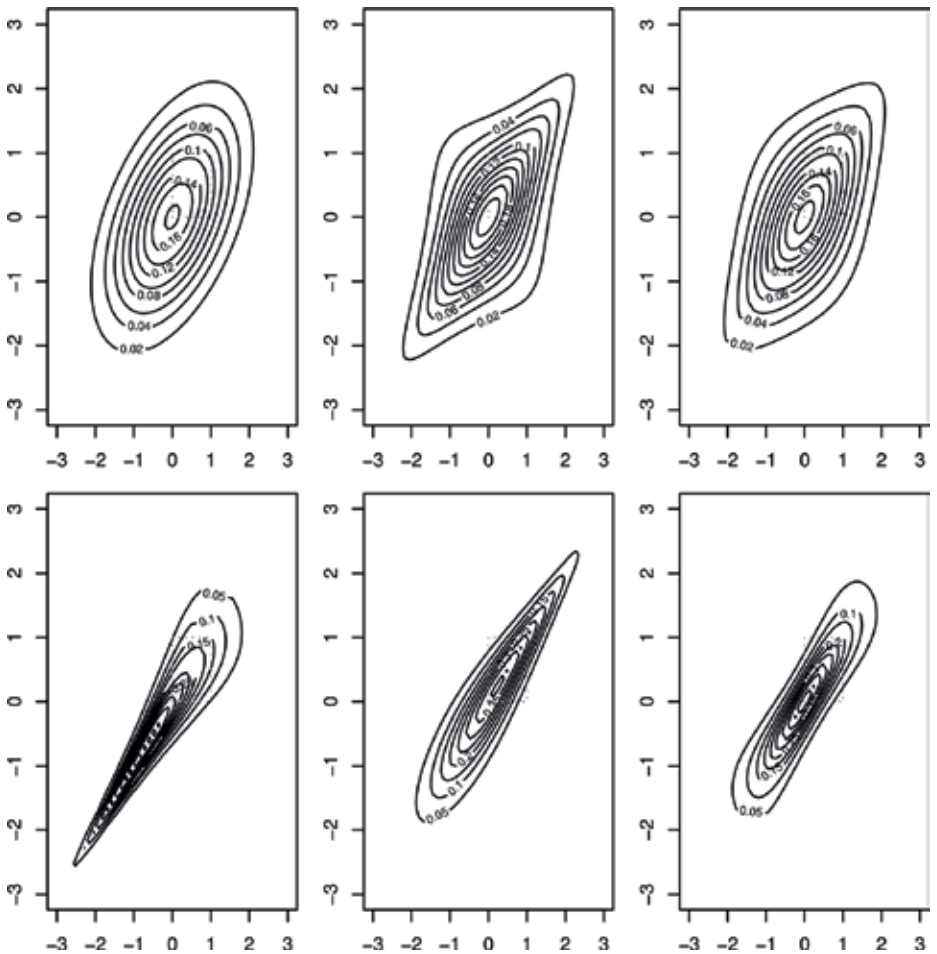

Figure 1. Contour plots of bivariate copula models with normal standard margins and dependence parameter θ such that the Kendall's correlation coefficient is $\tau = 0.7$; upper panel: Gaussian and t-Student copula models for 2 and 4 degrees of freedom; lower panel: Clayton, Gumbel, and Frank copula models.

since the choice of the type of model would seem less important in terms of the goodness of the final clustering (see [5], Section 4.3), and a classic information criterion can be used, such as the Bayesian or the Akaike information criterion. This topic is discussed in detail in Section 3.

3. CoClust algorithm

Di Lascio and Giannerini [4] proposed a clustering algorithm called CoClust that is able to cluster multivariate observations with a complex dependence structure. The basic underlying concept of CoClust is clustering multivariate dependent observations based on the likelihood copula fit estimated on the previously allocated observations. To do so, the CoClust assumes

that the data are generated by a multivariate copula function whose arguments represent the clusters, and each cluster is thus generated by a (marginal) univariate density function. The type and strength of multivariate dependence across clusters are modeled through a copula function and its dependence parameter, respectively. Being copula-based, CoClust inherits all the advantages of copula theory, and the multivariate complex dependence structure of the DGP can be taken into account to perform the cluster analysis. However, CoClust in its first version had some significant limitations. For example, it automatically allocated all the observations to the clusters without discarding potentially irrelevant observations, implying a high computational burden. Di Lascio and Giannerini [5] proposed a new version of the CoClust algorithm that satisfactorily overcomes these limitations. This section hereafter describes the latest version of the CoClust algorithm, which is implemented in the `R` package `CoClust`.

The starting point of the CoClust algorithm is the standard $(n \times p)$ data matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i'1} & \dots & x_{i'j} & \dots & x_{i'p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{np} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_{i'} \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \tag{8}$$

in which np-dimensional objects have to be grouped in K groups. CoClust can be applied either to the row or to the column data matrix according to the purpose of the analysis. In both cases, CoClust works with vectors and treats each row (column) of the data matrix $\mathbf{X}$ as a single element to be allocated to a cluster. The values within a row (or column) vector are treated as independent realizations of the same density function, thus observations for each cluster from the same distribution. Here, CoClust is described as applied to the rows of the data matrix.

3.1. The basic procedure and selection of the number of clusters

The basic idea behind the CoClust consists in a forward procedure that allocates a K-plet of the row data matrix at a time, that is, a p-dimensional vector for each cluster at a time, and the decision on the allocation of each K-plet of rows is based on the value of the log-likelihood of the copula fit. This likelihood is computed by using the K-plets already allocated and one allocation candidate, say $\mathbf{x}_{i'} = \left(x_{i'1}, \dots, x_{i'j}, \dots, x_{i'p}\right)$, by varying the permutations of observations in $\mathbf{x}_{i'}$ in order to find, if it exists, the combination that maximizes the copula fit. If the log-likelihood of the copula fitted on the observations already allocated plus the permutation of the selected K-plet increases, then the candidate K-plet is allocated to the clusters; otherwise, it is discarded, since theoretically it could be either independent from the identified DGP or derive from another DGP.

Before describing the clustering algorithm procedure, two aspects of the CoClust merit a discussion: the construction of the K-plet candidate to the allocation and the selection of number of clusters K. The K-plet of rows candidate to the allocation is constructed based on the following function $H(\cdot)$, which is a sort of multivariate measure of association based on the pairwise Spearman's ρ correlation coefficient:

$$H(\Lambda_2|\Lambda_1) = \max_{i' \in \Lambda_2} \left\{ \underset{i \in \Lambda_1}{\psi} \left(\rho(\mathbf{x}_i, \mathbf{x}_{i'}) \right) \right\} \tag{9}$$

where Λ_1 is the subset of row index vectors already selected to compose a K-plet, Λ_2 is the subset of the remaining candidate row index vectors to complete it, and ψ is an aggregation function, for instance, the mean, median, or maximum.

As for the selection of K, one of the advantages of CoClust with respect to classic clustering techniques is its ability to automatically choose the number of clusters. Indeed, CoClust explores all the possibilities among those given by the user and selects the K on the basis of the log-likelihood of the copula estimated on the subsets of k-plets allocated up to the user's predefined step. The technical details are given below.

The main steps of the CoClust algorithm to cluster the n row data matrix are described in the following:

1. by varying the number of clusters k in the set of possibilities defined by the user and such that $2 \leq k \leq n$,
   a. select a subset of n_k k-plets of rows in the data matrix in Eq. (8) on the basis of the measure in Eq. (9);
   b. fit the copula model on the n_k k-plets of rows through the semiparametric estimation method described in Section 2;
2. select the subset of n_k k-plets of rows, say n_K K-plets, that maximizes the log-likelihood of the copula; hence, the number of clusters K, that is, the dimension of the copula, is automatically chosen and n_K K-plets are already allocated;
3. select a K-plet of rows among those remaining by using the measure in Eq. (9) and estimate $K!$ copulas by using the observations already clustered and a permutation of the candidate to the allocation;
4. allocate the permutation of the selected K-plet to the clustering by assigning each row to the corresponding cluster only if it increases the log-likelihood of the copula fit, otherwise drop the entire K-plet of rows;
5. repeat steps 3 and 4 until all the observations are evaluated, that is, either allocated or discarded.

At the end of the procedure, we obtain a clustering of K clusters each containing a maximum $(n/K)p$ independent observations such that the multivariate dependence relationship across clusters can be revealed. Hence, attention in recovering the multivariate relationships does not

rely on the within-cluster relationships, typical of classic clustering methods. A picture of the final CoClust clustering is given in **Figure 2**. Each cluster is a set of independent and identically distributed realizations from the same marginal distribution while observations across clusters share the same multivariate dependence structure.

Note that since in each step of the procedure non-nested models are compared, that is, copula models with a single dependence parameter, the described log-likelihood based criterion is equivalent to the well-known Bayes information criterion and Akaike information criterion. Finally, note that in the current CoClust version, the selection of the number of clusters K is based on a representative subset of n_k observations. Hence, the algorithm chooses the number of clusters K by estimating the $\sum_{k=K_{\min}}^{K_{max}} \binom{n_k}{k}$ fits required, where $[K_{\min}, K_{max}]$ is the range of the number of clusters predefined by the user with $K_{\min} \geq 2$ and n_k is chosen by the user with $n_k \ll p$. This allows keeping the computational complexity under control since it does not depend on sample size.

3.2. Selecting the copula model

The CoClust algorithm has not been implemented to automatically perform the selection copula model task and requires employing an information criterion *a posteriori*. The Bayesian information criterion (henceforth BIC) is expressed as follows for a K-dimensional copula model m:

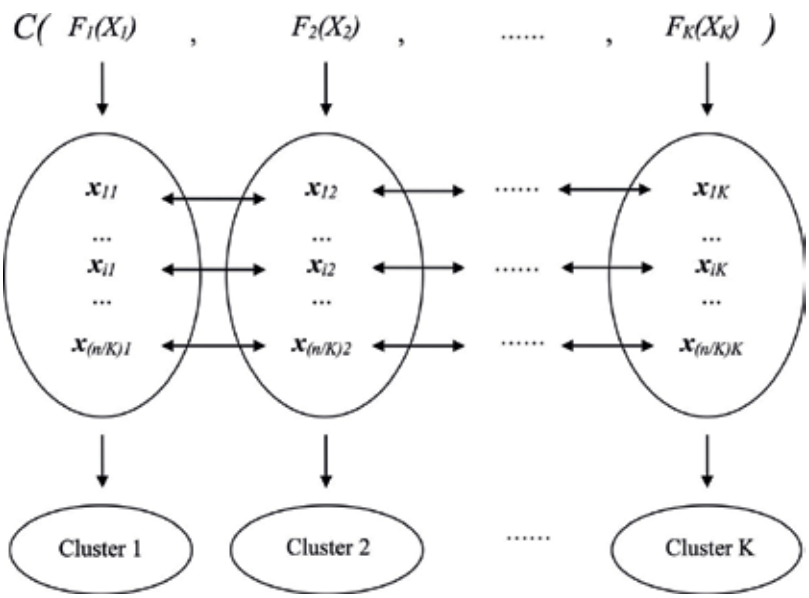

Figure 2. The basic concept underlying the CoClust algorithm. Each element in a cluster is a row data matrix of p elements.

$$BIC_{K,m} = -2\log\Pi_{i=1}^{n} c_m\left\{\widehat{F}_1(X_{1i}),\ldots,\widehat{F}_k(X_{ki}),\ldots,\widehat{F}_K(X_{Ki});\widehat{\theta}\right\} + s\log\left((n/K)p\right) \tag{10}$$

where $\widehat{\theta}$ is as in Eq. (5) or Eq. (7) with the summation over the number of allocated observations, which equals maximum $(n/K)p$ (i.e., n/K p-dimensional vectors) and s is the number of parameters. According to [23], we select the copula model that minimizes the BIC. Similarly, the Akaike information criterion (henceforth AIC) results in:

$$AIC_{K,m} = -2\log\Pi_{i=1}^{n} c_m\left\{\widehat{F}_1(X_{1i}),\ldots,\widehat{F}_k(X_{ki}),\ldots,\widehat{F}_K(X_{Ki});\widehat{\theta}\right\} + 2s \tag{11}$$

and can also be used to select the copula model.

3.3. Assessing the CoClust performance

The goodness of the CoClust algorithm in finding the true multivariate clustering structure underlying the data has been extensively investigated. Specifically, the first version of CoClust [4] was tested on simulated data for different scenarios and compared with model-based clustering [1, 2]. This shows that, both when the DGP is a copula and when it is misspecified, CoClust appears to be able to identify both the true number of clusters and their size in most situations. Moreover, in comparing model-based clustering, CoClust appears better suited to clustering dependent data. In [5], a more sophisticated Monte Carlo study was carried out, investigating the new features of the current version of the CoClust algorithm. Here, the current version of the algorithm clearly outperforms the previous version by [4] and CoClust's ability to find the correct number of clusters and to reconstruct the true k-plets by varying the dimension of the copula, the aggregation function ψ, and the copula model appears to be very satisfactory. Furthermore, [5] also obtained good results in assessing CoClust's ability to drop from the clustering observations that are independent of the true DGP as well as distinguishing two different DGPs in the same dataset.

As for real data applications, the CoClust algorithm has been successfully applied to several datasets. In relation to biomedical applications, [4] apply the CoClust to microarray data to formulate hypotheses on the possible co-regulation and functional relations between genes, [5] use the copula-based clustering method to identify biologically and clinically relevant groups of tumor samples, and [24] attempt to identify organ type from cancer cell lines from tumors. Applications in other fields include [25], where the purpose of the analysis is investigating changes in EU country diets in accordance with common European policies and guidelines on healthy diets, and [24], where CoClust is used to investigate the geographic distribution of (annual maxima) rainfall measurements.

4. The R implementation of the CoClust algorithm

The copula-based clustering algorithm procedure is implemented in the `R` package `CoClust` [26]. It must be installed in the usual way, that is:

```
R> install.packages("CoClust")
```

and then it must be loaded through the usual code:

```
R> library("CoClust")
```

The code of the `CoClust` package is entirely written in `R`, to enable using an easily accessible open source system and the input/output facilities.

4.1. List of functions and subroutines

The main R function is `CoClust()`, which performs the copula-based clustering, while the following auxiliary R functions.

`fit.margin()`, `fit.margin2()`, `fit.margin3()`, `fcond.mod()`, `CoClust_perm()`, `stima_cop()` are intended for internal use only and are not documented in the package.

4.2. The CoClust function

The main function of the package `CoClust` is the R function `CoClust()`, which performs copula-based clustering as described in Section 3. Some options are present, which mainly allow us to:

- fit a variety of copula models (by setting the argument `copula`) with different types of estimation procedures for margins and for copulas (arguments `method.ma` and `method.c`, respectively); specifically, all the copula models belonging to the Elliptical and the Archimedean family described in Section 2 can be estimated through the estimation methods implemented in the `R` package `copula` [27–30] that are maximum pseudo-likelihood estimators based on two different variance estimators, the inversion of Kendall's τ estimator and the inversion of Spearman's ρ estimator; as for the margins, two different estimation methods have been implemented, one parametric and one nonparametric: the maximum likelihood method as in Eq. (4) and the empirical cumulative distribution function in Eq. (6);
- set the range or set of dimensions for the copula model, that is, number of clusters, for which the function tries the clustering (argument `dimset`);
- set the dimension of the sample units used for selecting the number of clusters (argument `noc`);
- select the combination function of the pairwise Spearman's ρ used to select the k-plets among the mean, the median, or the maximum (argument `fun`) as defined in Eq. (9);
- specifies the likelihood criterion used for selecting the number of clusters among the AIC, the BIC (as defined in Eqs. (10) and (11)), and the log-likelihood without penalty terms (argument `penalty`).

The argument `copula` allows specifying a copula model among those described in Section 2.1. As for the selection of the "best" model, CoClust can be run by varying the type of models of

interest and selecting the one that fits best *a posteriori* using one of the criteria introduced in Section 3.2.

The typical use of the function `CoClust` is as follows:

```
CoClust(m, dimset = 2:5, noc = 4, copula = "frank", fun = median,
        method.ma = c("empirical", "pseudo"), method.c = c("ml", "mpl",
        "irho", "itau"),
        dfree = NULL, writeout = 5, penalty = c("BICk", "AICk", "LL"), ...)
```

where `m` is the entry data matrix and the `writeout` argument allows monitoring the allocation process, since it informs on each new allocated observation. Further details on the input arguments are given in the package help files.

The main output of the function `CoClust` is an object of S4 class "CoClust" which is a list with the following elements:

1. Number of Clusters: the number K of selected and identified clusters;
2. Index Matrix: a $n.obs \times (K+1)$ matrix where $n.obs$ is the number of observations put into each cluster; the matrix contains the row indexes of the observations of the data matrix `m` (Eq. (8)) and in the last column the log-likelihood of the copula fit;
3. Data Clusters: the data matrix of the final clustering; each column contains the observations allocated in a cluster;
4. Dependence: a list containing:
   a. `Model`: the copula model used for the clustering;
   b. `Param`: the estimated dependence parameter between/among clusters;
   c. `Std.Err`: the standard error of `Param`;
   d. `P.val`: the p-value associated to the null hypothesis $H_0 : \theta = 0$;
5. LogLik: the maximized log-likelihood copula fit;
6. Est.Method: the estimation method used for the copula fit;
7. Opt.Method: the optimization method used for the copula fit;
8. LLC: the value of the log-likelihood criterion for each k in `dimset`;
9. Index.dimset: a list that, for each k in `dimset`, contains the index matrix of the initial set of n_k observations used to select the number of clusters, together with the associated maximized log-likelihood copula fit.

4.3. Simulated examples

This section shows how to use the `CoClust` package on data simulated from different DGPs. In the first example, the data are drawn from a joint density function with different margins,

whereas in the second example, a misspecified DGP is used. In these examples, we focus only on the semi-parametric approach described in Section 2 due to its theoretical and computational advantages with respect to the full parametric approach. Moreover, the latter has only been implemented for Gaussian margins.

Example 1

In this example, we build a 3-variate joint density function through a 3-dimensional Frank copula with dependence parameter such that the Kendall's $\tau = 0.7$ and three different margins: a Gaussian with parameters $\mu = 7, \sigma = 2$, a Gamma with shape and rate, respectively, set to 3 and 4, and a Beta with parameters $\alpha = 2, \beta = 1$. To do so, we employ the function `mvdc` of the `copula` package [27–30]. Next, we generate a data matrix X with 15 rows and 21 columns and build the matrix of the true cluster indexes. Finally, we apply the function `CoClust` to the rows of X, recover the multivariate dependence structure of the data and compare the obtained clustering with the true one.

Code to generate the example dataset is given in the following. We first define the DGP:

```
R> n.marg <- 3
R> theta <- iTau(frankCopula(), 0.7)
R> copula <- frankCopula(theta, dim = n.marg)
R> mymvdc <- mvdc(copula, c("norm", "gamma", "beta"),list(list(mean=7, sd=2),
+              list(shape=3, rate=4), list(shape1=2, shape2=1)))
```

and then generate the data and save the true clustering in index.true:

```
R> set.seed(11)
R> n.col  <- 21
R> n.row  <- 15
R> n      <- n.row*n.col/n.marg
R> x.samp <- rMvdc(n, mymvdc)
R> X      <- matrix(x.samp, nrow = n.row, ncol = n.col, byrow=TRUE)
R> index.true <-  matrix(1:n.row, n.row/n.marg, n.marg)
R> colnames(index.true) <- c("Cluster 1", "Cluster 2", "Cluster 3")
R> index.true
```

Note that n is the number of observations for each margin.

We apply the `CoClust` to the 15×21 data matrix X using the maximum likelihood estimation method for the copula, the empirical cumulative distribution function for the three margins, and leaving by default the remaining arguments:

```
R> clust <- CoClust(X, dimset = 2:4, noc=2, copula="frank", method.
ma="empirical",
+         method.c="ml", writeout=1)
```

The output is as follows:

```
R> clust
```

```
An object of class "CoClust"
Slot "Number.of.Clusters":
[1] 3

Slot "Index.Matrix":
     Cluster 1 Cluster 2 Cluster 3 LogLik
[1,]       11         1         6 34.15693
[2,]       13         3         8 69.87149
[3,]       12         2         7 103.67653
[4,]       14         4         9 136.31506
[5,]       15         5        10 170.36557

Slot "Data.Clusters":
       Cluster 1 Cluster 2  Cluster 3
   [1,] 0.35776965  4.566417  0.1634203
   [2,] 0.36621352  5.532188  0.1470511
   [3,] 0.99290268 11.191092  1.4006169
   [4,] 0.60411081  6.613533  0.5457595
   [5,] 0.13946354  2.658381  0.3489739
   [6,] 0.80523424  9.526025  0.6908222
   [7,] 0.79477600  8.899494  0.7864765
   [..]    .......   .......    .......
[102,] 0.36904520 3.629722 0.2763105
[103,] 0.82647042 10.809628 1.3899675
[104,] 0.48283666 5.185873 0.2763133
[105,] 0.90394435 10.583053 0.9962130

Slot "Dependence":
$Copula
[1] "frank"

$Param
[1] 11.95576

$Std.Err
[1] 0.8261832

$P.value
[1] 0


Slot "LogLik":
[1] 170.3656
```

```
Slot "Est.Method":
[1] "maximum likelihood"

Slot "Opt.Method":
[1] "ml"

Slot "LLC":
         2           3         4
 -70.93179  -136.00532 -41.35904

Slot "Index.dimset":
$`2`
     1 2   LogLik
[1,] 11 1 19.75591
[2,]  8 3 37.33473

$`3`
     1 2 3   LogLik
[1,] 11 1 6 34.15693
[2,] 13 3 8 69.87149

$`4`
     1 2  3  4     LogLik
[1,] 11 1  6 12  3.653809
[2,]  7 3 13  8 22.548352
```

To look at specific objects of the resulting list, it is possible to select, among others, the following information:

```
R> clust@"Number.of.Clusters"   # Selected number of clusters
R> clust@"Dependence"$Param     # Estimated copula parameter
R> clust@"Data.Clusters"        # Clustered data
```

To compare the obtained clustering with the true clustering we can input:

```
R> index.clust <- clust@"Index.Matrix"
R> index.clust
R> index.true
```

to obtain as follows:

```
> index.clust
     Cluster 1 Cluster 2 Cluster 3   LogLik
[1,]        11         1         6  34.15693
[2,]        13         3         8  69.87149
[3,]        12         2         7 103.67653
```

```
[4,]        14         4          9 136.31506
[5,]        15         5         10 170.36557
> index.true
     Cluster 1 Cluster 2 Cluster 3
[1,]         1         6        11
[2,]         2         7        12
[3,]         3         8        13
[4,]         4         9        14
[5,]         5        10        15
```

The obtained clustering is perfect, CoClust is able to recognize the exact structure underlying the data. Note that the label of each cluster, that is, the order of the margins, is not relevant. The only important aspect is the composition of each cluster, that is, the row indexes in each column of `index.clust` and their order that has to be such that it reconstructs the exact 3-plets across the columns.

To apply the `CoClust` to the 15×21 data matrix X previously generated by changing the argument `fun` in `max`, or the range of number of clusters to be tried or the copula model, we can input respectively:

```
R> clust <- CoClust(X, dimset = 2:4, noc=2, copula="frank", fun=max,
+                   method.ma="empirical", method.c="ml", writeout=1)

R> clust <- CoClust(X, dimset = 3:5, noc=2, copula="frank",
+                   method.ma="empirical", method.c="ml", writeout=1)

R> clust <- CoClust(X, dimset = 2:4, noc=2, copula="clayton",
+                   method.ma="empirical", method.c="ml", writeout=1)
```

Example 2

In this example, we use a different DGP from the copula, thus showing the use of CoClust in the misspecification case. Specifically, a 30×21 data matrix is drawn from a three-dimensional skew-normal distribution through the R package `sn` [31], we then apply CoClust to cluster the row data matrix:

```
R> library(sn)
R> n.marg <- 3
R> rho    <- 0.7
R> mu     <- c(4,6,7)
R> v1     <- 1
R> v2     <- 1
R> v3     <- 1
R> omega <- matrix(c(v1, rho*sqrt(v1*v2), rho*sqrt(v1*v3), rho*sqrt(v1*v2), v2,
+        rho*sqrt(v3*v2), rho*sqrt(v1*v3), rho*sqrt(v3*v2), v3), n.marg)
R> alpha  <- c(-1,1,1)
R> n.col  <- 21
```

```
R> n.row  <- 60
R> n.k    <- n.row/n.marg
R> n      <- n.row*n.col/n.marg
R> set.seed(11)
R> x.samp <- rmsn(n, xi=mu, Omega=omega, alpha=alpha)
R> X      <- matrix(x.samp, nrow=n.row, ncol=n.col, byrow=TRUE)
R> clust  <- CoClust(X, dimset=2:5, noc=4, copula="clayton",method.
ma="empirical",
+                    method.c="ml", penalty = "BICk", writeout=1)
```

On the console, it is possible to monitor the number of observations already allocated (argument writeout). Indeed, while CoClust runs, the following information appears on the console:

```
Number of clusters selected: 3
Allocated observations: 5
Allocated observations: 10
Allocated observations: 15
```

To look at the obtained clustering and its details, one has to input:

```
R> clust
R> index.clust <- clust@"Index.Matrix"
R> index.true  <- matrix(1:n.row, n.row/n.marg, n.marg)
R> index.clust; index.true
```

Note that when the number of *K*-plets to be allocated is not small, the goodness of the obtained clustering is difficult to determine. Hence, for example, two functions can be exploited to assess the quality of the final clustering: `pca.coclust`, which counts how many *K*-plets of the true DGP have been correctly allocated in the final clustering, and `pcc.coclust`, which counts how many *K*-plets of the obtained clustering have been correctly allocated. In Appendix A., the R code of these two functions is shown. Here, we compute the true clustered index matrix as follows:

```
R> ind.t <- apply(matrix(1:n.row,n.k), FUN=paste, MARGIN=1, collapse="-")
```

and the two functions `pca.coclust` and `pcc.coclust` after loading the required package `gtools`:

```
R> library(gtools)

R> pca.coclust(clust, ind.t, n.marg)
[1] 65

R> pcc.coclust(clust, ind.t, n.marg)
[1] 86.66667
```

The obtained values inform us that 65% of 3-plets deriving from the true DGP are correctly allocated and 86.7% of 3-plets in the final clustering are correctly allocated.

http://dx.doi.org/10.5772/intechopen.74865

5. Application to wine dataset

In this section, an application of the CoClust package to a real dataset is shown. [32] analyze a set of Italian wines by observing the chemical properties of 178 specimens of three types of wines (Barolo, Grignolino, and Barbera) produced in the Piedmont region in Italy. The data are available in the package sn under the name wines.

A subset of randomly selected wines has been analyzed through CoClust by varying the number of clusters from 2 to 7 and the copula model among the three models of the Archimedean family. Since Grignolino is a type of wine with characteristics between those of Barolo and Barbera, we work with a sample of only these two last types of wines. Code is as follows:

```
R> data(wines)
R> n <- 6
R> set.seed(11)
R> ind.sample<- c(sample(1:59,n,replace=FALSE),sample(131:178,n,replace=FALSE))
R> X <- wines[ind.sample,-1]

R> clustF <- CoClust(X, dimset = 2:7, noc=1, copula="frank", method.
ma="empirical",
+             method.c="ml",writeout=1)

R> clustC <- CoClust(X, dimset = 2:7, noc=1, copula="clayton", method.
ma="empirical",
+             method.c="ml",writeout=1)

R> clustG <- CoClust(X, dimset = 2:7, noc=1, copula="gumbel", method.
ma="empirical",

+             method.c = "ml",writeout = 1)
```

To evaluate the final clustering obtained with a specific copula model, say the Frank model, and to compare it with the true classification of the 12 selected wines, the code is as follows:

```
R> Type.wine <- wines[ind.sample,1]
R> Type.wine

R > K < — clustF@"Number.of.Clusters".

R > index.clust <— clustF@"Index.Matrix".

R> index.clust
R> index.fin <- matrix(Type.wine[index.clust[,1:K]],nrow=nrow(index.clust),
 +                   ncol=(ncol(index.clust)-1))
R> index.fin
     [,1]       [,2]        [,3]       [,4]       [,5]       [,6]
```

```
[1,] "Barolo"  "Barolo"  "Barolo"  "Barolo"  "Barolo"  "Barolo"
[2,] "Barbera" "Barbera" "Barbera" "Barbera" "Barbera" "Barbera"
```

CoClust selects 6 clusters and allocates to each cluster the two types of wines. Thus, across clusters, we can perfectly recognize the two types of Italian wines and in each cluster we have different wines with different (e.g., independent) chemical characteristics.

Similarly, the other two copula models can be used as in `clustC` and `clustG` above. The results appear to not be affected by the type of model used even though, based on the log-likelihood of the copula fitted on the final clustering, the more appropriate model appears to be the Gumbel model with a log-likelihood equal to 527.3022 (compared to 500.8835 for the Frank copula and 429.184 for the Clayton copula).

6. Conclusion

In this chapter, we describe a copula-based clustering algorithm and its implementation in the `R` package `CoClust`. One major advantage of this new package is that it provides an algorithm that is able to cluster multivariate observations by taking into account their underlying complex multivariate dependence structure. Being copula-based, the CoClust algorithm inherits the benefits of the copula. Thus, potentially any type of multivariate dependence structure can be handled and the most appropriate method can be employed to estimate both a probability model for each cluster/margin and the copula model.

The current version of the R package implements the clustering algorithm procedure in the main function `CoClust`. It enables the user to simultaneously choose the copula model, the estimation method for the margins and for the copula, the aggregation function for constructing the k-plet of observation allocation candidates. Moreover, the range (or set) of the number of clusters from among which the procedure automatically selects the best one and the sample size to be used to select it can be varied.

As with many other software packages, `CoClust` package is continually being augmented and improved. We are currently investigating possible graphical solutions for the final clustering and implementing some measures to validate the clustering solution. Another future direction includes expanding the functionality of the `CoClust` package to allow comparing the solution of other clustering algorithms, such as mixture-based clustering and hierarchical clustering methods.

Acknowledgements

The author acknowledges the support of the Free University of Bozen-Bolzano, Faculty of Economics and Management, via the project “Aggregation functions for Innovation and Data

http://dx.doi.org/10.5772/intechopen.74865

Analysis (AIDA)" and Professor Simone Giannerini, University of Bologna, with whom the first version of the package was developed.

A. Appendix

The following two functions are useful to evaluate the goodness of the final clustering obtained through the CoClust algorithm when true clustering or benchmark clustering is available. The arguments of these two functions are `ccfit`, which is the object `CoClust` as given by the corresponding R function; `ind.t`, which is the true clustering expressed through the clustered index matrix with clusters by columns and the row index of matrix in Eq. (8) by rows; and `nmarg`, which is the dimension of the copula model, that is, the selected number of clusters. For an example of the use of these two functions see Section 4.3, "Example 2".

```
R> library("gtools")

pca.coclust <- function(ccfit, ind.t, nmarg){
    n.marg <- ccfit@"Number.of.Clusters"
    ind.perm <- permutations(n.marg,n.marg)
    n.comb <- nrow(ind.perm)
    if(n.marg==nmarg){
        ind.cc <- ccfit@"Index.Matrix"[,1:n.marg]
        n.kp <- nrow(ind.cc)
        res <- rep(NA,n.kp)
        for(i in 1:n.kp){
            dum <- ind.cc[i,]
            res0 <- rep(NA,n.comb)
            for(j in 1:n.comb){
                ind.ccs <- dum[ind.perm[j,]]
                ind.ccs <- paste(ind.ccs, collapse="-")
                res0[j] <- as.integer(ind.ccs%in%ind. t)
            }
            res[i] <- any(res0)
        }
        pca.k <- sum(res)/length(ind.t)*100
    }
    return(pca.k=pca.k)
}
pcc.coclust <- function(ccfit, ind.t, nmarg){
    n.marg <- ccfit@"Number.of.Clusters"
    ind.perm <- permutations(n.marg,n.marg)
    n.comb <- nrow(ind.perm)
    if(n.marg==nmarg){
       ind.cc <- ccfit@"Index.Matrix"[,1:n.marg]
```

```
        n.kp <- nrow(ind.cc)
        res <- rep(NA,n.kp)
        for(i in 1:n.kp){
            dum <- ind.cc[i,]
            res0 <- rep(NA,n.comb)
            for(j in 1:n.comb) {
                ind.ccs <- dum[ind.perm[j,]]
                ind.ccs <- paste(ind.ccs, collapse="-")
                res0[j] <- sum(ind.ccs%in%ind.it)
            }
            res[i] <- any(res0)
        }
        pcc.k    <- sum(res)/nrow(ccfit@"Index.Matrix")*100
    }
    return(pcc.k=pcc.k)
}
```

Author details

Francesca Marta Lilja Di Lascio

Address all correspondence to: marta.dilascio@unibz.it

Faculty of Economics and Management, Free University of Bozen-Bolzano, Italy

References

[1] Fraley C, Raftery AE. How many clusters? Which clustering method? Answers via model–based cluster analysis. The Computer Journal. 1998;**41**(8):578-588

[2] Fraley C, Raftery AE. Model–Based Clustering, Discriminat Analysis and Density Estimation. Technical report. Department of Statistics, University of Washington; 2000

[3] Sklar A. Fonctions de répartition à n dimensions et leures marges. Publications de l'Institut de Statistique de L'Université de Paris. 1959;**8**:229-231

[4] Di Lascio FML, Giannerini S. A copula-based algorithm for discovering patterns of dependent observations. Journal of Classification. 2012;**29**(1):50-75

[5] Di Lascio FML, Giannerini S. Clustering dependent observations with copula functions. Statistical Papers. 2016:1-17

[6] Chessa A, Crimaldi I, Riccaboni M, Trapin L. Cluster analysis of weighted bipartite networks: A new copula-based approach. PLoS One. 2014;**9**(10):e109507

[7] Durante F, Pappadà R, Torelli N. Clustering of financial time series in risky scenarios. Advance in Data Analysis and Classification. 2014;**8**:359-376

[8] Durante F, Pappadà R. Cluster analysis of time series via Kendall distribution. In: Grzegorzewski P, Gagolewski M, Hryniewicz O, Gil MA, editors. Strengthening Links Between Data Analysis and Soft Computing, Volume 3l5 of Advances in Intelligent Systems and Computing. Springer International Publishing; 2015. pp. 209-216

[9] Durante F, Pappadà R, Torelli N. Clustering of time series via non–parametric tail dependence estimation. Statistical Papers. 2015;**56**(3):701-721

[10] De Luca G, Zuccolotto P. A tail dependence-based dissimilarity measure for financial time series clustering. Advances in Data Analysis and Classification. 2011;**5**(4):323-340

[11] De Luca G, Zuccolotto P. Time series clustering on lower tail dependence for portfolio selection. In: Corazza M, Pizzi C, editors. Mathematical and Statistical Methods for Actuarial Sciences and Finance. Berlin: Springer; 2014. pp. 131-140

[12] De Luca G, Zuccolotto P. Dynamic tail dependence clustering of financial time series. Statistical Papers, page in press. 2015

[13] De Luca G, Zuccolotto P. A double clustering algorithm for financial time series based on extreme events. Statistics *and Risk Modeling*. 2017;**34**(1–2):1-12

[14] D'Urso P, Disegna M, Durante F. Copula-based fuzzy clustering of time series. In: Mola F, Conversano C, editors. Book of Abstracts of the 10th Scientific Meeting of the Classification and Data Analysis Group, Page 4. Cagliari: CUEC; 2015

[15] Arakelian V, Karlis D. Clustering dependencies via mixtures of copulas. Communication in Statistics - Simulation and Compution. 2014;**43**(7):1644-1661

[16] Kosmidis I, Karlis D. Model-based clustering using copulas with applications. Statistics and Computing. 2016;**26**(5):1079-1099

[17] Cherubini U, Luciano E, Vecchiato W. Copula Methods in Finance. Chichester: Wiley Finance Series. John Wiley & Sons Ltd.; 2004

[18] Durante F, Sempi C. Principles of Copula Theory. Boca Raton: CRC Press; 2015

[19] Nelsen RB. Introduction to Copulas. New York: Springer; 2006

[20] Trivedi PK, Zimmer DM. Copula Modeling: An Introduction for Practitioners, volume 1. Foundations and Trends in Econometrics. 2005

[21] Joe H, Xu J. The Estimation Method of Inference Functions for Margins for Multivariate Models. Technical report. Department of Statistics, University of British Columbia; 1996

[22] Genest C, Ghoudi K, Rivest LP. A semiparametric estimation procedure of dependence parameters in multivariate families of distributions. Biometrika. 1995;**82**:543-552

[23] Raftery AE, Nema D. Variable selection for model-based clustering. Journal of the American Statistical Association. 2006;**101**(473):168-178

[24] Di Lascio FML, Durante F, Pappadà R. Copulas and Dependence Models with Applications - Contributions in Honor of Roger B. Nelsen, chapter copula–based clustering methods. Springer; 2017. pp. 49-67

[25] Di Lascio FML, Disegna M. A copula-based clustering algorithm to analyse eu country diets. Knowledge-Based Systems. 2017;**132**:72-84

[26] Di Lascio FML, Giannerini S. CoClust: Copula based cluster analysis. R package version 0.3-2; 2017

[27] Hofert M, Ivan Kojadinovic I, Maechler M, copula JY. Multivariate dependence with copulas. R package version 0.999-18; 2017

[28] Hofert M, Maechler M. Nested archimedean copulas meet r: The nacopula package. Journal of Statistical Software. 2011;**39**(9):1-20

[29] Kojadinovic I, Yan J. Modeling multivariate distributions with continuous margins using the copula r package. Journal of Statistical Software. 2010;**34**(9):1-20

[30] Yan J. Enjoy the joy of copulas: With a package copula. Journal of Statistical Software. 2007;**21**(4):1-21

[31] Azzalini A. The R Package `sn`: The Skew-Normal and Skew-T Distributions, version 1.5-0; 2017

[32] Forina M, Armanino C, Castino M, Ubigli M. Multivariate data analysis as a discriminating method of the origin of wines. Vitis. 1986;**25**:189-201

Chapter 7

Temporal Clustering for Behavior Variation and Anomaly Detection from Data Acquired Through IoT in Smart Cities

Vladimir Urosevic, Ana Kovacevic,
Firas Kaddachi and Milan Vukicevic

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75203

Abstract

In this chapter, we propose a methodology for behavior variation and anomaly detection from acquired sensory data, based on temporal clustering models. Data are collected from five prominent European smart cities, and Singapore, that aim to become fully "elderly-friendly," with the development and deployment of ubiquitous systems for assessment and prediction of early risks of elderly Mild Cognitive Impairments (MCI) and frailty, and for supporting generation and delivery of optimal personalized preventive interventions that mitigate those risks, utilizing smart city datasets and IoT infrastructure. Low level data collected from IoT devices are preprocessed as sequences of activities, with temporal and causal variations in sequences classified as normal or anomalous behavior. The goals of proposed methodology are to (1) recognize significant behavioral variation patterns and (2) support early identification of pattern changes. Temporal clustering models are applied in detection and prediction of the following variation types: intra-activity (single activity, single citizen) and inter-activity (multiple-activities, single citizen). Identified behavioral variations and anomalies are further mapped to MCI/frailty onset behavior and risk factors, following the developed geriatric expert model.

Keywords: temporal clustering, IoT, smart cities, behavior recognition, anomaly detection

1. Introduction

Frailty and Mild Cognitive Impairment are common and inevitable conditions in the elderly citizen population defined as premature or accelerated physical and mental declines. These conditions are often an early indicator of more severe states, such as Alzheimer's disease. Control (delaying or decelerating) of the onset and progression of MCI/frailty is becoming one of the major tasks of global efforts in maintaining the functional independence and quality of life of the globally growing elderly population. In 2016, for the first time in history, estimated majority of the world population can expect to live into their sixties and beyond. Beside the global initiatives, strategies and action plans on healthy ageing conducted by the relevant key organizations such as World Health Organization (WHO), or United Nations (UN), the growing market trend for the so-called "silver economy" sector is booming. The increase of population aged 65 and over is projected to reach 28.1% of the whole population in the EU by 2050, they have a spending power higher than the generation segment aged 18 to 39, and they account for approx. 60% of total expenditures in the US and 50% in the UK, generating demand for new services and products, ranging from personalized care to age-friendly technologies and other solutions that enable the maintenance and prolongation of healthy, independent lives. Technologies and systems supporting innovative ways of influencing people's behavior and lifestyles at all ages also present a significant economic and business opportunity.

Geriatric practice has in this aim utilized different standardized instruments based on traditional data collection methods (administration of questionnaires, meter-based measurement or direct observation in controlled conditions) which are in most cases intrusive and demand citizens' presence in geriatric centers and a lot of time for data collection. More importantly, these methods do not enable real time monitoring of behavioral changes (e.g., data from questionnaires are collected on semi-annual or annual intervals) and thus prevent predictive and preventive interventions. Finally, data collected from such method is often subjective or incomplete.

With the goal to overcome the stated drawbacks**, many technological instruments and methods emerged**, aiming to automate as much as possible the detection and mitigation of behavior deteriorations and anomalies. Particularly, the recent development and expansion of wearable technologies/devices and Internet of Things (IoT) has enabled the build-up of infrastructures of smart devices that collect vast and heterogeneous volumes of various sensory data in smart cities. These social and technological infrastructural advances potentiate the public health and prevention aspects of smart cities, transforming the urban public health from a reactive to a predictive system. In the specific area of support for (active and healthy) ageing, the transformation and progress direction of particular interest is the expansion of concept of **ambient-assisted environments**, from currently predominant implementations in residential and social indoor spaces (homes, elderly care/community centers) to outdoor and public environments. However, there is still a large gap between potential and actual IoT data exploitation because of many challenges that have to be overcome before putting data driven predictive and preventive models in geriatric or healthcare practice [1, 2].

Research presented in this paper is mainly the part of **City4Age project** (www.city4ageproject.eu) that develops age-friendly **Cities and Environments in** deployments in six different prominent pilot smart cities in EU and beyond—Athens, Birmingham, Lecce, Madrid,

Montpellier and Singapore. **The main goal** of this research and City4Age project is to develop a framework for predictive and preventive risk control of Frailty and MCI, as one of the core system infrastructure assets of age-friendly cities. **Specific goals** are development of methods based on smart cities IoT data for early risk detection and enrichment of traditional geriatric instruments. In order to achieve these goals, we are faced with several **challenges**: (1) identification and characterization of temporal behavioral patterns from sensor data, (2) Identification of behavior changes (transitions) and (3) Anomalous behavior and anomalous data detection.

Given that IoT data are collected from smart devices in form of unlabeled data streams, for initial behavior variation detection models, unsupervised machine learning techniques have to be employed. From data analytics point of view, behavior can be defined as alternating pattern of sequences of activities. Based on this definition, clustering is identified as natural technique for behavioral pattern recognition, change and anomaly detection. Clustering techniques that allow grouping objects into homogenous groups where objects in the same group are similar (intra-cluster distance is low) and objects between groups are dissimilar (inter-cluster distance is high). For building cluster models, we employed Hidden Markov models (HMMs), since they allow direct modeling of time series, provide framework for anomaly detection and have high degree of interpretability. Interpretability is very important property for incorporation of data driven models in healthcare practice and integration with domain knowledge of geriatricians.

Contributions of this chapter are twofold: (1) we propose a framework for behavior characterization, change and anomaly detection in IoT data in smart cities environment and (2) we provide first experimental evidence of usefulness of data driven modeling of behavioral data on collected City4Age IoT data.

2. State-of-the-art

World Health Organization (WHO) had recognized the importance as well as human and economic impact of age-friendly environments and launched the age-friendly Cities and Communities Programme that introduced the terms in 2006/2007, as the foundation initiative aimed for local and metropolitan governance and development levels. The European Commission (EC) supports the pursues of goals and objectives of age-friendly environments and sustainable development by numerous different instruments, primarily through R&D funding programs such as Horizon 2020 or specialized Active and Assistive Living (AAL) Programme. Important higher-level EC initiatives that foster innovation and concentrate stakeholder efforts are the recently established:

- European Innovation Partnership on Smart Cities and Communities (EIP on SCC), involving almost 400 committed cities and other partners, with a marketplace of specialized initiatives, solutions and tools.
- European Innovation Partnership on active and healthy ageing (EIP on AHA), first established EIP, in 2011, with specialized dedicated groups A3 for Functional decline & frailty, and D4 for age-friendly environments, among others.

These efforts increased research efforts in the area of smart city IoT data analytics through different projects.

Geriatric practice has in this aim utilized different standardized instruments based on traditional data collection methods (administration of questionnaires, meter-based measurement or direct observation in controlled conditions) and quantification and categorization of functional domains of daily life behavior and known frailty/MCI risk factors, such as Lawton IADL scale, Mini-Mental State Examination (MMSE), Fried Frailty Index, Nottingham Extended Activities of Daily Living, and numerous others. A comprehensive summary of such traditional generally **psychogeriatric instruments** and methods is provided in [3]. These instruments have evident major drawbacks, of late detection and problem identification (analysis and interpretation of questionnaires or conducting of exams can span intervals of months), and being generally ineffective, possibly subjective to a high degree, and costly for deployment.

The **City4Age project** (www.city4ageproject.eu), funded through the mentioned EC Horizon 2020 programme, is one of the pioneering efforts acting as a bridge between the mentioned two European Innovation Partnerships, EIP on SCC and EIP on AHA, contributing to specific and shared objectives and involving the committed participants from both Partnerships. The primary aim of the project is to enable fully Ambient Assisted age-friendly cities, through development and deployment of a range of ICT tools and services that will improve the unobtrusive early detection of MCI/frailty risks from heterogeneous IoT and smart city data sources at homes or on the move within the city, comprising the research and development work performed and results presented in this chapter as part of the work on the Data Analytics Platform. Coupled with the appropriate interventions—the developed tools will mitigate the detected risks as secondary aim. The developed system and components are being validated through in-situ deployments in six pilot smart cities.

Besides the City4Age project, there are numerous other related efforts in development of IoT driven systems for maintaining the functional independence and quality of life of the globally growing elderly population, or in development of data-driven health-related behavior recognition systems or platforms. Some of the recent relevant ones are the following:

The ActivAGE project (www.activageproject.eu), started in January 2017 and likewise funded through the Horizon 2020 programme, is a European multi-centric large-scale pilot on Smart Living Environments. The main objective is to build the first European IoT ecosystem across nine Deployment Sites (DS) in seven European countries, reusing and scaling up underlying open and proprietary IoT platforms, technologies and standards, and integrating new interfaces needed to provide interoperability across these heterogeneous platforms, that will enable the deployment and operation at large scale of active & healthy ageing IoT-based solutions and services.

Participatory Urban Living for Sustainable Environments (PULSE) project (www.pulseproject.eu), started in January 2016 and likewise funded through the Horizon 2020 programme, harvests open city data, and data from health systems, urban and remote sensors, and personal devices, to enable evidence-driven and timely management of public health events and processes, leveraging diverse data sources and big data analytics to transform urban public health from a reactive to a predictive system, and from a system focused on surveillance to an

inclusive and collaborative system supporting health equity. The clinical focus of the project is on chronic respiratory (asthma) and metabolic diseases (type 2 Diabetes), developing risk stratification models based on risk factors in each urban location (pilot deployment in five global cities—Barcelona, Birmingham, Paris, New York and Singapore), taking account of biological, behavioral, social and environmental risk factors, community resilience and well-being in cities.

IGERT project, ended in 2016, funded by the US National Science Foundation grant, comprises a multi-disciplinary doctoral training programme focused on designing and studying health-assistive smart environments, with particular emphasis on automatic monitoring and analysis of human health and behavior, unsupervised data-driven detection of activity/behavior and lifestyle changes, potential simulation/prediction of human behavior and activities, and enhancement of human physical and cognitive abilities [4].

Recently exhaustive and comprehensive reviews about temporal clustering algorithms and applications are published [5–7] and thus we will focus here only on the concepts that are closest to this research. As said behavior recognition, change and anomaly detection can be modeled naturally with clustering algorithms. Clustering techniques that allow grouping objects into homogenous groups where objects in the same group are similar (intra-cluster distance is low) and objects between groups are dissimilar (inter-cluster distance is high). Since the definition of clustering is based on the notion of similarity it is utterly important to define the notion of similarity and types of similarity measures. Unlike stationary data, time series have several aspects of similarity [7]:

Similarity in time: this is the simplest form of similarity with the assumption that instances that are close in time have similar values. This is a naïve assumption and is used for benchmark purposes in most of the cases.

Similarity in shape: these similarities disregard the time of occurrence of patterns. Using this definition, clusters of time-series with similar patterns of change are constructed regardless of time points—for example, extracting groups of elderly citizens who have a common pattern in their visits to pharmacy, regardless when these pharmacy visits occur in time-series. Dynamic time warping (DTW) is one of the mostly used dissimilarity measures of this type.

Structural similarity: the types of metrics used to find similarities in changes of time series. This is done by building models like AutoRegressive Moving Average (ARMA) or Hidden Markov models (HMMs), and the similarity is measured between parameters of models. In case of intra-activity behavior variations, structural similarity could recognize patterns such as: after three weeks of decrease of outdoor_time, citizen has registered increased outdoor_time, and this behavior is repeated every month.

3. City4Age analytic framework

Main challenges for the Data Science and Analytics in the related research in City4Age coincide with main generic challenges for the potential of collected IoT data from smart cities for health (and other personal) monitoring—volume and diversity of collected data is huge and

promising, but the development of formalized and applicable knowledge and learning models is lagging with adequate potential for interpretation, classification and exploitation. At the same time, the unobtrusively acquired dataset for each specific person over time is often sparse, incomplete and erroneous, and with high degree of variation caused by temporary sensor imprecisions or influence of external factors beyond the sensing or modeling scope. High-level City4Age Analytics and detection process flow is depicted on **Figure 1**. City4Age has from the beginning adopted the combined hybrid knowledge- and data-driven approach, with the initial contribution of the knowledge-based approach turning out somewhat overestimated in the meantime due to the issues mentioned above and consequent non-deterministic and volatile semantic integrity of "known" or presumed universal geriatric causalities and concepts. The main focus is thus currently on the **data driven** behavior change variation recognition and characterization, analysis of relative changes in time series data for each specific person since the start of the pilot monitoring and determining baseline referent points, values and features (individual geriatric care analytics), and subsequently on discovering correlations and underlying features and interdependencies in the complete studied and monitored populations and clusters and groups within it (group exploratory analytics), starting from minimal initial domain model knowledge.

Majority of the functional domains and parameters of daily life behavior and geriatric risk are nevertheless known and established, and formalized in the City4Age hierarchic computational model of geriatric behavior and risk [8].

The main model constructs and variables are based on the notions of Geriatric factors (GEF), representing monthly behavior characterizations from all various functional behavioral domain variables and known MCI/frailty risk indicators, on unified Likert scale, with 1 denoting the least favorable and five the most favorable behavior with respect to MCI and Frailty risk, a common and standard adopted representation in geriatric practice and many of the used traditional instruments and questionnaires. GEF are further structured on several hierarchic levels of decomposition (GES—geriatric sub-factors, GFG—geriatric factor groups), and can be synthesized or derived from "Measures," native numeric values generated by the

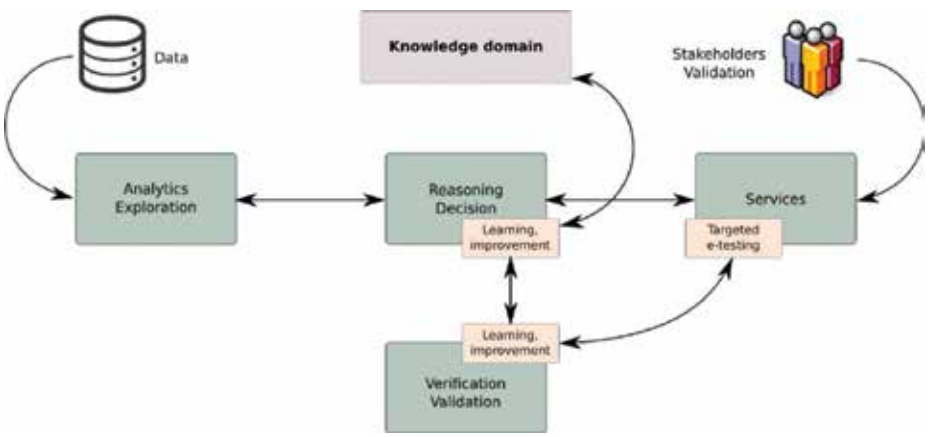

Figure 1. High-level City4Age analytics and detection process flow.

various sensing technologies and methods for collecting data (e.g., daily number of walked steps, weekly number of visits to relatives, daily time in seconds spent in public transport, etc.), as exemplified on the diagram above. Measures are analyzed and processed using various algorithmic techniques and/or methods, some of which are the clustering and grading algorithms described in this chapter, but others have also been tried and tested, and therein is the flexibility and scalability of the model and the Analytics Framework, supporting the registration of various algorithmic methods through metadata and deploying them on the "detection" variables (GFG, GEF, GES, Measures) on various model hierarchy and derivation levels. Example of network representation of GES, GEF and Measures is presented on the diagram on **Figure 2** below. Relations between nodes shown on the diagram are not fixed/persistent, can variate according to different model configurations in different cities, or adaptively according to the results of the data-driven detection.

The results of the data-driven detection are in turn used for expanding and building-up the domain knowledge base. The structure (ontologies and semantics) and mechanisms for this are established [9] are in parallel ongoing development, and will be still more intensely in the future work, in the scope and frame of City4Age contributions and breakthroughs in establishment of data-driven geriatrics. The unobtrusively acquired temporal dataset on individual level, currently being acquired for each single elderly person, is highly likely to expand with the increase and improvement of deployment scope and reliability of data acquisition and detection infrastructure and technologies.

Figure 2. Example network representation of the City4Age geriatric model main constructs: Measures (purple), GES (green), and GEF (red).

4. Hidden Markov models for behavioral modeling of smart cities IoT data

As discussed, the main tasks of City4Age analytic framework are recognition of behavioral patterns, behavior changes (transitions) in time and anomaly detection. Additionally, models derived from data should be interpretable in order to integrate data driven insights with domain knowledge expertise. Hidden Markov models (HMMs) provide a framework for all main tasks and thus we employed these models for behavior variation analyses. Additionally, HMMs allow prediction of identified behavioral patterns in future and this adds predictive and preventive component in analytic framework. Here we will consider first order HMMs where each temporal state depends only on one previous state. This is strong assumption, but allows development of scalable models and real-time inference. **Figure 3** describes first order Markov chain where each state x depends on previous state (x-1) and observed data (y).

Hidden Markov models can be explained as total probability of X and Y by following formulae:

$$p(X,Y) = p(x_1)\prod_{T-1}^{t=1} p(x_{t+1} \mid x_t)\prod_{T}^{t'=1} p(y_{t'} \mid x_{t'}) \quad (1)$$

where $p(y_{t'} \mid x_t)$ represents observation probability, while $p(x_{t+1} \mid x_t)$ prepresents transition probability.

In our case observations are series of IoT sensory data while hidden states represent categorized, homogenous series parts (that wil.l be characterized as behavioral patterns or behaviors). This is why we use Gaussian HMMs that characterize states with Gaussian distributions. This is depicted on **Figure 4**.

Each HMM model is thus constituted from three elements:

1. Prior probability distribution of hidden states (vector π) that describes how frequently each state occurs in general.
2. Transition matrix ($A_{i,j}$) that describe the transition probabilities from one state to another.
3. Probability distribution functions (one for each state) with corresponding parameters. In our case Gaussian distributions are modeled and thus means and standard deviations are used for definition of hidden state (behavior) probability distribution. HMMs allow modeling of discrete data too, but in that case probability distributions are represented by conditional distributions.

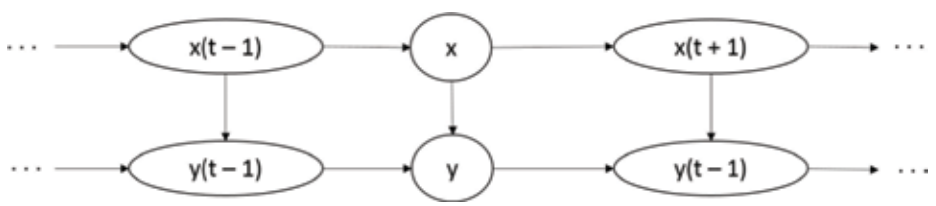

Figure 3. First-order Markov chain.

http://dx.doi.org/10.5772/intechopen.75203

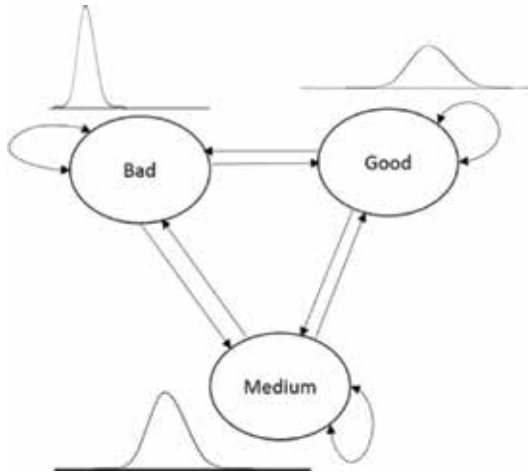

Figure 4. Behavior modeling with Gaussian HMMs.

Based on HMM definition, we can work on following tasks [10]:

Training—Learning parameters of HMM (A, B, and the prior distribution π), given a training sequence of observations $y_1, y_2, \ldots, y_T$. By solving this task, we will be able to characterize behavioral patterns (distributions). This task is solved by forward-backward algorithm.

Decoding—given an observation sequence and an HMM, determine the most probable hidden state (behavior) sequence. We used this task for state prediction and model evaluation. This task is solved by Viterbi (backward algorithm).

Likelihood—Calculation of probability that given sequence originates from given HMM model. In this research, we did not work on this task, since we built personalized behavioral models, but it will be used in later stages of the project when we will model behavior of groups of care recipients.

5. Framework for behavioral pattern recognition and change detection

Based on definition of behavior as pattern of sequences of activities and corresponding measure values, clustering algorithms emerge as natural algorithmic approach for behavioral pattern recognition and change detection. In City4Age setting, inputs for clustering algorithms are time series. These time series can be represented by values of activity measures, GES, GEF or Geriatric Score of care recipients. Based on time series, temporal clustering algorithms can identify patterns (similar time series values in consecutive time-steps) that are repeated over time. We characterize these patterns as behaviors and transition between patterns, behavior changes. Very important component in derivation of GES and GEF from activity measures are numerical indicators (NUIs). NUIs represent aggregations (e.g., mean, std., trend, etc.) of activity measure values on monthly level. This granularity level is convenient since it allows direct

conversion of NUIs to GES and GEF that are interpretable to geriatricians. However, monthly statistics in some cases do not capture important within month variations in time series.

This is why, in contrast to NUIs, clusters are not restricted to monthly level. Depending on input data, clusters can be identified on daily or monthly level. For example, if number_of_steps activity measure is clustered over time, model can identify similar groups of daily values: days with high values (i.e. average of 3000 steps with standard deviation of 200 steps) and days with low values (i.e. average of 600 steps with standard deviation of 100 steps). Similarly, cluster models can identify patterns of series of GES or GEF. For example: care recipient have periods of time where motility have average motility value of 4.1 with standard deviation of 0.2. So, behavioral patterns encapsulated in cluster models provide characterization of behavior on finer grade than monthly level. Additionally, the level of granularities does not have to be defined in advance (e.g., weeks).

For example, in first 22 days of January, care recipient had high values and high variability of number_of_steps, but in next 12 days he or she had low values with low variability. Even though, behavioral patterns described by clusters are not necessarily aligned with monthly representations of NUIs, GES and GEF, they can be exploited for definition of new NUIs that will capture within month behavior variations (e.g., care recipient showed improved behavior in last eight days of a month). NUIs based can be further graded as described in previous sub-section. Based on previous examples it is intuitively clear that clusters (behavior patterns) encapsulate smaller variations of time series and allows data driven discretization and characterization of discrete categories. This discretization allows easier inspection of behavioral changes (than observing unclustered series with many variations) and thus, results of clustering (cluster labels) can be directly represented on City4Age interactive dashboards or used as NUIs for derivations of GES and GEF (**Figure 5**). This way Geriatricians can access visualizations of natural groupings among series data, and label behaviors (e.g., normal, bad or good or on the Likert scale) or revise data driven grading of clusters.

It is important to emphasize that grouping of activities and/or citizens will play an important role in extending of City4Age interactive dashboards, with mentioned easier identification

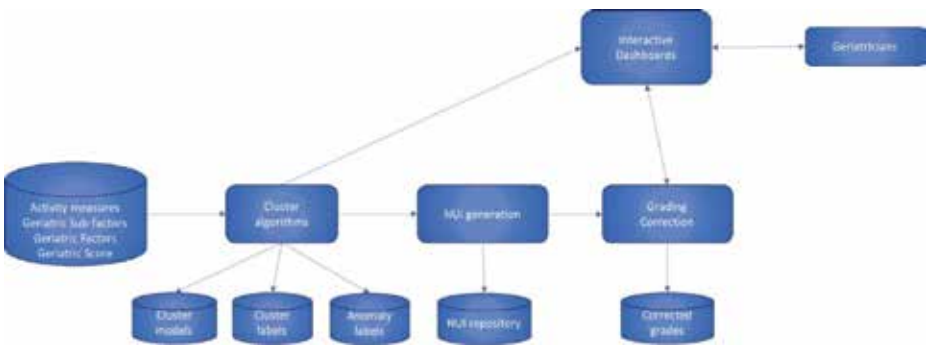

Figure 5. Cluster model flow.

and labeling of patterns in streaming data. The labels will allow development of supervised models and further automation of City4Age analytics processes and improvement of alarming systems.

6. Experiments

In order to evaluate proposed framework we conducted experiments collected from City4Age Pilot sites. The main goal of the experimental evaluation was to evaluate usefulness of HMM models for behavior recognition, behavior change and anomaly detection in context of City4Age IoT data. We will describe process of data collection, modeling and evaluation in following text.

6.1. Data collection and preparation

Data used in experiments originate from the Birmingham Pilot. Data have been acquired by monitoring 3 Care recipients during a 6-month period (from January to July 2017, and ongoing). Sensory data are collected using Nokia Steel (e.g., Withings Activité) smartwatches. Nokia Steel tracks the following activities: sleep cycle, movements tracking, walked steps and distance, burned calories, elevation, heart rate, and optionally SP02 (peripheral capillary oxygen saturation, an estimate of the amount of oxygen in the blood, taken with additional pulse oximeter). Integration of sensor data with City4Age analytics platform is described in the following text. The proximity positioning data are gathered through smartphone BLE transceiver and relayed through the smartphone 4G connection to the City4Age Platform. Nokia/Withings API is used for initial pre-processing step on the sleep, activity, and other data obtained from the smartwatches, before sending to the City4Age Platform. So, input for building clustering algorithms in this research was sets of activity measures for each citizen. Summary of observed activity measures is presented in **Table 1**.

6.2. Experimental setup

The main goal of our experiments was to show that HMM models can be efficiently used for behavioral pattern recognition, behavior change detection and anomaly detection. In order to achieve this goal we faced several challenges: identification of adequate model evaluation (selection) measure, identify optimal number of behavioral states for each care recipient and each activity and finally to characterize identified behaviors (clusters or behavioral patterns). Since HMM models cannot implicitly learn optimal number of hidden states, we built HMM models with varying number of clusters (in the range 2–10) for each care recipient and each activity. Additionally, since there is no consensus for evaluation of cluster models in unsupervised setting, each model was evaluated with log likelihood, BIC and AIC evaluation measures. So setting we conducted 810 experiments in total (3 care recipients × 10 activities × 9 variations of state numbers × 3 evaluation measures). Each experiment lasted for 15–24 s (including learning and evaluation). Since HMM is one of the most scalable algorithm from

Geriatric sub-factor	Activity	Measure unit
Walking	WALK_STEPS	# of steps
	WALK_DISTANCE	meters
Quality of sleep	SLEEP_LIGHT_TIME	seconds
	SLEEP_DEEP_TIME	seconds
	SLEEP_AWAKE_TIME	seconds
	SLEEP_WAKEUP_NUM	seconds
	SLEEP_TOSLEEP_TIME	seconds
Physical activity	PHYSICALACTIVITY_SOFT_TIME	seconds
	PHYSICALACTIVITY_MODERATE_TIME	seconds
	PHYSICALACTIVITY_INTENSE_TIME	seconds
	PHYSICALACTIVITY_CALORIES	# of calories

Table 1. Observed activity measurements.

Probabilistic Graphical models family (it is frequently used for signal processing and speech recognition) it allows adaption for much larger series as City4Age streaming data arrives. After building models, they are applied to activity measure time series for each citizen and each activity. In this way we labeled each time point with cluster (behavioral pattern or state) assignment. When scoring HMM models, probabilities that time point originates from cluster distributions are identified and largest probabilities are stored for anomaly detection purposes. Experimental setup is implemented in Python. Hmmlearn library is used for building HMM models while Pandas DataFrame is used for data manipulation. All experiments are conducted on a testing cloud comprising three servers with quad-core Intel Xeon class CPU each, 8 GB of RAM combined for data storage processes and up to 252 GB of RAM combined at disposal for data analytics and applicative processes.

6.3. Results and discussion

In this section we will analyze and discuss experimental results from the aspects of identification of adequate model selector, behavioral pattern recognition, behavioral change (transition) recognition and anomaly detection.

6.3.1. Identification of adequate model selector

Since there is no consensus about the best HMM model selection and evaluation metric in unsupervised setting, our first objective was to identify well suited metric for data at hand. Good metric should enable automated identification of parsimonious solutions: ones with high performance but as less complex as possible. For that purpose, we inspected general behavior of AIC, BIC over all experiments (care recipients and activity measures) and correlated these values with log likelihood performances. Log likelihood measures how probable is model given the series data. It is intuitively clear that models with maximum possible log

http://dx.doi.org/10.5772/intechopen.75203

likelihood are desired, however in general, likelihood monotonically increases with increase of model complexity. This means that larger number of clusters will almost always be preferred by log likelihood criteria. The aim of the first analyses was to inspect how AIC and BIC measurements capture degree of changes (slope) in likelihood values of the model. Distributions of average values of log likelihood, AIC and BIC over different model complexities (numbers of states) are shown on **Figure 6**.

On X-axis numbers of clusters are showed and on Y-axis average AIC, BIC and log likelihood values (over all experiments), respectively. It can be seen on figure below that AIC values follow adequately identify steep growth of log likelihood on log likelihood curve. Meaning that average AIC shows better model performance while log likelihood performance increases in large steps.

Optimal number of clusters (in average over all experiments) according to AIC measure is 5 where "elbow" in AIC curve is detected. This point corresponds to transition from higher growth (for number of clusters 2–5) of log likelihood to Lower growth (for number of clusters 6–10). On the other side, BIC model selector, ignores steep increase of log likelihood and identifies three as optimal number of clusters.

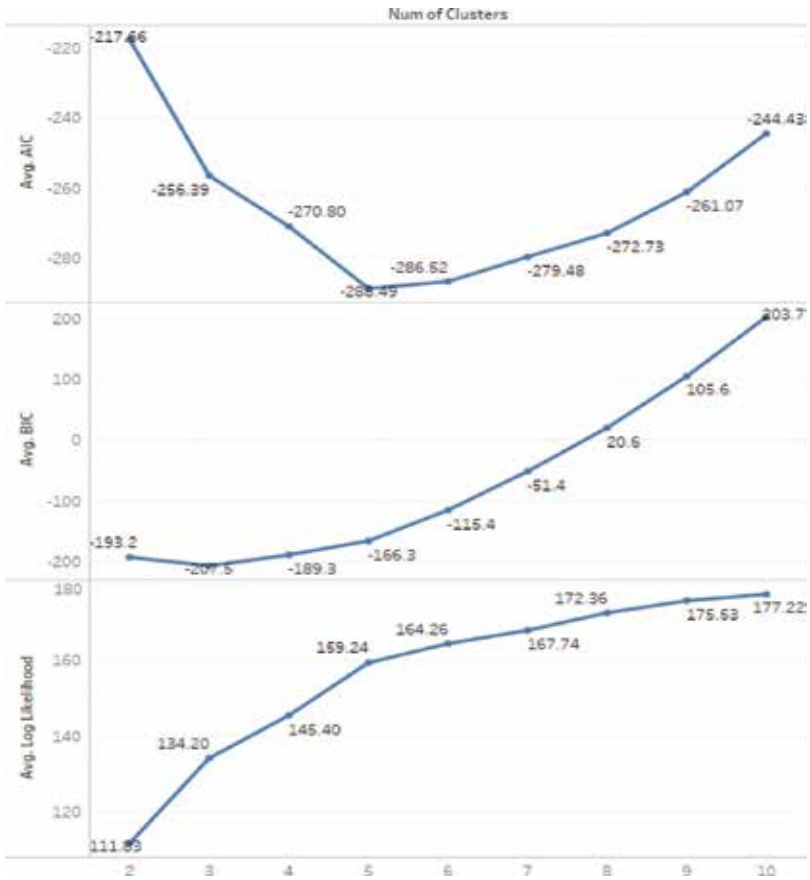

Figure 6. Distribution of average values of log likelihood, AIC, and BIC over different model complexities.

After this point, BIC curve grows super linearly meaning that it does not prefer models with higher number of clusters than 2 or 3. Deeper inspection of AIC, BIC and log likelihood curves for each care recipient and each activity showed consistent behavior with ones described on **Figure 6**. Thus we selected AIC as measure of choice for HMM model selection. Based on previous discussion we took AIC as measure of choice for model selection and identification of optimal number of behavioral state for each care recipient and each activity.

However, it is very important to emphasize that insights presented in previous text cannot be considered as conclusive and cannot generalize over all problems. This is because cluster performance is dependent on data distributions that are different for each dataset, but also because depends on the context of analyses.

6.3.2. Choosing optimal number of clusters

Based on previous insights, we used AIC measure to analyze quality of the models with respect to number of clusters. Results for each activity for one care recipient are shown on **Figure 7**. It can be seen from **Figure 7** that different activities have different "optimal" number of clusters. In this analyses term "optimal" have to be considered very loosely because, in many cases difference in AIC performance is very similar for different number of clusters. This means that for behavior analysis purposes adequate model can be selected in range of models with good and similar AIC performance. Most often, parsimonious solution is applied: model with satisfying performance and the least number of cluster is selected. On the other hand, in case of existence of global saddle point model selection is clearer process. Saddle points have strict mathematical definition based on function derivatives, but in this case, saddle point can be descriptively defined as: point with property that all points from the left side (lower number of clusters) are larger and all points from the right side (higher number of clusters) are larger. In these situations, model selection is based on minimal (optimal value of AIC). Clear example of saddle point on **Figure 7** is labeled with k = 4 for physical_activity_calories activity measure.

6.3.3. Behavior characterization

Figure 8 depicts behavioral patterns for activity sleep_light_time for one care recipient identified by HMM. X-axis represents temporal dimension in day units is presented for the period and Y-axis represents cumulative duration of sleep_light time for each day.

It can be seen that HMM model based on AIC model selection criteria identified three different clusters (behavioral patterns) that can be characterized as following:

1. Behavior (purple line): medium values of sleep_light_time (between 8000 and 13000 s) with low deviations,
2. Behavior (green line): high values of sleep_light_time (between 13000 and 20000 s) with low deviations and
3. Behavior (red line): low values of sleep_light_time (between 0 and 15000 s) with high deviations.

http://dx.doi.org/10.5772/intechopen.75203

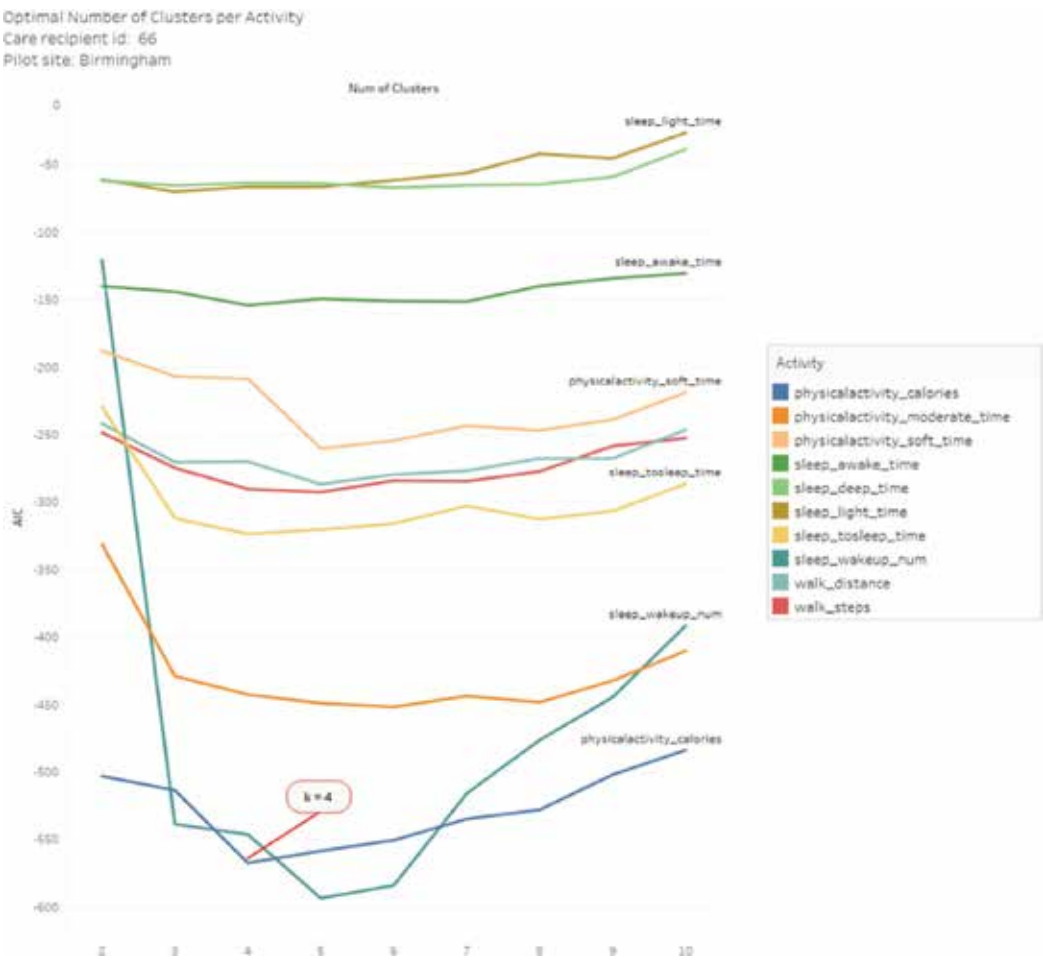

Figure 7. Selection of "optimal" number of behavioral patterns based on AIC values.

Normal sleeping process includes interchange of light sleep and deep sleep. First and second behaviors are considered desirable and such times of light sleep lead to mitigation of frailty risk. On the other hand lack of light sleep time and high variations are considered as negative behavior and could indicate increase of stress and chance of MCI/frailty risk development. Based on these observations behavioral patterns are quantified and ordered (e.g., 1—worst behavior, 2—medium behavior, and 3—good behavior) and pushed in further process of risk quantification through derivation of numerical indicators and grading (described in previous section).

6.3.4. Behavior variation change and anomaly detection

After characterization of behavioral patterns, we analyzed behavior (pattern) changes over time. Identification and characterization of behavior changes (transitions) over time is crucial step for building proactive systems and providing timely and preventive interventions. **Figure 9** describes transitions of behaviors identified in previous sub-section.

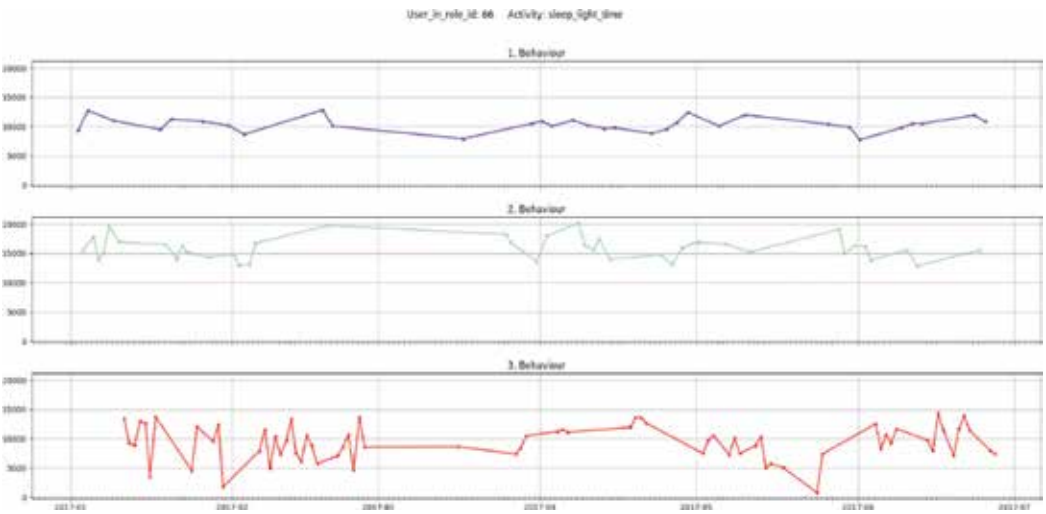

Figure 8. Behavioral patterns identified by HMM model.

Frequent pattern changes from **Figure 9** can be observed from green ("good" behavior) to red ("bad" behavior) lines. It can also be observed that red behavior appears more frequently than other two.

Finally, in most cases "medium" behavior (purple line) transitions to "good" behavior (green line). Based on this analysis it can be observed that after behavior improvement (from "medium" to "good") care recipients often have sudden worsening of behavior. Recognition of such transitional patterns enables predictive and preventive approach in risk prevention. Namely, HMM models, based on transitional probability matrices identify probabilities of

Figure 9. Behavior variations (transitions) and anomalous point.

http://dx.doi.org/10.5772/intechopen.75203

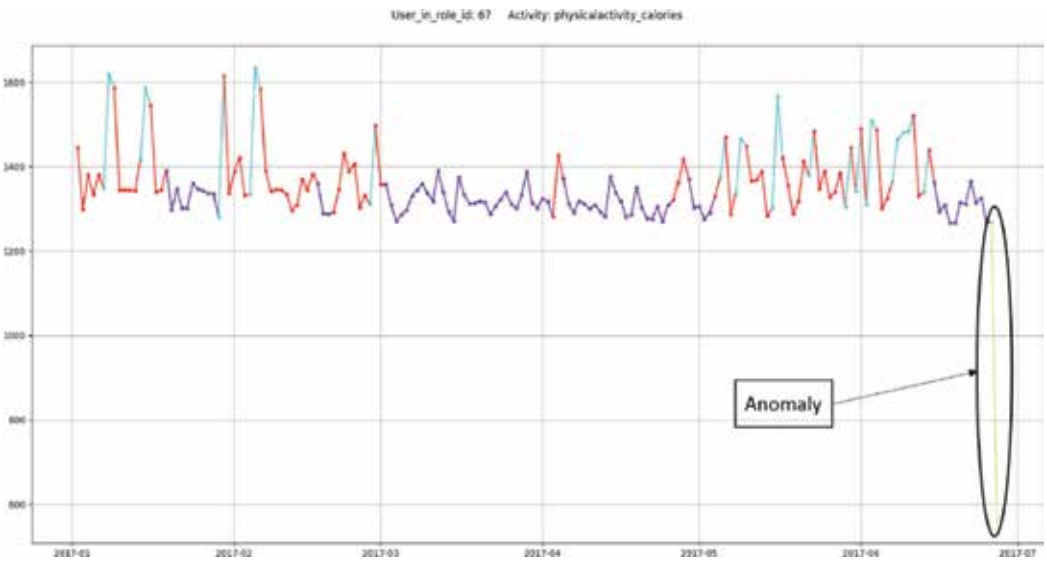

Figure 10. Anomalous state.

behavior transitions and if behaviors are characterized well, these probabilities can be used as early risk identification indicators. Furthermore, based on HMM, model anomalies can be automatically identified per user defined thresholds. For example, by manual labeling on behavioral series presented on **Figure 9**, the lowest point of bad behavior (red line between 2017-05 and 2017-06) is identified. This point is captured as anomalous based on probability threshold of 70%. This means that behavioral point (instance) has max. probability of belonging to any state less than 70%. Experiments on all other activities showed that optimal value of threshold should be between 65 and 75%. Similarly, anomalous states (behaviors) can be identified by setting threshold for minimum number of instances (behavioral measurements) that should constitute behavior (cluster). Since number of behavior measurements is variable for different users, activities and even periods of measurements, we define threshold as percentage of total number of measurements for selected period. In all our experiments series were constituted from 140 to 180 measurements. Experiments showed that good anomaly scoring is achieved by setting threshold to 3–5%. **Figure 10** illustrates situation where anomalous behavior is detected (last two measurements connected with yellow line).

7. Conclusion and future work

In this chapter we addressed the problem of behavioral pattern recognition, behavior change detection and anomaly detection based on IoT data in smart city environment. We proposed a framework for behavioral change detection that will be utilized in context of mild cognitive impairment (MCI) and frailty risk assessment and detection in the City4Age project. Behavioral modeling and risk assessment for MCI and Frailty are very challenging tasks because of the large variations between each specific personal case, and the practical lack

of universally agreed and adopted criteria in geriatric practice (in real-life environment, not controlled "lab" settings) on the referent thresholds or ranges of quantified risk factors or geriatric domain variables that actually denote certain MCI/frailty risk or potential onset.

Thus we developed data driven models based on HMMs that exploit IoT sensory data and allow automated behavior recognition, change and anomaly detection. Models are used for characterization of data that serves as an input for exploratory analytics through interactive dashboarding and/or enrichment of modeled Geriatric factors that quantify the specific behavior characterizations and risk levels for MCI and Frailty.

In future work, we will integrate results from this research in City4Age interactive monitoring dashboards and thus enable geriatricians to gain additional insights into care recipients behavior and potential risk. This will open the space for supervised behavioral scoring and risk prediction. Further, we will develop data driven behavioral models for multivariate IoT data series and explore mutual influence between series. Finally, we will evaluate more unsupervised models for behavioral modeling including deep learning models (e.g., recurrent neural networks) in the analyses.

Acknowledgements

Main body of this research is part of the European project City4Age that received funding from the Horizon 2020 research and innovation funding programme, under grant agreement number 689731.

Author details

Vladimir Urosevic[1], Ana Kovacevic[2], Firas Kaddachi[3] and Milan Vukicevic[4]*

*Address all correspondence to: vukicevicm@fon.bg.ac.rs

1 Belit Ltd., Belgrade, Serbia

2 Big Data Analytics, Belgrade, Serbia

3 Montpellier Laboratory of Informatics, Robotics and Microelectronics (LIRMM), Montpellier, France

4 Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia

References

[1] Van Poucke S, Zhang Z, Schmitz M, Vukicevic M, Vander Laenen M, Celi LA, De Deyne C. Scalable predictive analysis in critically ill patients using a visual open data analysis platform. PloS One. 2016b;**11**(1):e0145791

[2] Van Poucke S, Thomeer M, Heath J, Vukicevic M. Are randomized controlled trials the (G) old Standard? From clinical intelligence to prescriptive analytics. Journal of Medical Internet Research. 2016;**18**(7):e185. DOI: 10.2196/jmir.5549. PMID: 27383622, PMCID: 4954919. http://www.jmir.org/2016/7/e185

[3] Kaddachi F, Aloulou H, Abdulrazak B, Fraisse P, Mokhtari M. Unobtrusive Technological Approach for Continuous Behavior Change Detection Toward Better Adaptation of Clinical Assessments and Interventions for Elderly People. In: International Conference on Smart Homes and Health Telematics. Cham: Springer; 2017, August. pp. 21-33

[4] Sprint G, Cook DJ, Schmitter-Edgecombe M. Unsupervised detection and analysis of changes in everyday physical activity data. Journal of Biomedical Informatics. 2016 July;**63**:54-65. ISSN: 1532-0464

[5] Gupta M, Gao J, Aggarwal CC, Han J. Outlier detection for temporal data: A survey. IEEE Transactions on Knowledge and Data Engineering. 2014;**26**(9):2250-2267

[6] Silva JA, Faria ER, Barros RC, Hruschka ER, de Carvalho AC, Gama J. Data stream clustering: A survey. ACM Computing Surveys (CSUR). 2013;**46**(1):13

[7] Aghabozorgi S, Shirkhorshidi AS, Wah TY. Time-series clustering–A decade review. Information Systems. 2015;**53**:16-38

[8] Copelli S, Mercalli M, Ricevuti G, Venturini L. City4Age frailty and MCI risk model, v2. City4Age Project Public Deliverable D. 2017 October;**2**(06)

[9] Azkune G, Almeida A, López-de-Ipiña D, Chen L. Extending knowledge-driven activity models through data-driven learning techniques. Expert Systems with Applications. 2015 April;**42**(6):3115-3128. ISSN: 0957-4174

[10] Bilmes JA. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. International Computer Science Institute. 1998;**4**(510):126

Chapter 8

A Class of Parametric Tree-Based Clustering Methods

Fred Glover and Yang Wang

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.76406

Abstract

We introduce a class of tree-based clustering methods based on a single parameter W and show how to generate the full collection of cluster sets C(W), without duplication, by varying W according to conditions identified during the algorithm's execution. The number of clusters within C(W) for a given W is determined automatically, using a graph representation in which cluster elements are represented by nodes and their pairwise connections are represented by edges. We identify features of the clusters produced which lead to special procedures to accelerate the computation. Finally, we introduce a related node-based variant of the algorithm based on a parameter Y which can be used to generate clusters with complementary features, and a method that combines both variants based on a parameter Z and a weight that determines the contribution of each variant.

Keywords: clustering, minimum spanning trees, spanning forests, machine learning, big data analytics

1. Introduction

Clustering methods have long been a mainstay of statistics and machine learning [1–3], and have experienced a surge in importance with the advent of Big Data Analytics [4, 5]. A highly successful use of clustering in practical applications has been to seek out particular kinds of clustering methods that are effective in particular settings, based on the finding that different classes of problems respond best to specific classes of clustering methods. This finding motivates the work of this paper, which introduces a new class of tree-based clustering methods with an ability to modify the kinds of clusters produced by changing the value of a particular parameter. Moreover, we show all members of class can be generated without duplication by a process that adaptively determines each new parameter value from the information produced by executing the class member that precedes it.

We are motivated to use a tree-based algorithm due to their applications in genome analysis [6–8], image segmentation [9, 10], statistics [11] and microaggregation [12]. The most common forms of the tree-based clustering methods in the literature [8, 13–15] begin with a minimum spanning tree and then successively delete edges according to various criteria. However, our approach has a greater level of flexibility than these commonly applied methods due to the fact that the clusters produced include those that cannot be obtained by removing edges of a minimum spanning tree.

We introduce special techniques for accelerating the execution of our basic approach by exploiting its underlying properties and then introduce a closely related clustering algorithm that replaces an "edge-based" focus with a complementary "node-based" focus. We unify these two classes of approaches by identifying a third class that marries their complementary features, and which provides additional variation by means of a weight that permits the contribution of these complementary approaches to be varied along a continuum. We conclude by demonstrating how the procedures for accelerating the first method can be expressed in a more general form to accelerate the execution of the combined procedure as well.

The ability to generate a family of clustering methods from each of the three basic clustering designs by varying a single parameter (and the weight employed by the third method) invites empirical research to determine parameter ranges that are effective for specific types of clustering applications, opening the possibility to produce clusters exhibiting features different from those customarily obtained.

2. Cluster problem formulation

The clustering problem in our treatment is formulated by reference to a graph G = (N, E) where N = {1, …, n} is a set of nodes (cluster elements) and E is a set of edges (pairwise connections between elements) given by E $\subset$ N × N = {(p,q): p,q$\in$N}. The notation (p,q) is understood to represent an unordered pair (hence (p,q) = (q,p), and is equivalently represented by the set notation {p,q}). Each edge e = (p,q) $\in$ E has an associated cost (or length) denoted by c(e) (= c(p,q)). It is not necessary to assume that G is complete or connected. We also do not require that the costs c(e) be nonnegative.

The goal is to partition N into sets (clusters) N_k, k$\in$ K = {1, …, ko}, where the value ko is automatically determined by the clustering process. We also identify an associated set of edges $E_k \subset$ {(p,q), p,q$\in$ N_k}, where the subgraph (N_k,E_k) of G constitutes a min cost spanning tree over the nodes of N_k. In contrast to those tree-based clustering approaches that begin with a min cost spanning tree over all of G and selectively delete particular edges, our algorithm produces subgraphs (N_k,E_k), k $\in$ K, that may not be possible to obtain by deleting edges from such a tree.

The class of clustering methods we describe is based on specifying the value of a parameter W, whose value uniquely determines the outcome of each clustering method within the class. W is expressed as an additive threshold for selecting edges and hence nodes to be added to a current construction (collection of subgraphs), and observe that W can equally be expressed as a multiplicative threshold in the case where the costs are nonnegative and the two approaches are equivalent in this instance.

http://dx.doi.org/10.5772/intechopen.76406

We start with any selected value W = Wo≥ 0 and after obtaining a collection of clusters C(W) for a given W we systematically modify W so that over successive iterations all possible cluster collections C(W) for W ≥ Wo will be generated without duplication. The complete range of cluster collections results by choosing Wo = 0 (or Wo = 1 in the multiplicative version).

3. Algorithm to generate the cluster collections C(W)

In overview, we index the edges of E in ascending cost order so that c(e(1)) ≤ c(e(2)) ≤ … ≤ c (e(|E|)), and identify the nodes of edge e(s) by writing e(s) = (p(s), q(s)). We start with each cluster N_k consisting of just the node k, that is, each cluster is a degenerate single node tree given by.

$$N_k = \{k\}, k \in K \text{ for } K = N = \{1, \ldots, n\}$$

The associated set E_k of edges in the tree corresponding to N_k is empty ($E_k = \emptyset$). As the algorithm progresses, the composition of the clusters will change and the index set K of clusters will change accordingly.

In addition, we keep a cost value denoted by MinCost(k) for each k ∈ K which identifies the cost of the minimum cost edge e ∈ E_k. To begin, since no cluster yet contains an edge, we define MinCost(k) = Large, a large positive number, for all k ∈ K. (We will not have to examine the set E_k to identify MinCost(k) = Min(c(e): e ∈ E_k) because the structure of the algorithm will insure that MinCost(k) will equal the cost of the first edge added to E_k. In general, while we describe the composition of E_k and the manner in which it changes, the organization of the algorithm assures that it is unnecessary to keep track of E_k since the sets N_k, for k∈ K, will identify the elements in the clusters produced.)

We also maintain a list L(i) for each i∈ N that names the cluster that node i belongs to. Hence, initially, L(i) = (i) since i∈ Ni = {i} for all i∈ N. The redundancy provided by this list enables updates to be performed efficiently. Subsequently, L(i) is modified as node i becomes the member of a new cluster N_k. As this is done, the list K will come to have "holes" in it, i.e., will not consist of consecutive indexes. (At the end of the algorithm we can rename the clusters indexes, if desired, so that K = {1, 2, …, ko} where ko = |K|.)

Finally, during the process of generating the cluster collection C(W) for the current W value, we will identify a value Wnext so that the process may then be repeated for W: = Wnext to generate a new collection of clusters. As previously noted, by starting with W = Wo = 0 (or W = Wo = 1 in the multiplicative version), and then successively identifying Wnext each time a cluster collection C(W) is generated, we can ultimately generate all possible collections C(W), without duplication. The process terminates when W becomes large enough that C(W) consists of a min cost spanning tree over each connected component of G. (A simple condition for identifying this termination point is identified below.)

Building on these observations, we now state the full form of our algorithm.

C(W) Algorithm (Multiplicative Version)

Inputs: The graph G(N, E), cost vector c(e), e ∈ E, initial Wo value for W.

Edges are ordered so that the costs satisfy $c(e(1)) \leq c(e(2)) \leq \ldots \leq c(e(|E|))$.

Set W = Wo and sLast = |E|

Begin Outer Loop

While W < Large

Initialization(A). Set Wnext = Large, K = {1, …, n}, and for each k ∈ K let L(k) = k,

$N_k = \{k\}$, $E_k = \emptyset$, and MinCost(k) = Large.

Initialization(B). Let i′ = p(1) and i″ = q(1) and select e(1) (= (i′, i″)), to create the first non-degenerate cluster (containing more than one node and hence more than 0 edges) by identifying k′ = L(i′) and k″ = L(i″) and absorbing $N_{k''}$ into $N_{k'}$ to create the cluster $N_{k'}$: = $N_{k'} \cup N_{k''}$ = {i′, i″} with edge set $E_{k'}$ = e(1). Set MinCost(k′) = c(e(1)) and conclude by eliminating the superfluous cluster $N_{k''}$ (now contained within $N_{k'}$) by setting K: = K \ {k″}. Finally, initialize the edge index s by setting s = 1.

Begin Inner Loop

While s < sLast

Set s: = s + 1 and identify edge e(s) = (p(s), q(s)). Let i′ = p(s), i″ = q(s) and let k′ = L(i′) and k″ = L(i″). There are three cases:

Case (1): If k′ = k″ (i′ and i″ belong to the same cluster), then continue to the next iteration of the Inner Loop.

Case (2): If c(e(s)) > W + MinCost0, for MinCost0 = Min(MinCost(k′), MinCost(k″)), then edge e(s) is forbidden to be added to join the clusters $N_{k'}$ and $N_{k''}$ into a single cluster. In this case, compute Wnext = Min(Wnext, c(e(s)) –MinCost0) and continue to the next iteration of the Inner Loop.

Case (3) (If (1) and (2) do not apply)[1]: Absorb $N_{k''}$ into $N_{k'}$ to create the larger cluster $N_{k'}$:= $N_{k'} \cup N_{k''}$ with its associated edge set $E_{k''}$: = $E_{k'} \cup E_{k''} \cup \{e(s)\}$. Correspondingly, update L(i) by setting L(i) = k′for all i∈ Nk″, and set MinCost(k′) := Min(MinCost(k′), MinCost(k″), c(e(s)). Finally, eliminate the superfluous cluster Nk″(whose elements are now contained within $N_{k'}$) by setting K : = K \ {k″}.

Endwhile.

// The node and edge sets for the collection of clusters C(W) for the current W are given.

// by N_k and E_k for k ∈ K. The node sets can alternatively be recovered by reference to.

// the values L(i), i = 1, …, n.

W = Wnext

Endwhile

End of C(W) Algorithm

[1]Case (3) generalizes Initialization(B).

We employ the customary convention that a loop of the form "While x < Constant" will be bypassed if the beginning value of x does not satisfy "x < Constant" and that the execution of the loop will not be interrupted if x is changed so that x ≥ Constant within the loop (though the execution will then terminate at the loop's conclusion). Hence, for example, in the Inner Loop when s: = s + 1 results in s = sLast, the loop will continue its execution until the current iteration ends.

We now make several observations about the algorithm.

Remark 1: The multiplicative version of the C(W) Algorithm results by modifying Case (2) to replace W + MinCost0 by W·MinCost0 and to replace Wnext = Min(Wnext, c(e(s)) – MinCost0) by Wnext = Min(Wnext, c(e(s))/MinCost0). (Hence, addition is replaced by multiplication and subtraction is replaced by division.) These approaches will generate the same collection of clusters under the assumption that all c(e) > 0 for the following reason: a positive value W′ can always be found for the multiplicative case that will cause Wnext to screen out the same set of elements as any positive value W for the additive case, and vice versa. This relationship can also be extended to cover the situation where all c(e)are nonnegative.

Remark 2: The assignment W = Wnext at the end of the outer loop can be replaced by setting W:= Wnext + Δ for a chosen increment Δ to generate only a subset of the possible C(W) collections. Experimentation with a given class of cluster applications may additionally lead to identifying upper and lower bounds on W (or specific intervals for W) that prove most effective for that class.

Remark 3: To reduce the updating effort of Case (3), the indexes i′ = p(s) and i″ = q(s) can be interchanged (hence also interchanging k′ and k″) to assure that |Nq(s)| ≤ |Np(s)|. (More comprehensive ways of reducing computation are identified in Sections 4 and 7.)

Remark 4: The justification of terminating the outer loop of the algorithm when W = Large (after setting W = Wnextat the conclusion of the inner loop) derives from the observation that Wnext = Large implies the condition c(e(s)) > W + MinCost0 is never satisfied in Case (2). (When this terminating condition occurs in a connected graph, the method will have generated a min cost spanning tree.) Moreover, if the algorithm is repeated for W = Large, the same outcome will result.

Remark 5: When Wo = 0 (or Wo = 1 for the multiplicative case), each resulting node-disjoint subgraph (N_k, E_k) in the collection C(W) consists of a tree in which the cost c(e) for all edges $e \in E_k$ is the same.

Remark 6: In a complete graph, the algorithm will leave at most one node isolated (with $N_k = \{k\}$ and $E_k = \varnothing$) at the conclusion of the Inner Loop for any W. In a graph that is not complete or not connected, no node that is not isolated in G will be left isolated in the collection C(W) for W sufficiently large. (To permit additional isolated nodes, a limit c_{lim} may be imposed that prevents C(W) from including any edges e such that $c(e) > c_{lim}$.)

Remark 7: When there are tied (duplicate) cost values c(e), all orderings of e(1) to e(|E|) satisfying $c(e(1)) \le c(e(2)) \le \ldots \le c(e(|E|))$ will produce the same collection of clusters C(W) in the following sense: For a given value of W, all orderings will produce the same node sets N_k defining C(W), and the sum of costs over the edge sets E_k will also be the same, though the edges within these sets may differ.

4. Fundamental relationships for accelerating the algorithm

A number of key relationships hold for the C(W) Algorithm that make it possible to accelerate its execution. We discuss the relationships here in broad outline and then incorporate them in Section 7 within a template for a computer code that applies not only to C(W) but to additional related types of cluster collections C(Y) and C(Z) whose algorithms are described in Sections 5 and 6.

4.1. Early termination of the inner loop

The Inner Loop can typically terminate far in advance of satisfying the condition s = sLast for sLast = |E|, hence making it unnecessary to examine all edges of the graph.

First note that the process of examining the edges in ascending cost order implies that once c(e(s)) > W + MinCost(k) for a given k = ko∈ K, then the inequality c(e(v)) > W + MinCost(ko) will also hold for all subsequent edges e(v) for v > s. Hence, by Case (2) of the algorithm, no nodes or edges will be adjoined to the cluster sets N_{ko} and E_{ko} for v > s. In addition, it will be unnecessary to update Wnext by reference to ko in the future.

It may further be observed that the MinCost(k) values are generated in a sequence that makes it possible to readily identify (without sorting) the values k(1), k(2), …, k(m), so that Min Cost(k(1)) ≤ MinCost(k(2)) ≤ … ≤ MinCost(k(m)). It is convenient to define m so that these values refer just to those k ∈ K such that MinCost(k) < Large. (Recall that MinCost(k) = Large implies that N_k consists of a single node k, and $E_k = \varnothing$.)

Thus if c(e(s)) > W + MinCost(k(m)), we know that none of the clusters indexed from k(1) to k(m) can take part in the creation of new clusters. Alternatively, if we start by checking whether c(e(s)) > W + MinCost(k(h)) holds for h = 1 and work forward until finding the first index k(h*) for which the inequality does not hold, then on future encounters with Case (2) it is possible to start from k(h*) rather than k(1) to begin checking whether c(e(s)) > W + MinCost(k(h)).

In consideration of these relationships, it should be kept in mind that when two clusters k′ and k″ are joined, then MinCost(k″) will no longer be referenced (since the cluster k″ will no longer exist). To see the consequences of this, suppose that k′ and k″ are interchanged, if necessary, so that MinCost(k′) ≤ MinCost(k″). Then when $N_{k''}$ is absorbed into $N_{k'}$, the following two possibilities arise:

- **i.** MinCost(k′) < Large (hence MinCost(k′) identifies the cost of an edge previously added) and MinCost(k′) will be unchanged;
- **ii.** MinCost(k′) = Large, and the new MinCost(k′) will be the value c(e(s)) of the edge e(s) currently added.

This implies that in the sequence MinCost(k(1)) ≤ MinCost(k(2)) ≤ … ≤ MinCost(k(m)), the value MinCost(k″) will drop out, and the value MinCost(k′) will either be unchanged and retain its position, or else it will change from a Large value to become the new value MinCost(k(m)) at the end of the ordered list.

However, applying this knowledge to shortcut the checks performed in Case (2) does not make it possible to save appreciable computation, since the amount of effort to perform the

checks of Case (2) is not great in any case. Instead, we can make use of the foregoing relationships in a simpler manner without having to keep track of the values k(1), k(2), …, k(m).

To accomplish this, we record the number of elements n_k in each node set N_k by initializing all $n_k = 1$, and then setting $n_{k'}$: = $n_{k'} + n_{k''}$ when $N_{k''}$ is absorbed into $N_{k'}$ in Case (3). We also record the number of times t(i) each node i is encountered as a node i′ = p(s) or i″ = q(s) by initializing t(i) = 0 for all i, and then setting t(i′): = t(i′) + 1 and t(i″): = t(i″) + 1 when the edge e(s) is examined in the prelude to Case (1)of the algorithm (and also for i′ and i″ in the Initialization). Note that t(i) is bounded by tMax(i) which is the number of nodes adjacent to i in the graph G (where tMax(i) = n – 1 if G is complete).

We are interested in determining when t(i) = tMax(i) for an isolated node. We can conveniently identify the condition of being isolated by i = L(i). In conjunction with the preceding records, this makes it possible to keep track of the number nTrack of nodes that cannot take part in any further steps of adding an edge to C(W), and hence permitting the inner loop to terminate when nTrack = n.

Specifically, by initializing nTrack = 0, the first time c(e(s)) > W + MinCost(k)occurs for a given k = k′ or k″ in Case (2), we set nTrack: = nTrack + n_k. (To identify this first occurrence, initialize FirstTime(k) = True, and then set FirstTime(k) = False at the point of setting nTrack: = nTrack + nk.) We also set nTrack: = nTrack + 1 whenever t(i) is incremented for i = i′ and i″ in the prelude to Cases (1) to (3) to yield t(i) = tMax(i) under the condition that i = L(i). By checking for nTrack = n at each point where nTrack changes its value, we can then terminate the inner loop when this condition occurs.

Having performed the foregoing operations to terminate early for W = Wo, we may take advantage of another useful relationship to terminate early for all W > Wo. In particular, let sEnd(W) equal the value of s for the final edge e(s) added to C(W) for a given W. Then for values W′ and W″ such that W″ > W′, we are assured that sEnd(W″) ≤ sEnd(W′). Consequently, we can exploit this fact by introducing a variable sEnd which is set to sEnd = s at the conclusion of Case (3), which will cause sEnd to be the index s of the final edge added in constructing the current C(W). Then it is only necessary to set sLast = sEnd after the termination of the Inner Loop, thus overriding the initialization sLast = |E| to permit the next execution of the Inner Loop to terminate earlier. We can also allow the final execution of the Inner Loop to terminate earlier by the fact that the spanning tree generated on this execution will have n − 1 edges, while all other constructions must have fewer than this number of edges.

4.2. Advanced starting for successive W values

We can also accelerate the computation of the algorithm by saving information to produce an advanced start on successive executions of the Inner Loop. The underlying relationships are as follows.

Let s2(1), s2(2), …, s2(v_2) denote the s indexes starting with s2(1) = 1 (when Wnext = Large) where the values s2(v) for v > 1 identify successive edges e(s) for which a new (smaller) value of Wnext is identified in Case (2). Also, starting with W(1) = Large, let W(1), W(2), …, W(v″) denote the corresponding values for Wnext identified at these points (hence, for $v_2 > 1$, W(1) > W(2) > … > W(v_2)). Similarly, let s3(1), s3(2), …, s3(v_3) denote the s indexes starting with

s3(1) = 1 where the values s3(v) for v > 1 identify successive edges e(s) that are added in Case (3) of the Inner Loop(to generate the current cluster collection C(W)).

After completing the Inner Loop for W = Wo (while saving this information), upon assigning W the value Wnext = W(v_2), the fact that Wnext < W(v) for v < v_2 implies that the algorithm will perform exactly the same sequence of steps until reaching s = s2(v_2), at which point the edge e(s) for s = s2(v_2), will be added to the construction (although this edge was not added on the previous execution of the inner loop).

Consequently, all edges e(s3(v)) for s3(v) < s2(v_2) will again be added to the current construction, and the values s2(v) for v < v_2 will also be unchanged. Hence, letting v* = Max(v: s3(v) < s2(v_2)) we can start the current construction by simply adding the edges e(s) for s = s3(1) to s3(v*), followed by adding the edge e(s) for s = s2(v_2) (whose index s2(v_2) therefore becomes recorded as the new index s3(v* + 1)). Then the customary Inner Loop for W > Wo can be executed starting with s initialized by setting s = s2(v_2) instead of s = 1. Subsequent executions of the Inner Loop continue to save the same information, which is used again to create an advanced start in the manner described.

By this means, we avoid examining all edges e(s) for s < s2(v_2) that were not added to the previous construction. We also avoid having to re-do the checks to determine that the remaining edges qualify to be added. Together this can amount to a considerable savings in computation.

A possibility arises to save additional computation by using more memory. Each time a new candidate for Wnext is identified, in the process of identifying the indexes s2(1), s2(2), …, s2(v_2), we can save a current copy of the arrays N_k, E_k, MinCost(k) and K used by the algorithm, avoiding the burden of excessive memory by overwriting the previous copy each time a new one is made. Then the latest copy will be available at the point where the edge e(s) for s = s2(v_2) is added on the current execution of the loop, making it possible to recover the arrays without having to regenerate them to resume the current loop.

However, it may not be possible to take advantage of a current copy of the saved arrays on every iteration of the Inner Loop (unless previous copies are not overwritten when new ones are made). After re-starting by recovering the arrays for s2(v_2), if now a new Wnext value is determined for s > s2(v_2) (referring to the v_2 of the previous execution) then we can proceed by again making a copy of the arrays for the next execution of the loop. But if it no new value of Wnext is found for s > s2(v_2), then the previous value W(v_2–1) (for s = s2(v_2–1) will be the new final Wnext value, and no copies of the arrays remain in memory for this value.

Consequently, in this latter case we resort to the construction that does not rely on the copied arrays, generating the arrays instead in the process of adding edges. Thus, on the next execution of the inner loop we will again have the copies available. Hence in this fashion we will be able to take advantage of the copied arrays at least on every second execution of the loop, if not more frequently.

As previously noted, the foregoing relationships and their implications are embodied in a format suitable for creating a computer code in Section 7, after we first describe two additional algorithmic variants that can be exploited by analogous relationships.

http://dx.doi.org/10.5772/intechopen.76406

5. Algorithm C(Y): a node-based algorithmic variant

It is possible to formulate a node-based variant of the C(W) algorithm which follows a closely related format and is supported by a similar rationale.

In the node-based approach, we replace the parameter W by a parameter Y which is linked to costs associated with nodes in N_k rather than to costs associated with edges in E_k. (More precisely, the costs associated with nodes are also derived from edges—i.e., the edges that meet these nodes—though these edges are different from those referenced in the C(W) algorithm.)

Accompanying this parameter change, we replace the value MinCost(k) associated with the sets indexed by $k \in K$ with a value MinCostB(i) associated with the nodes $i \in N$, and more particularly, we replace MinCost(k′) for k′ = L(i′) by MinCostB(i′), and replace MinCost(k″) for k″ = L(i″) by MinCostB(i″)).

This replacement changes the updating rule when $N_{k''}$ is absorbed into $N_{k'}$ in Case (3). Specifically, the values MinCostB(i′) and MinCostB(i″)) are updated by setting MinCostB(i): = Min(MinCostB(i),c(e(s)) for i = i′ and i″, in contrast to the update involving MinCost(k′) and MinCost(k″) (which setsMinCost(k′): = Min(MinCost(k′),MinCost(k″), c(e(s))).

The reason for these changes is as follows. In the node-based version, to permit the edge e(s) = (i′, i″) (= (p(s),q(s))) to be added and hence to join the subgraphs ($N_{k'}$, $E_{k'}$) and ($N_{k''}$, $E_{k''}$), we require that c(e(s)) ≤ Y + MinCostB(i) for both i = i′ and i″. Hence we require c(e(s)) ≤ Y + MinCostB0, for MinCostB0 = Min(MinCostB(i′), MinCostB(i″)). On the other hand, if c(e(s)) > Y + MinCostB0, we are prevented from adding edge e(s), and by the preceding relationships this causes the first part of Case (2) to retain exactly the same form as in the C(W) algorithm.

To update MinCostB(i′) and MinCostB(i″) in Case (3), we must account for the fact that each of these two values is affected only by the cost of the edge e(s), and hence will either retain its present value or become equal to c(e(s)), according to which is smaller. (It may be noted that once node i for i = i′ or i″ has been assigned an edge cost c(e(s)), then MinCostB(i) will not change thereafter, since any edge e(s) that is added later to meet node I will have a cost no less than that of the earlier edge.)

Based on these observations, we can state the form of the C(Y) algorithm as follows.

C(Y) Algorithm (Node-Based Version)

Inputs: The graph G(N, E), cost vector c(e), $e \in E$, initial Yo value for Y.

Edges are ordered so that the costs satisfy $c(e(1)) \leq c(e(2)) \leq \ldots \leq c(e(|E|))$.

Set Y = Yoand sLast = |E|.

Begin Outer Loop

While Y < Large

Initialization(A). Set Ynext = Large, K = {1, …, n}, and for each k ∈ K let L(k) = k,

Nk = {k}, $E_k = \emptyset$, and MinCostB(k) = Large.

Initialization(B). Let i′ = p(1) and i″ = q(1) and select e(1) (= (i′, i″)) by identifying k′ = L(i′) and k″ = L(i″) and absorbing $N_{k''}$ into $N_{k'}$ to create the cluster $N_{k'}$: = $N_{k'} \cup N_{k''}$= {i′, i″} with edge set $E_{k'}$ = e(1). Set MinCostB(k′) = c(e(1)) and set K: = K \ {k″}. Finally, initialize the edge index s by setting s = 1.

Begin Inner Loop

While s < sLast |

Set s: = s + 1 and identify edge e(s) = (p(s), q(s)). Let i′ = p(s), i″ = q(s) and let k′ = L(i′) and k″ = L(i″).

Case (1): If k′ = k″ (i′ and i″ belong to the same cluster), then continue to the next iteration of the Inner Loop.

Case (2): If c(e(s)) > Y + MinCostB0, for MinCostB0 = Min(MinCostB(i′), MinCostB(i″)), then compute Ynext = Min(Ynext, c(e(s)) – MinCostB0) and continue to the next iteration of the Inner Loop.

Case (3) (If (1) and (2) do not apply): Absorb $N_{k''}$ into $N_{k'}$ to create $N_{k'}$: = $N_{k'} \cup N_{k''}$ with its associated edge set $E_{k''}$: = $E_{k'} \cup E_{k''} \cup \{e(s)\}$. Set L(i) = k′ for all i∈ $N_{k''}$, and setMinCostB(i): = Min(MinCostB(i), c(e(s)) for i = i′ and i″. Finally, set K: = K \ {k″}.

Endwhile.

Y = Ynext.

Endwhile.

End of C(Y) Algorithm

The Remarks concerning the C(W) algorithm in Section 3 can be applied as well to the C(Y) algorithm.

Now we show how to join the C(W) and C(Y) algorithms.

6. Algorithm C(Z): a combination of C(W) and C(Y)

Each of the C(W) and C(Y) algorithms has features lacking in the other. However, the C(Y) algorithm has a potential deficiency, which resides in the fact that it is subject to "drift"—a phenomenon where the costs of edges in an edge set E_k can grow along a chain, where each new edge added to E_k has a higher cost than the previous one. Such an eventuality can arise because the cost of an edge in a chain is limited only by the cost of the previous edge.

http://dx.doi.org/10.5772/intechopen.76406

It is possible to combat drift and also take advantage of the different features of the C(W) and C(Y) algorithms by joining these algorithms to create an algorithm C(Z) that incorporates the MinCost evaluation criteria of both C(W) and C(Y) simultaneously.

Let α be a nonnegative weight applied to the edge selection criterion of C(W) and let $\beta = 1 - \alpha$ be a nonnegative weight applied to the edge selection criterion of C(Y). We construct Algorithm C(Z) so that it will be the same as C(W) if $\alpha = 1$ and will be the same as C(Y) if $\alpha = 0$ ($\beta = 1$).

For notational convenience, we refer to the value MinCost(k) of the C(W) algorithm as MinCostA(k). Then the MinCost evaluation criterion of C(Z) is given by.

$$\text{MinCostC(i)} = \alpha \cdot \text{MinCostA(k)} + \beta \cdot \text{MinCostB(i)}, \text{ for } k = L(i).$$

To apply this criterion, we create values MinCostC(i′) and MinCostC(i″) for nodes i′ = p(s) and i″ = q(s), and for k′ = L(i′) and k″ = L(i″), given by

$$\text{MinCostC}(i') = \alpha \cdot \text{MinCostA}(k') + \beta \cdot \text{MinCostB}(i')$$

$$\text{MinCostC}(i'') = \alpha \cdot \text{MinCostA}(k'') + \beta \cdot \text{MinCostB}(i'')$$

Associated with the foregoing values, we define

$$\text{MinCostC0} = \text{Min}(\text{MinCostC}(i'), \text{MinCostC}(i''))$$

We state the C(Z) algorithm by reference to these definitions.

C(Z) Algorithm.

Inputs: The graph G(N, E), cost vector c(e), e ∈ E, initial Zo value for Z.

Edges are ordered so that the costs satisfy $c(e(1)) \le c(e(2)) \le \ldots \le c(e(|E|))$.

Set Z = Zo and sLast = |E|

Begin Outer Loop

While Z < Large

Initialization(A). Set Znext = Large, K = N (= {1, …, n}), and for each k ∈ K let.

$L(k) = k$, $N_k = \{k\}$, $E_k = \varnothing$, and MinCostA(k) = MinCostB(k) = MinCostC(k) = Large.

Initialization(B). Let i′ = p(1) and i″ = q(1) and select e(1) (= (i′, i″)).Set k′ = L(i′) and k″ = L(i″) and absorb $N_{k''}$ into $N_{k'}$ to create the cluster $N_{k'} := N_{k'} \cup N_{k''} = \{i', i''\}$ with edge set $E_{k'}$ = e(1). Set MinCostA(k′) = c(e(1)) and MinCostB(i) = c(e(1)) for i = i′ and i″. Set.

K: = K \ {k″} and s = 1.

Begin Inner Loop

While s < sLast

Set s: = s + 1 and identify edge e(s) = (p(s), q(s)). Let i′ = p(s), i″ = q(s) and let k′ = L(i′) and k″ = L(i″).

Case (1): If k′ = k″, then continue to the next iteration of the Inner Loop.

Case (2): If c(e(s)) > Z + MinCostC0 (for MinCostC0 determined by the preceding definitions), then compute Znext = Min(Znext, c(e(s)) – MinCostC0) and continue to the next iteration of the Inner Loop.

Case (3) (If (1) and (2) do not apply): Absorb $N_{k''}$ into $N_{k'}$ to create the larger cluster $N_{k'}$: = $N_{k'} \cup N_{k''}$ with its associated edge set.

$E_{k''}$: = $E_{k'} \cup E_{k''} \cup \{e(s)\}$.

Correspondingly, update L(i) by setting L(i) = k′ for all i∈ Nk″, and set MinCostA(k′): = Min(MinCostA(k′),MinCostA(k″), c(e(s)); MinCostB(i): = Min(MinCostB(i),c(e(s)) for i = i′ and i″ (thus yielding MinCostC(i) by the definitions above).Finally, set K: = K \ {k″}.

Endwhile

Z = Znext

Endwhile

End of C(Z) Algorithm

The next section shows how to carry out accelerated updates in the context of the preceding algorithm.

7. Implementing accelerated updates for the C(Z) algorithm

We build on the relationships identified in Section 4 to apply them to the more general C(Z) algorithm. By the implications described earlier, our general updates also apply directly to the C(W) and C(Y) algorithms by respectively replacing Z with W and Y (hence Znext with Wnext and Ynext) and replacing MinCostC(·) by MinCost(·) and MinCostB(·).

7.1. Early termination for the C(Z) inner loop

Early termination for the Inner Loop of C(Z) is effected by creating an Initialization(C) immediately following Initialization(B) (hence inheriting the assignments of Initialization(B)) and modifying the Inner Loop as follows. We apply these changes for Z = Zo and thereafter take advantage of setting sLast = sEnd as indicated below, in order to allow for the advanced updating of C(Z) for successive Z values.

Initialization(C) for Z = Zo:

Set $n_k = 1$, $k \in K \setminus \{k'\}$, and $n_{k'} = 2$ (hence $n_k = |N_k|$, $k \in K$); set t(i) = 0 for $i \in N \setminus \{i', i''\}$and set t(i′) = t(i″) = 1 (hence t(i) identifies the number of edges e(s) = (i,j) for all s currently examined in the Inner Loop (and its initialization). Finally, set nTrack = 0 and set FirstTime(k) = True, $k \in K$ (to identify the sets that have not been prevented from having an edge added to them).

Modification of Inner Loop for Z = Zo:

In the prelude to Case (1): Execute the following for i = i′ and i″: If i = L(i) then set t(i): = t(i) + 1 and if (now) t(i) = tMax(i) then set nTrack: = nTrack +1 and if (now) nTrack = n terminate the Inner Loop. (tMax(i) denotes the number of nodes adjacent to i in the graph G.)

Re-organize Case (2) to become as follows: Execute the following for k = k′ and k = k″: If FirstTime(k) = True, then if c(e(s)) > Z + MinCostC(k): set FirstTime(k) = False, set nTrack: = nTrack + $n_{k'}$, compute Znext = Min(Znext, c(e(s)) – MinCostC(k) and if (now) nTrack = n, terminate the Inner Loop.After performing the preceding for both k′ and k″: If FirstTime(k) = False for k = k′ or k = k″, then continue to the next iteration of the Inner Loop. (Note: Case (2) could also be moved to precede Case (1).)

In Case (3) (when $N_{k''}$ is absorbed into $N_{k'}$): Set $n_{k'}$: = $n_{k'} + n_{k''}$.

Modifications for all Z values:

Set sEnd = s at the end of Case (3) and set sLast = sEnd immediately after the conclusion of the Inner Loop (following Z = Znext).

For early termination of the last execution of the Inner Loop:

Set $n_e = 1$ in Initialization(B). Then at the end of Case (3) set $n_e = n_e + 1$ and if $n_e = n - 1$ terminate the Inner Loop. (The Outer Loop will automatically terminate as well, by the condition Z = Large.)

Because of the special modifications for Z = Zo that do not apply for Z > Zo, it is convenient to insert the portion of the algorithm for Z = Zo at the very beginning of the C(Z) Algorithm, before the Outer Loop.

7.2. Advanced updating for successive Z values

We draw on the relationships of Section 4.2 to create the instructions for updating C(Z) to reduce the amount of computation required on successive iterations of the Inner Loop.

We adopt the following notation of Section 4.2, re-expressed in terms of the C(Z) algorithm: s2(1), s2(2), …, s2(v_2)identifies the successive s indexes that occur each time a new (smaller) value of Znext is identified in Case (2) of the Inner Loop, beginning with the initialized value s2(1) = 1. Similarly, s3(1), s3(2), …, s3(v_3)identifies the s indexes of successive edges e(s) that are added in Case (3), beginning with the initialized value s3(1) = 1. (In the re-organized Case (2) for Z = Zo in Section 7.1, the value Znext may be reduced twice for a given edge e(s) and we only consider the last (smaller) Znext value produced for e(s).) The sequence Z(1), Z(2), …, Z(v_2) with the initialization Z(1) = Large, denotes the corresponding candidate values for Znext generated in Case (2). We therefore have Z(1) > Z(2) > … > Z(v_2), where the final Znext is given by Znext = Z(v_2).

Assume the algorithm for Z = Zo has been inserted before the start of the Outer Loop, as indicated in Section 7.1.

Modification of Initialization(B) for Z = Zo only:

Add $v_2 = v_3 = 1$; s2(1) = s3(1) = 1 and Z(1) = Large.

Modification of the Inner Loop for all Z:

In Case (2) when Znext is updated (reduced): set $v_2 = v_2 + 1$; $s2(v_2) = s$, and $Z(v_2) =$ Znext.

In Case (3) upon adding e(s): set $v_3 = v_3 + 1$; $s3(v_3) = s$.

Modification After the Inner Loop for all Z:

Following the instructions Z = Znext, and sLast = sEnd:If $v_2 = 1$, terminate.

Modification Before the Inner Loop for Z > Zo:

Insert a Preliminary Loop before the Inner Loop for Z > Zo as follows:

(After Initialization(A) and Initialization(B))

Preliminary Loop

Set v* = Max(v: $s3(v) < s2(v_2)$); $v_3 = v^* + 1$; $s3(v_3) = s2(v_2)$; v_2: = v_2–1.

v = 1

While $v < v_3$

v: = v + 1

s = s3(v)

Insert the prelude to Case (1) followed by Case (3) (excluding the modification above that adds $v_3 = v_3 + 1$; $s3(v_3) = s$)

EndWhile[2]

Set $s = s3(v_3)$

7.3. Modifications involving additional memory

Added Modification to Initialization for Z = Zo:

Copy = False (where Copy indicates whether a copy is made of the arrays indicated in the modification below).

Added Modification of the Inner Loop for all Z:

Each time Case (2) yields a new value of Znext (after checking for both k′ and k″ for Z = Zo) record a copy of ne, the set K and arrays $N_{k'}$,$E_{k'}$, MinCostA(k), k ∈K and the arrays L(i),

[2]As before, we adopt the convention whereby the loop is bypassed if $v_3 = 1$ but will be executed on the iteration where v: = v + 1 results in $v = v_3$.

MinCostB(i), MinCostC(i), i∈ N (writing over any previous copy). Let Copy = True if such a copy is made. (As remarked earlier, it is not necessary to keep track of the E_k array.)

Added Modification for the Preliminary Loop

If Copy = False, execute the Preliminary Loop. Otherwise, if Copy = True, instead of executing the Preliminary Loop read the copies of the saved arrays into the active form of these arrays, followed by setting s = s2(v_2) and Copy = False. (The Inner Loop immediately follows this modification.)

The complete C(Z) Algorithm that incorporates all of these changes is shown in the Appendix for convenience.

8. Conclusions

The new classes of tree-based clustering algorithms represented by C(W), C(Y) and in the most general case by C(Z), afford the possibility to generate clusters with a range of different characteristics as the parameters W, Y and Z are varied. The fact that the key parameter can be varied adaptively to generate all cluster collections in its class without duplication invites empirical studies to identify parameter ranges that are effective for particular types of clustering applications.

The C(Z) Algorithm can be made still more general by changing the implicit definition of MinCostC(i) for i = i′ and i″ by defining MinCostA(k), for k = k′ and k″, to be any convex combination of the costs of edges in the set $E(N_k) = \{e = (i,j) \in N_k\} \subset E$ (hence $(N_k, E(N_k))$ is the subgraph of G spanned by the nodes of N_k) and defining MinCostB(i) to be any convex combination of the edges of $E(N_k)$ that are adjacent to node i in G. However, this requires updating these MinCost values in a more complex way than in the current form of Case (3).

Future work to exploit the properties of these algorithms can include an investigation of the choice of the parameter α (and hence $\beta = 1 - \alpha$) in Algorithm C(Z) to similarly determine ranges that are effective for particular types of clustering applications.

Acknowledgements

This research was supported in part by the Key Laboratory of International Education Cooperation of Guangdong University of Technology and by the Fundamental Research Funds for the Central Universities (No3102017zy059).

A. Appendix

A.1. Algorithm C(Z) with accelerated updates

We identify the form of the C(Z) algorithm that results by incorporating all of the accelerated updates of Section 7. As before, the sets E_k do not need to be maintained unless they are of

interest for experimental purposes. The case where added memory is used is identified in Case (2) so that this memory need not be used if desired. (In that case, the variable Copy will always remain False, as in Initialization(B).)

C(Z) Algorithm with Accelerated Updates

Inputs: The graph G(N, E), cost vector c(e), e ∈ E, initial Zo value for Z.

Edges are ordered so that the costs satisfy c(e(1)) ≤ c(e(2)) ≤ … ≤ c(e(|E|)).

Set Z = Zo and sLast = |E|. Copy = False.

Execute Routine for Z = Zo (Given Below)

Begin Outer Loop for Z > Zo

While Z < Large

Initialization(A). Set Znext = Large, K = N (= {1, …, n}), and for each k ∈ K let.

L(k) = k, N_k = {k}, E_k = ∅, and MinCostA(k) = MinCostB(k) = MinCostC(k) = Large.

Initialization(B). Let i′ = p(1) and i″ = q(1) and select e(1) (= (i′, i″)).Set k′ = L(i′) and k″ = L(i″) and absorb $N_{k''}$ into $N_{k'}$ to create the cluster $N_{k'}$: = $N_{k'} \cup N_{k''}$= {i′, i″} with edge set $E_{k'}$ = e(1). Set MinCostA(k′) = c(e(1)) and MinCostB(i) = c(e(1)) for i = i′ and i″ and set K: = K \ {k″}. Then set n_e = 1 and s = 1.

If (Copy = False) then.

Execute Preliminary Loop

Set v* = Max(v: s3(v) < s2(v_2)); v_3 = v* + 1; s3(v_3) = s2(v_2); v_2: = v_2–1.

v = 1

While v < v_3

v: = v + 1

s = s3(v)

Identify edge e(s) = (p(s), q(s)). Let i′ = p(s), i″ = q(s) and let k′ = L(i′) and k″ = L(i″).

Execute Case (3): Absorb $N_{k''}$ into $N_{k'}$ by setting $N_{k'}$: = $N_{k'} \cup N_{k''}$ with its associated edge set $E_{k''}$: = $E_{k'} \cup E_{k''} \cup$ {e(s)}. Set L(i) = k′ for all i∈ $N_{k''}$, and set MinCostA(k′): = Min(MinCostA(k′),MinCostA(k″), c(e(s)); MinCostB(i): = Min(MinCostB(i),c(e(s)) for i = i′ and i″ (thus yielding MinCostC(i) by the definitions of Section 7). Set K: = K \ {k″} and sEnd = s. Set ne: = n_e + 1 and if n_e = n – 1 terminate the C(Z) Algorithm.

EndWhile

Set s = s3(v_3)

Else

Read the latest copy of n_e and the arrays saved in Case (2) of the Inner Loop into the active form of these arrays. Set s = s2(v_2) and Copy = False.

Endif

Begin Inner Loop

While s < sLast

Set s: = s + 1 and identify edge e(s) = (p(s), q(s)). Let i′ = p(s), i″ = q(s) and let k′ = L(i′) and k″ = L(i″).

Case (1): If k′ = k″, then continue to the next iteration of the Inner Loop.

Case (2): If c(e(s)) > Z + MinCostC0, then compute Znext = Min(Znext, c(e(s)) – MinCostC0), set $v_2 = v_2 + 1$, $s2(v_2) = s$, $Z(v_2) = Znext$.

For added memory case:

Set Copy = True and record a copy of ne, the set K and arrays N_k, E_k, MinCostA(k), k ∈K and the arrays L(i), MinCostB(i), MinCostC(i), i∈ N (writing over any previous copy).

Continue to the next iteration of the Inner Loop.

Case (3): Absorb $N_{k''}$ into $N_{k'}$ by setting $N_{k'}$: = $N_{k'} \cup N_{k''}$ with its associated edge set $E_{k''}$: = $E_{k'} \cup E_{k''} \cup \{e(s)\}$. Set L(i) = k′ for all i∈$N_{k''}$, and set MinCostA(k′):=Min(MinCostA(k′), MinCostA(k″), c(e(s)); MinCostB(i): = Min(MinCostB(i), c(e(s)) for i = i′ and i″ (thus yielding MinCostC(i) by the definitions of Section 7). Set K: = K \ {k″} and sEnd = s.Set v_3:= $v_3 + 1$, $s3(v_3) = s$, ne:= $n_e + 1$ and if $n_e = n - 1$ terminate the Inner Loop.

Endwhile

Z = Znext

sLast = sEnd

If $v_2 = 1$ terminate the C(Z) Algorithm

Endwhile

End of C(Z) Algorithm

Routine for Z = Zo

Initialization(A). Set Znext = Large, K = N (= {1, …, n}), and for each k ∈ K let.

L(k) = k, N_k = {k}, E_k = ∅, and MinCostA(k) = MinCostB(k) = MinCostC(k) = Large.

Initialization(B). Let i′ = p(1) and i″ = q(1) and select e(1) (= (i′, i″)).Set k′ = L(i′) and.

k″ = L(i″) and absorb $N_{k''}$ into $N_{k'}$ to create the cluster $N_{k'}$: = $N_{k'} \cup N_{k''}$= {i′, i″} with edge set $E_{k'}$ = e(1). Set MinCostA(k′) = c(e(1)) and MinCostB(i) = c(e(1)) for i = i′ and i″ and set K: = K \ {k″}. Set $v_2 = v_3 = 1$; s2(1) = s3(1) = 1 and Z(1) = Large. Set $n_e = 1$ and s = 1.

Initialization(C): Set nk = 1, k ∈ K \ {k′}, and $n_{k'} = 2$; set t(i) = 0 for i∈ N \{i′, i″} and set t(i′) = t(i″) = 1. Set nTrack = 0 and set FirstTime(k) = True, k ∈ K.

Begin Inner Loop

While s < sLast

Set s: = s + 1 and identify edge e(s) = (p(s), q(s)). Let i′ = p(s), i″ = q(s) and let k′ = L(i′) and k″ = L(i″). For i = i′ and i″: If i = L(i) then set t(i): = t(i) + 1 and if (now) t(i) = tMax(i) then set nTrack: = nTrack +1 and if (now) nTrack = n terminate the Inner Loop.

Case (1): If k′ = k″ (i′ and i″ belong to the same cluster), then continue to the next iteration of the Inner Loop.

Case (2): Execute the following for k = k′ and k = k″: If FirstTime(k) = True, then if c(e(s)) > Z + MinCostC(k): set FirstTime(k) = False, set nTrack: = nTrack + $n_{k'}$, compute Znext = Min(Znext, c(e(s)) – MinCostC(k) and if (now) nTrack = n, terminate the Inner Loop. After performing the preceding for both k′ and k″: If FirstTime(k) = False for k = k′ or k = k″ execute the following (and otherwise proceed to Case (3)): set $v_2 = v_2 + 1$, $s2(v_2) = s$, $Z(v_2) = Znext$.

For added memory case:

Set Copy = True and record a copy of ne, the set K and arrays $N_{k'}$, $E_{k'}$, MinCostA(k), k ∈K and the arrays L(i), MinCostB(i), MinCostC(i), i∈ N (writing over any previous copy).

Continue to the next iteration of the Inner Loop.

Case (3): Absorb $N_{k''}$ into $N_{k'}$ to create the larger cluster $N_{k'}$: = $N_{k'} \cup N_{k''}$ with its associated edge set $E_{k''}$: = $E_{k'} \cup E_{k''} \cup \{e(s)\}$. Correspondingly, update L(i) by setting L(i) = k′ for all i∈ $N_{k''}$, and set MinCostA(k′): = Min(MinCostA(k′),MinCostA(k″), c(e(s)); MinCostB(i): = Min(MinCostB(i),c(e(s)) for i = i′ and i″ (thus yielding MinCostC(i) by the definitions of Section 7). Set K: = K \ {k″}, $n_{k'}$: = $n_{k'} + n_{k''}$, and sEnd = s. Set v_3:= $v_3 + 1$, $s3(v_3) = s$, n_e:= $n_e + 1$ and if $n_e = n - 1$ terminate the Inner Loop.

Endwhile

Z = Znext

sLast = sEnd

If $v_2 = 1$ then

Terminate the C(Z) Algorithm

Else.

$v^* = Max(v: s3(v) < s2(v_2))$; $v_3 = v^* + 1$; $s3(v_3) = s2(v_2)$; v_2: = $v_2 - 1$.

Endif

End of Routine for Z = Zo

Author details

Fred Glover[1]* and Yang Wang[2]

*Address all correspondence to: fredwglover@yahoo.com

1 School of Engineering and Science, University of Colorado, Boulder, CO, USA

2 School of Management, Northwestern Polytechnical University, Xian, China

References

[1] Anderberg MR. Cluster analysis for applications. In: Monographs and Textbooks on Probability and Mathematical Statistics. New York: Academic Press, Inc.; 1973

[2] Stefik MJ. Machine learning: An artificial intelligence approach. R.S. Michalski, J.G. Carbonell and T.M. Mitchell, (Tioga, Palo Alto, CA); 572 pages, $39.50. Artificial Intelligence. 1985;**25**(2):236-238

[3] Michalski RS, Carbonell JG, Learning TMMM. An artificial intelligence approach. Understanding the Nature of Learning. 1983;**2**:3-26

[4] Rinzivillo S, Pedreschi D, Nanni M, Giannotti F, Andrienko N, Andrienko G. Visually driven analysis of movement data by progressive clustering. Information Visualization. 2008;**7**(3):225-239

[5] Chen H, Chiang RHL, Storey VC. Business intelligence and analytics: From big data to big impact. MIS Quarterly. 2012;**36**(4):1165-1188

[6] Xu Y, Olman V, Xu D. Minimum spanning trees for gene expression data clustering. Genome Informatics. 2001;**12**:24-33

[7] Xu Y, Olman V, Xu D. Clustering gene expression data using a graph-theoretic approach: An application of minimum spanning trees. Bioinformatics. 2002;**18**(4):536-545

[8] Jana PK, Naik A, editors. An efficient minimum spanning tree based clustering algorithm. In: Proceedings of International Conference on Methods and MODELS in Computer Science; 2009

[9] Grygorash O, Zhou Y, Jorgensen Z, editors. Minimum spanning tree based clustering algorithms. In: IEEE International Conference on TOOLS with Artificial Intelligence; 2008

[10] Wang ZM, Soh YC, Song Q, Kang S. Adaptive spatial information-theoretic clustering for image segmentation. Pattern Recognition. 2009;**42**(9):2029-2044

[11] Gower JC, Ross GJS. Minimum spanning trees and single linkage cluster analysis. Journal of the Royal Statistical Society. 1969;**18**(1):54-64

[12] Laszlo M, Mukherjee S. Minimum spanning tree partitioning algorithm for microaggregation. IEEE Transactions on Knowledge and Data Engineering. 2005;**17**(7):902-911

[13] Asano T, Bhattacharya B, Keil M, Yao F, editors. Clustering algorithms based on minimum and maximum spanning trees. In: Symposium on Computational Geometry; 1988

[14] Päivinen N. Clustering with a minimum spanning tree of scale-free-like structure. Pattern Recognition Letters. 2005;**26**(7):921-930

[15] Wang X, Wang X, Wilkes DM. A divide-and-conquer approach for minimum spanning tree-based clustering. IEEE Transactions on Knowledge and Data Engineering. 2009;**21**(7):945-958

Chapter 9

Robust Spectral Clustering via Sparse Representation

Xiaodong Feng

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.76586

Abstract

Clustering high-dimensional data has been a challenging problem in data mining and machining learning. Spectral clustering via sparse representation has been proposed for clustering high-dimensional data. A critical step in spectral clustering is to effectively construct a weight matrix by assessing the proximity between each pair of objects. While sparse representation proves its effectiveness for compressing high-dimensional signals, existing spectral clustering algorithms based on sparse representation use those sparse coefficients directly. We believe that the similarity measure exploiting more global information from the coefficient vectors will provide more truthful similarity among data objects. The intuition is that the sparse coefficient vectors corresponding to two similar objects are similar and those of two dissimilar objects are also dissimilar. In particular, we propose two approaches of weight matrix construction according to the similarity of the sparse coefficient vectors. Experimental results on several real-world high-dimensional data sets demonstrate that spectral clustering based on the proposed similarity matrices outperforms existing spectral clustering algorithms via sparse representation.

Keywords: spectral clustering, high-dimensional data, weight matrix, sparse representation

1. Introduction

As an important task in data mining cluster analysis aims at partitioning data objects into several meaningful subsets, called clusters, such that data objects are similar to those in the same cluster and dissimilar to those in different clusters. With advances in database technology and real-world need of informed decisions, data sets to be analyzed are getting bigger—with many more records and variables. Examples of high-dimensional data sets include document data [1], user ratings data [2], multimedia data [3], financial time series data [4], gene expression data [5] and so on. Due to the "curse of dimensionality" [6], clustering

high-dimensional data has been a challenging task and therefore, attracts much research in data mining and related research domains [7].

Many of the existing high-dimensional clustering approaches can be categorized into the following three types: dimension reduction [8], subspace clustering [9] and spectral clustering [10–12]. The first two types transform the original feature space to a lower-dimensional space and then apply an ordinary clustering algorithm (such as K-means). The focus is on how to extract more important features of data objects and avoid noises from the less important dimensions. Spectral clustering is based on the spectral graph model, which searches for clusters in the full feature space and is equivalent to graph min-cut problem based on a graph structure constructed from the original objects in vector space [12]. Spectral clustering is also considered superior to traditional clustering algorithms due to its deterministic and polynomial time solution. All these characteristics make spectral clustering suitable for high-dimensional data clustering [10].

The effectiveness of spectral clustering depends on the weights between each pair of data objects. Thus, it is vital to construct a weight matrix that faithfully reflects the similarity information among objects. Traditional simple weight construction, such as ε-ball neighborhood, k-nearest neighbors, inverse Euclidean distance [13, 14] and Gaussian radial basis function (RBF) [12], is based on the Euclidean distance in the original space, thus not suitable for high-dimensional data.

In this chapter, we focus on effective weight construction for spectral clustering, based on sparse representation theory. Sparse representation aims for representing each object approximately by a sparse linear combination of other objects, which comes from the theory of compressed sensing [15]. Coefficients of the sparse linear combination represent the closeness of each object to other objects. Traditional spectral clustering methods based on sparse representation [16] use these sparse coefficients directly to build the weight matrix, thus only local information is utilized. However, exploiting more global information of the whole coefficient vectors promises better performance, followed by an assumption that the sparse representation vectors corresponding to two similar objects should be similar, since they can be reconstructed in a similar fashion using other data objects. If two objects are contributing in a similar manner to the reconstruction of all other objects in the same data set, they are considered similar.

Therefore, this chapter presents a spectral clustering approach of high-dimensional data exploiting global information from sparse representation solution. More specifically, using sparse representation, we firstly convert each high-dimensional data object into a vector of sparse coefficients. Then, the proximity of two data objects is assessed according to the similarity between their sparse coefficient vectors. This construction considers the complete information of the solution coefficient vectors of two objects to analyze the similarity between these two objects rather than directly using a particular single sparse coefficient, which only considers local information. In particular, we propose two different weight matrix construction approaches: one of which is based on consistent sign set (CSS) and the other is based on the cosine similarity (COS) between the two vectors. Extensive experimental results on several image data sets show that similarly exploiting the global information from the solutions of sparse representation works better than using local information of the solutions under a variety of clustering performance metrics.

2. Related work

2.1. Techniques for high-dimensional data

There are many techniques for dealing with high-dimensional signals (or data), popular of which include non-negative matrix factorization (NMF), manifold learning, compressed sensing and some combinations between them.

Non-negative matrix factorization (NMF) is a powerful dimensionality reduction technique and has been widely applied to image processing and pattern recognition applications [17], by approximating a non-negative matrix X by the product of two non-negative low-rank factor matrices W and H. It has attracted much attention since it was first proposed by Paatero and Tapper [18] and has already been proven to be equivalent in terms of optimization process with K-means and spectral clustering under some constraints [19]. The research about NMF can be generally categorized into the following groups. The first group is focused on the distance measures between the original matrix and the approximate matrix, including Kullback–Leibler divergence (KLNMF) [17], Euclidean distance (EucNMF) [20], earth mover's distance metric [21] and Manhattan distance-based NMF (MahNMF) [22]. Besides, there are researches about how to solve the optimization of NMF efficiently and the scalability of NMF algorithms for large-scale data sets, for example, fast Netwon-type methods (FNMA) [23], online NMF with robust stochastic approximation (OR-NMF) [24] and large-scale graph-regularized NMF [25]. Moreover, how to improve the performance of NMF using some constrains or exploiting more information of data is also popular, such as sparseness constrained NMF (NMFsc) [26], convex model for NMF using $l_{1,\infty}$ regularization [27], discriminant NMF (DNMF) [28], graph-regularized NMF (GNMF) [29], manifold regularized discriminative NMF (MD-NMF) [30] and constrained NMF (CNMF) [31] incorporating the label information.

Manifold learning is another theory to process high-dimensional data, assuming that the data distribution is supported on a low-dimensional sub-manifold [32]. The key idea of manifold learning is that the locality structure of high-dimensional data should be preserved in low-dimensional space after dimension reduction, which is exploited as a regularization term [33–35] or constraint [36, 37] to be added to the original problem. It has been widely used to machine learning and computer vision, such as image classification [38], semi-supervised multiview distance metric learning [39], human action recognition [40], complex object correspondence construction [41] and so on.

Besides the abovementioned two approaches for high-dimensional data, in recent years, sparse representation coming from compressed sensing has also attracted a great deal of attention and proves to be an extremely powerful tool for acquiring, representing and compressing high-dimensional data. The following section will briefly review of sparse representation.

2.2. Brief review of sparse representation

Given a sufficient high-dimensional training data set, $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathrm{R}^{\mathrm{m} \times \mathrm{n}}$, where $\boldsymbol{x}_i = [\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \ldots, \boldsymbol{x}_{im}]\mathrm{T} \in \mathrm{R}^{\mathrm{m}}$ is the column vector of the i-th object. Research on manifold learning [32] has

proved that any new test data y lie on a lower dimensional manifold, which can be approximately represented by a linear combination of the training objects:

$$\mathbf{y} = \alpha_1 x_1 + \cdots + \alpha_i x_i + \cdots + \alpha_n x_n = \mathbf{X}\alpha \in \mathbf{R}^m. \tag{1}$$

Obviously, if m ≫ n, Eq. (1) is overdetermined, and α can usually be found as its unique solution. Typically, the number of attributes is much less than that of training objects (i.e. m ≪ n) and Eq. (1) is undetermined, so its solution is not unique.

However, if we add the constraint that the best solution of Eq. (1) should be as sparse as possible, which means that the number of non-zero elements is minimized, the solution becomes unique. Such a sparse representation can be obtained by solving the optimization problem:

$$\alpha^* = \arg\min_{\alpha} \ \|\alpha\|_0 \ \ subject \ \ to \ \ \mathbf{y} = \mathbf{X}\alpha, \tag{2}$$

where $||.||_0$ denotes the l_0-norm of a vector, counting the number of non-zero entries in the vector. Donoho [42] proves that if matrix **X** satisfies restricted isometry property (RIP) [43], Eq. (2) has a unique solution of $\boldsymbol{\alpha}$.

However, it is NP-hard to find the sparsest solution of an underdetermined equation: that is, there is no known approach to find the sparsest solution that is significantly more efficient than exhausting all subsets of the entries for α. Researchers in emerging theory of compressed sensing [44] reveal that the non-convex optimization in (2) is equal to the following convex l_1 optimization problem if the solution α is sparse enough:

$$\alpha^* = \arg\min_{\alpha} \ \|\alpha\|_1 \ \ subject \ \ to \ \ \mathbf{y} = \mathbf{X}\alpha, \tag{3}$$

where $||.||_1$ denotes the l_1-norm of a vector, summing the absolute value of each entry in the vector. This problem can be solved in polynomial time by standard linear programming methods [45].

Since the real data contains noise, it may not be possible to express the test sample exactly as a sparse representation of the training data. The sparse solution α can still be approximately obtained by solving the following stable l_1 optimization problem:

$$\alpha^* = \arg\min_{\alpha} \ \|\alpha\|_1 \ \ subject \ \ to \ \ \|\mathbf{y} - \mathbf{X}\alpha\|_2 \le \varepsilon, \tag{4}$$

where ε is the maximum residual error; $||.||_2$ denotes the l_2-norm of a vector.

In many situations, we do not know the noise level ε beforehand. Then we can use the Lasso (least absolute shrinkage and selection operator) [46] optimization algorithm to recover the sparse solution from the following l_1 optimization:

$$\alpha^* = \arg\min_{\alpha} \ \lambda\|\alpha\|_1 + \|\mathbf{y} - \mathbf{X}\alpha\|_2, \tag{5}$$

where λ is a scalar regularization parameter of the Lasso penalty, which directly determines how sparse α will be and balances the trade-off between reconstruction error and sparsity.

In addition to Lasso, other sparse learning models are also developed. It will be the elastic net model [47] if the l_2-norm of α is also added to Eq. (5) as another penalty term. Double shrinking algorithm (DSA) [48] compresses image data on both dimensionality and cardinality via building either sparse low-dimensional representations or a sparse projection matrix for dimension reduction. Go decomposition (GoDec) [49] tried to efficiently and robustly decompose a matrix with the low-rank part L and the sparse part S. Locality structure of manifold can also be combined with sparse representation, such as manifold elastic net (MEN) [50] and graph-regularized sparse coding (GraphSC) [51], laplacian sparse coding (LSc) [35] and Hypergraph laplacian sparse coding (HLSc) [35].

Learning tasks such as classification and clustering usually perform better and cost less (time and space) on compressed representations than on the original data [48]. Therefore, supervised learning and pattern recognition based on the sparse representation coefficients using these sparse learning models are proposed, such as sparse representation-based classification (SRC) [52], Local_SRC [53], Kernel_SRC [54] and the methods outperform traditional classifier, such as SVM, nearest neighbor (NN) and nearest subspace (NS).

2.3. Sparse representation for clustering

Inspired by the successful application of sparse representation in the above-supervised learning approaches, researchers have also exploited sparse representation in unsupervised [55–57] and semi-supervised clustering [58, 59]. The main idea of clustering via sparse representation is to build weight matrix directly from normalized and symmetrized coefficients of sparse representation coefficients, called sparsity-induced similarity (SIS) measure [59]. To a certain extent, weight measure approaches derived from sparse representation can reveal the neighborhood structure without calculating Euclidean distance, which means a great potential to clustering high-dimensional data.

Some significant work applying SIS to spectral clustering is reviewed as follows. Sparse subspace clustering [55] directly uses the sparse representation of vectors lying in a single low-dimensional linear subspace to cluster the data into separate subspaces, followed by applying spectral clustering. It is also extended to clustering data contaminated by noise, missing entries or outliers. Experiments show that its performance for clustering motion trajectories outperforms state-of-the-art methods, such as power factorization and principal component analysis. Image clustering via sparse representation [56] characterizes the graph adjacency structure and graph weights by sparse linear coefficients, which is more effective than Gaussian RBF [12] to cluster an image data set. In semi-supervised learning by sparse representation [18], the graph adjacency structure as well as the graph weights of the directed graph construction is derived simultaneously and in a parameter-free manner to utilize both labeled and unlabeled data. Experiments on semi-supervised face recognition and image classification demonstrate the superiority over the counterparts based on traditional graphs (e.g. ε-ball neighborhood, k-nearest neighbors). Compared to approaches using SIS of real

numbers, non-negative SIS measure [57] exploits the symmetric coefficients of non-negative sparse representation as weight matrix, which outperforms similarity measures, such as SIS and Euclidean (with Gaussian RBF baseline [12]), in cluster analysis of spam images.

However, all the above-existing approaches based on sparse representation treat directly the coefficients or just normalized coefficients of sparse representation as the weight matrix. These cannot exactly reflect the similarity between objects because the coefficients of sparse representation are somehow local similarity and sensitive to outliers. Our approach is expected to provide more effective weight matrix construction using more global content from the solution coefficients of sparse representation.

2.4. Graph construction with sparse representation

In clustering analysis, given a high-dimensional object data set $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathrm{R}^{\mathrm{m} \times \mathrm{n}}$, $\boldsymbol{x}_i = [\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \ldots, \boldsymbol{x}_{im}]^{\mathrm{T}} \in \mathrm{R}^{\mathrm{m}}$, we can use Eq. (5) to represent each objects $\boldsymbol{x}_{\mathrm{i}}$ as a linear combination of other objects. The coefficients vector α_i of $\mathbf{x}_{\mathrm{i}}$ can be calculated by solving the following Lasso optimization:

$$\alpha_i^* = \arg\min_{\alpha_i} \ \lambda\|\alpha_i\|_1 + \|x_i - \mathbf{X}_i\alpha_i\|_2, \tag{6}$$

where $\mathbf{X}_i = \mathbf{X} \backslash \boldsymbol{x}_{\mathrm{i}} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{\mathrm{n}}]$; $\boldsymbol{\alpha}_i = [\boldsymbol{\alpha}_{i,1}, \ldots, \boldsymbol{\alpha}_{i,(i-1)}, \boldsymbol{\alpha}_{i,(i+1)}, \ldots, \boldsymbol{\alpha}_{i,n}]^{\mathrm{T}}$.

Once we get the coefficient vector $\boldsymbol{\alpha}_i$ for each object $\boldsymbol{x}_i$ (i = 1, 2, …, n) as a sparse representation of all other data objects by solving the l_1 optimization Eq. (6),we can construct the weight matrix using different approaches.

Existing weight matrix constructions via sparse representation are based on the assumption that coefficients in the sparse representation reflect the closeness or similarity between two data objects. For example, the SIS measure [20] is computed as:

$$w_{ij} = \frac{\max\{\alpha_{i,j}, 0\}}{\sum_{k=1,k\neq i}^{n} \max\{\alpha_{i,k}, 0\}}; \ \ SIS_{ij} = \frac{w_{ij} + w_{ji}}{2}. \tag{7}$$

The l_1 Directed Graph Construction (DGC) measure [19] is computed as:

$$DGC_{ij} = \frac{|\alpha_{i,j}| + |\alpha_{j,i}|}{2}. \tag{8}$$

Obviously, the similarity calculation using the absolute coefficients in Eq. (8) will mistake the big negative coefficient as high similarity, resulting in a cluster of two objects with apparent opposite attributes value.

The non-negative SIS measure [22] adds a non-negative constraint in l_1 optimization Eq. (6):

$$\alpha_i^* = \arg\min_{\alpha_i} \ \lambda\|\alpha_i\|_1 + \|x_i - \mathbf{X}_i\alpha_i\|_2 \ \text{s.t.}\alpha_{i,j} > 0. \tag{9}$$

Then the non-negative SIS measure is computed as:

$$NN_{ij} = \frac{\alpha_{i,j}}{\sum_{k=1,k\neq i}^{n} \alpha_{i,k}}. \tag{10}$$

3. Sparse representation for spectral clustering

Our proposed clustering algorithm consists of three steps: (1) solving l_1 optimization of sparse representation to obtain the coefficients of each object; (2) constructing weight matrix between objects on the basis of coefficients using more global content forms the solution coefficients of sparse representation; and (3) exploiting the spectral clustering algorithm with the weight matrix to get partition of the graph.

Compared to the direct construction methods using the independent solutions of Eq. (6), we have the assumption that for any two objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$; the more similar they are, the more similar the corresponding coefficient vectors (e.g., α_i and α_j) are not only a particular coefficient ($\alpha_{i,j}$ or $\alpha_{j,i}$). According to this assumption, we propose the following two graph adjacency structure and weight matrix constructions, which are expected to use the global information of the solution coefficients.

3.1. Proximity based on a consistent sign set

To get the similarity of two objects clearly and logically, we firstly find an object set for each of the two different objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from the object data set $\mathbf{X}$, called CSS. This definition is based on the assumption that the more objects of which a pair of objects both positively contribute to the reconstruction, the more similar the pair of objects are. In particular, the sparse reconstruction coefficients corresponding to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ for every object in this set are both positive, defined as follows:

$$CSS(x_i, x_j) = \left\{x_k \middle| (\alpha_{k,i} > 0 \wedge \alpha_{k,j} > 0), k \neq i, k \neq j\right\} \ \forall i \neq j. \tag{11}$$

Furthermore, we can construct graph adjacency structure and weight matrix as follows. A directed edge is placed between objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_{jj}$ if $CSS(\boldsymbol{x}_i, \boldsymbol{x}_j) \neq \Phi$ and the weight between object $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as the ratio of the $CSS(\boldsymbol{x}_i, \boldsymbol{x}_j)$'s cardinal to the total number of objects:

$$w_{ij} = \begin{cases} \dfrac{|CSS(\mathbf{x}_i, \mathbf{x}_j)|}{n} & i \neq j \\ 0 & i = j \end{cases}, \tag{12}$$

where n is the total number of objects in $\mathbf{X}$. Obviously, the weight is between 0 and 1.

3.2. Proximity based on cosine similarity of coefficient vector

We can construct coefficient matrix $\mathbf{A}$ of data set $\mathbf{X}$, to which transforming solution coefficients of Eq. (6) are:

$$\mathrm{A}(i,j) = \alpha'_{i,j} = \begin{cases} \alpha_{i,j} & i \neq j \\ 0 & i = j \end{cases}. \tag{13}$$

A directed edge is placed from object $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if angle cosine of the two corresponding vectors is greater than 0, that is:

$$\frac{\alpha'_i \cdot \alpha'_j}{\left\|\alpha'_i\right\|_2 \times \left\|\alpha'_j\right\|_2} > 0, \tag{14}$$

where $\boldsymbol{\alpha}_i'$ denotes the i-th row vector of A.

The weight between object $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as the cosine similarity of $\boldsymbol{\alpha}_i'$ and $\boldsymbol{\alpha}_j'$:

$$w_{ij} = \begin{cases} \max\left(0, \dfrac{\alpha'_i \cdot \alpha'_j}{\left\|\alpha'_i\right\|_2 \times \left\|\alpha'_j\right\|_2}\right) & i \neq j \\ 0 & i = j \end{cases}. \tag{15}$$

From the above similarity calculation formula, two objects have large similarity in condition that the corresponding solution coefficients of Eq. (6) are much similar, which is expected to use the whole solution coefficients.

3.3. The relationship between consistent sign set (CSS) and cosine similarity of coefficient vector (COS)

Since proposed proximity based on both CSS and COS are trying to exploit more information from the solution coefficient of sparse representation, the relationship between each other is following:

1. Both of them asses the weight between two objects according to the similarity between the corresponding coefficient vectors of the two objects. However, the difference is that proximity based on CSS uses the column vectors of the coefficient matrix **A** while proximity based on COS calculates the similarity between row vectors, which means two understandings of the coefficient matrix. The reason for defining these two approaches like this is just experimental.
2. Proximity based on COS is to calculate the similarity of the original coefficient vector, while CSS can be considered as the discretization of the original coefficient vector with threshold zero. Therefore, proximity based on COS can be seen as the generalization of that based on CSS.
3. Specifically, another equivalent way to understand proximity based on CSS is as follows:
   - Transform the coefficients matrix A to DA: $DA(i,j) = \begin{cases} 1 & A(i,j) > 0 \\ 0 & else \end{cases}$;
   - The weight between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is:

- $w_{ij} = \begin{cases} \dfrac{DA^i \cdot DA^j}{n} & i \neq j \\ 0 & i = j \end{cases}$, where DA^i denotes the i-th column vector of DA.

Obviously, the inner product (DA^i DA^j) between DA^i and DA^j is equal to CSS ($\boldsymbol{x}_i$, $\boldsymbol{x}_j$)'s cardinal $|\mathrm{CSS}(\boldsymbol{x}_i, \boldsymbol{x}_j)|$.

To illustrate the differences between our approaches for weight construction and others also using sparse representation, an example is given as follows. Assume that the coefficient matrix **A** of a data set with five objects obtained from solution coefficients of Eq. (6) is as the following 5×5 matrix:

$$\mathbf{A} = \alpha'_{i,j} = \begin{bmatrix} 0 & 0.3 & 0.6 & 0.6 & -0.7 \\ 0.4 & 0 & 0.5 & 0.6 & -0.6 \\ 0.4 & 0.4 & 0 & -0.1 & -0.2 \\ -0.6 & -0.3 & 0.2 & 0 & 0.7 \\ -0.5 & 0.3 & 0.2 & 0.4 & 0 \end{bmatrix}$$

According to the above introduction of different weight constructions:

1. SIS_{13} = 0.4, SIS_{12} = 0.2, DGC_{13} = 0.5 and DGC_{12} = 0.35, and these numbers show that the similarity between $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$ is larger than that between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. However, in our approaches using more entries in **A**, CSS_{13} = 1/5 = 0.2, CSS_{12} = 2/5 = 0.4, COS_{13} = 0.24 and COS_{12} = 0.98 and these numbers show the different weights compared to the first group, where *CSS* and *COS* are the abbreviation of the above two proximity approaches, respectively.
2. DGC_{25} = 0.45, CSS_{25} = 0, COS_{25} = 0.16, thus *DGC* mistakes the big negative coefficient (α_{25}') as high similarity while *CSS* and *COS* both give lower similarity.

3.4. Algorithm description

Algorithm 1 describes the general procedure for spectral clustering of high-dimensional data, using sparse representation. The basic idea is to extract coefficients of sparse representation (Lines 1–4); construct a weight matrix using the coefficients (Line 5); and feed the weight matrix into a spectral clustering algorithm (Line 6) to find the best partitioning efficiently.

Algorithm 1. General procedure for spectral clustering of high-dimensional data.

Input: high-dimensional training data set $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathrm{R}^{\mathrm{m} \times \mathrm{n}}$, where $\boldsymbol{x}_i = [\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \ldots, \boldsymbol{x}_{im}]\mathrm{T} \in \mathrm{R}^{\mathrm{m}}$ represents the i-th data object; the number of clusters K.

Parameter: penalty coefficient λ for Lasso optimization

Output: cluster labels corresponding to each data object: $\mathbf{c} = [c_1, c_2, \ldots, c_n]$

//standardize the input data for Lasso optimization

1 for each *data object* $\boldsymbol{x}_i \in$ X do

// Solve Eq. (6) with Lasso optimization

2 Set $\mathbf{X}_i = \mathbf{X} \backslash \boldsymbol{x}_i = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n]$;

3 $\boldsymbol{\alpha}_i^* \leftarrow \arg\min \lambda \| \boldsymbol{\alpha}_i \|_1 + \| \boldsymbol{x}_i - \mathbf{X}\boldsymbol{\alpha}_i \|_2$;

4 end

5 $\mathbf{W} \leftarrow$ ConstructWeightMatrix($\boldsymbol{\alpha}$);

6 $\mathbf{c} \leftarrow$ SpectralClustering($\mathbf{W}$);

7 return **c.**

The construct weight matrix () sub-routine can exploit any weight matrix construction method, such as those mentioned in Section 4. In particular, we describe the algorithm for computing the two newly proposed weight matrices, one based on the CSS (see Section 4.1) and the other based on COS of sparse coefficient vectors (see Section 4.2).

Algorithm 2 describes the procedure to construct the weight matrix according to the concept of CSS. To find the *CSS* of each pair of data objects (the two outermost loops), there is the need of checking the sparse coefficients of each remaining object to these two objects, so the time complexity of weight matrix constructions based on *CSS* is $O(n^3)$.

Algorithm 2. Construct weight matrix based on consistent sign set.

Input: Coefficients for sparse representation $\boldsymbol{\alpha}$

Output: Weight matrix $\mathbf{W}$

1 for $i \leftarrow 1$ to n do

2 for $j \leftarrow 1$ to$n$ do

3 if $j = i$ then $w_{ij} \leftarrow 0$;

4 else

5 $n_{css} \leftarrow 0$;

6 for $k \leftarrow 1$ to n do

7 if $k \neq i$ and $k \neq$ j and $\boldsymbol{\alpha}_{k,i} > 0$ and$\boldsymbol{\alpha}_{k,j} > 0$ then

8 $n_{css} \leftarrow n_{css} + 1$;

9 end

10 end

11 $w_{ij} \leftarrow n_{css}/n$;

12 end

13 end

14 end

Algorithm 3 describes the procedure to construct the weight matrix according to the COS of the sparse coefficients between each pair of items. The computation complexity for calculating the

COS of two vectors of length n is $O(n)$, and there are $O(n^2)$ pairs of data objects whose COS needs to be computed. Thus the complexity for COS-based weight matrix construction is $O(n^3)$.

Algorithm 3. Construct weight matrix based on similarity of coefficient vector.

Input: Coefficients for sparse representation $\boldsymbol{\alpha}$

Output: Weight matrix $\mathbf{W}$

1 for $i \leftarrow 1$ to n do

2 for $j \leftarrow 1$ to$n$ do

3 if $j = i$ then $w_{ij} \leftarrow 0$;

4 else

5 $cosine \leftarrow \frac{\alpha_i' \cdot \alpha_j'}{\|\alpha_i'\|_2 \times \|\alpha_j'\|_2}$

6 if $cosine > 0$ then $w_{ij} \leftarrow cosine$;

7 else $w_{ij} \leftarrow 0$;

8 end

9 end

10 end

As shown in line 6 of Algorithm 1, after constructing the weight matrix **W**, we can use the classical spectral clustering algorithm [10] to discover the cluster structure of high-dimensional data.

The main characteristics of our proposed algorithm include the following: (1) compared to traditional graph construction induced from the Euclidean distance or other measures in the original high-dimensional space, the weight matrix is constructed after transforming the high dimensional data space into another space via sparse representation, which is expected to have better performance resulting from the superiority of compressed sensing [58] for high-dimensional data; (2) our graph construction based on consistent sign set or similarity of coefficient vector can simultaneously complete both the graph adjacency and weight matrix, while traditional graph constructions (such as ε-ball neighborhood or k-nearest neighbors) complete the two tasks separately, which are interrelated and should not be separated [19]; (3) rather than existing graph constructions via sparse representation directly and independently applying the solution of l_1 optimization for each object in Eq. (6) to determine a row of the weight matrix, our approach considers the global information from the coefficients of the whole object set to calculate one element in the weight matrix.

4. Experimental results

In this section, we use experimental results to demonstrate the performance of our proposed approaches on real-world data sets using several effectiveness evaluations.

We select three data sets from the UCI machine-learning repository [60] and three face recognition data sets [61–63], which are well known in the machine learning and data mining research community. **Table 1** lists a summary of these data sets.

In Yale and ORL face data sets, each image is transformed into a 32×32 pixel configuration using Matlab Image Processing Toolbox. In Yale B data set, which has 10 clusters, and each cluster has 585 image data, since the original size (5850) is too big, which leads to too much time consumption in clustering, we randomly select 60 images from the totally 585 images in each cluster. In all the data sets, each image is normalized to have unit norm.

We use interior-point method-based l1_ls_matlab tool [64] to solve Eq. (6) for each data object and then implement different weight matrices for algorithm 1 in Matlab to cluster each data set. **Table 2** shows a summary of the proposed and baseline algorithms.

Since the true class labels of each data set are known, five commonly used external cluster validation metrics [66–68] are employed to evaluate the clustering results, namely clustering accuracy (CA) and normalized mutual information (NMI)[1].

Data set	# Instance	# Attributes	# Classes	Source
Heart	270	13	2	UCI
Image (Image segmentation)	2310	18	7	UCI
Yale	165	1024	15	[61]
Yale B	600	1200	10	[62]
ORL Face	400	1024	40	[63]
Movement	360	90	15	UCI

Table 1. Summary of data sets.

Name	Description	Source	Role
CSS	Spectral clustering with weight matrix from consistent sign set	Section 3.1	Solution proposed
COS	Spectral clustering with weight matrix from cosine similarity of sparse coefficients	Section 3.2	Solution proposed
RBF	Spectral clustering with weight matrix from Gaussian RBF	[12]	Baseline
SIS	Spectral clustering with weight matrix from sparsity induced similarity measure	[59]	Baseline
DGC	Spectral clustering with weight matrix from l_1 Directed Graph Construction	[58]	Baseline
NN	Spectral clustering with weight matrix from non-negative sparsity induced similarity measure	[57]	Baseline
KM	k-means clustering	[65]	Baseline

Table 2. Summary of algorithms to be compared.

[1]We use the matlab toolbox from: http://www.cad.zju.edu.cn/home/dengcai/Data/data.html

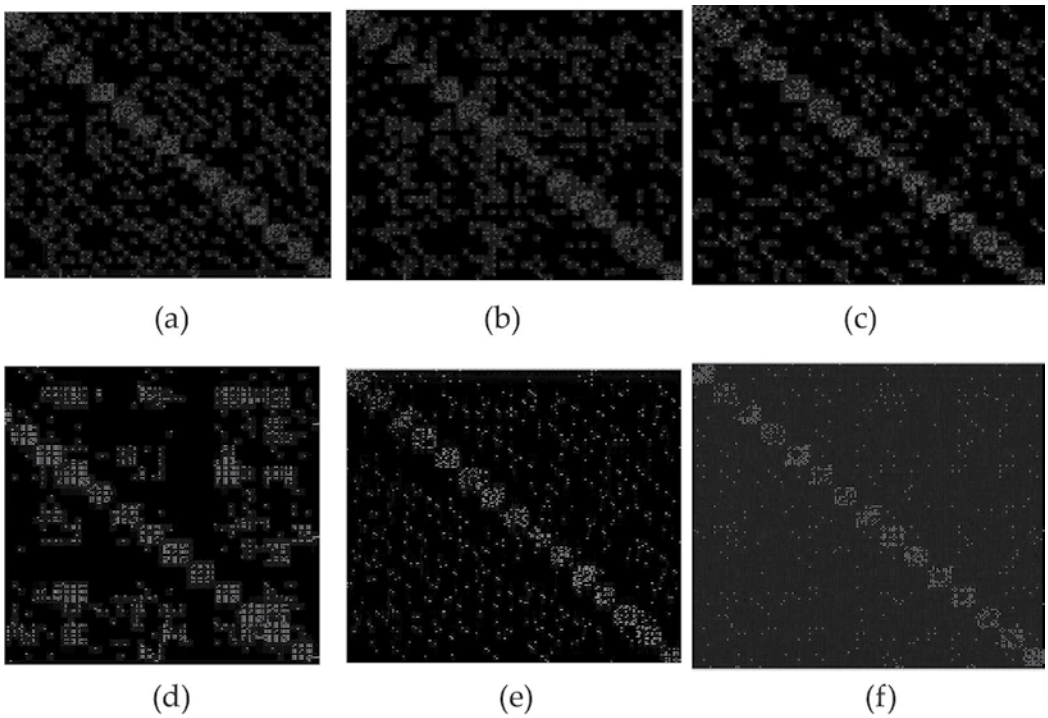

Figure 1. Visualization of the graph weight matrices of the Yale data set, where images from the same subject are arranged together. (a) SIS, (b) DGC, (c) NN, (d) RBF, (e) CSS, (f) COS.

To illustrate the weight matrices from different approaches, we demonstrate the visual property of the proposed graph weight matrices in comparison with traditional ones in **Figure 1**, taking the Yale data set as an example. In **Figure 1**, each subfigure is a weight matrix with N*N (entries larger than the threshold is shown in white, otherwise black) and images from the same cluster are arranged together. These sparse representation-based graphs include consistent sign set (CSS), cosine similarity of coefficient vectors (COS), induced similarity measure (SIS), l_1 directed graph construction (DGC), nonnegative sparsity induced similarity measure (NN) and Gaussian RBF (RBF).

Since none of the original weight matrices constructed by the five approaches is sparse, we set threshold values (0.2 for COS; 0.388 for CSS; 0 for RBF (σ is set 4 in RBF); 0.02 for the other three matrices) to get the best sparse matrices of different threshold values in **Figure 1**. A value larger than the threshold is shown in white, otherwise black. Normally, the clustering performance will be good if the weights between two objects from different clusters are little while weights from the same cluster are large. This comment can be equalized in the matrix with above arrangement that good matrix should be compact in diagonal position and sparse in other positions.

From **Figure 1**, we have the following observations: (1) matrices in all subfigures are compact in diagonal position; (2) the matrix of COS is sparser than others in lower left or upper right

parts. This means that there are less inter-cluster adjacency connections in the COS than other graphs, so COS can encode more discriminating information and hence is more effective in spectral clustering than other traditional graphs; (3) CSS has a similar performance to SIS, DGC and NN Graph in Yale data set.

The clustering results obtained from the seven clustering algorithms with different evaluation metrics are reported in **Tables 3** and **4**, each of which corresponds to one evaluation metric. For each data set, the best results are in bold. All the numbers, except the last two rows in each table, represent the best clustering results using different lasso parameters (λ). The last two rows in each table present the average performance of each algorithm over all six data sets. Since the k-means clustering within spectral clustering is sensitive to initial centroids, we run spectral clustering 50 times for each case and report the mean and standard deviation (std).

From **Tables 3** and **4**, we can clearly see that, generally, CSS or COS algorithm gets the best clustering performance with all the two evaluation metrics. However, there are also some particular cases where CSS or COS does not get best result. For example, though NN gets the best CA on Yale B data sets, COS gets almost the same CA result as NN, that is, from 0.8937 to 0.8940; though NN also gets the best NMI for movement data set, COS gets best result in other metric on this data set. In particular, COS performs better than CSS with mean value of evaluation metrics, and the average standard deviation between 50 random tests of CSS is lowest for all metrics except CA.

Overall, for most data sets, CSS and COS show better performance than those baselines, which are robust across various external validation metrics. However, it is noticed that COS outperforms CSS in terms of all average mean metrics except CA, and CSS outperforms COS in

Data set		CSS	COS	DGC	SIS	NN	BRF	KM
Heart	Mean	0.7704	**0.8174**	0.5852	0.7889	0.7519	0.7963	0.7320
	(Std)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	(0.0882)
Image	Mean	0.7631	**0.7921**	0.7020	0.7820	0.7360	0.5335	0.6215
	(Std)	**(0.0148)**	(0.0323)	(0.0339)	(0.0341)	(0.0379)	(0.0305)	(0.0355)
Yale	Mean	0.6823	**0.7408**	0.7178	0.7023	0.6417	0.6635	0.5482
	(Std)	(0.0432)	(0.0345)	(0.0414)	**(0.0314)**	(0.0412)	(0.0395)	(0.0529)
Yale B	Mean	0.8572	0.8937	0.8320	0.8620	**0.8940**	0.6918	0.6862
	(Std)	(0.0791)	(0.0713)	(0.0768)	(0.0635)	(0.0767)	**(0.0226)**	(0.0721)
ORL face	Mean	0.7315	**0.7570**	0.7225	0.7243	0.6903	0.7314	0.7196
	(Std)	(0.0247)	(0.0218)	(0.0222)	(0.0244)	**(0.0207)**	(0.0252)	(0.0311)
Movement	mean	0.5241	**0.5472**	0.5009	0.5183	0.5304	0.4874	0.4653
	(Std)	(0.0222)	**(0.0187)**	(0.0248)	(0.0193)	(0.0271)	(0.0232)	(0.0203)
Average	Mean	0.7214	**0.7580**	0.6767	0.7296	0.7074	0.6506	0.6288
	(Std)	(0.0307)	(0.0298)	(0.0332)	(0.0288)	(0.0339)	**(0.0235)**	(0.0500)

Table 3. Evaluation of all algorithms with CA as metric.

Data set		CSS	COS	DGC	SIS	NN	BRF	KM
Heart	Mean	0.2208	**0.3149**	0.0511	0.1791	0.0331	0.2712	0.2028
	(Std)	**(0.0000)**	**(0.0000)**	**(0.0000)**	**(0.0000)**	(0.0312)	**(0.0000)**	(0.1013)
Image	Mean	0.7088	**0.7451**	0.5921	0.7319	0.6637	0.7451	0.6122
	(Std)	**(0.0071)**	(0.0184)	(0.0171)	(0.0176)	(0.0357)	(0.0284)	(0.0437)
Yale	Mean	0.7137	**0.7815**	0.7513	0.7641	0.6926	0.6989	0.6484
	(Std)	(0.0211)	**(0.0183)**	(0.0203)	(0.0188)	(0.0198)	(0.0271)	(0.0342)
Yale B	Mean	0.9008	**0.9526**	0.9202	0.9379	0.9510	0.7768	0.7858
	(Std)	(0.0302)	(0.0360)	(0.0422)	(0.0326)	(0.0398)	**(0.0224)**	(0.0621)
ORL face	Mean	0.8477	**0.8688**	0.8492	0.8547	0.8426	0.8512	0.8620
	(Std)	(0.0116)	(0.0108)	**(0.0105)**	(0.0118)	(0.0109)	(0.0140)	(0.0157)
Movement	Mean	0.5891	0.5914	0.5933	0.6000	**0.6306**	0.5741	0.5818
	(Std)	(0.0128)	**(0.0124)**	(0.0140)	(0.0130)	(0.0173)	(0.0169)	(0.0180)
Average	Mean	0.6635	**0.7090**	0.6262	0.6779	0.6356	0.6529	0.6155
	(Std)	**(0.0138)**	(0.0160)	(0.0174)	(0.0156)	(0.0258)	(0.0181)	(0.0458)

Table 4. Evaluation of all algorithms with NMI as metric.

terms of all average standard metrics. It can be explained that CSS is more stable because its discretization may lower the variance of the pairwise of similarity, while COS get more generalized information of the pairwise of similarity leading to better average metrics but higher variance. Therefore, the choice between stability and quality should be taken into account when it is facing the clustering problem, in practice, using this kind of approach.

Finally, we plot the averages of the mean value and standard deviation (from the last two rows of the five tables), for comparing clustering algorithms, as shown in **Figure 2**.

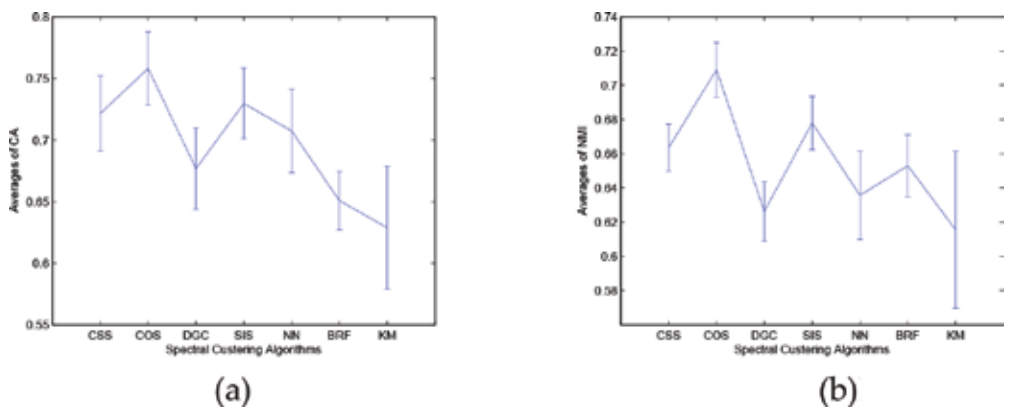

Figure 2. Error bar of different algorithms (a) CA, (b) NMI.

5. Conclusion

In this chapter, we present a study of spectral clustering based on sparse representation, using two novel weight matrix construction approaches to assess the consistency of two sparse vectors. This construction considers the global information of the solution coefficient vectors of two objects to analyze the similarity between these two objects rather than directly using the sparse coefficients, which only considers local information. Evaluation experiments on real-world data sets show that spectral clustering for high-dimensional data using our novel weight matrix construction exploiting global information outperforms direct k-means and spectral clustering approaches using Gaussian RBF, SIS, l_1-directed graph construction and non-negative SIS in five evaluation metrics (CA and NMI).

These results demonstrate a reliable performance of our algorithm and therefore promise wide applicability in practice. The findings also shed light on developing global solutions theories in the future work.

Figure 2 clearly demonstrates that COS and CSS algorithms outperform other algorithms, and COS is better than CSS on average. CSS obtains the least average value of standard deviation among all seven algorithms. The KM and DGC algorithms have comparable performance, which is usually worse than the other algorithms.

Author details

Xiaodong Feng

Address all correspondence to: fengxd1988@hotmail.com

School of Public Administration, University of Electronic Science and Technology of China, Chengdu, Sichuan, China

References

[1] Liu Y, Wang X, Wu C. ConSOM: A conceptional self-organizing map model for text clustering. Neurocomputing. 2008;**71**:857-862

[2] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009;**42**:30-37

[3] Bhatt CA, Kankanhalli MS. Multimedia data mining: State of the art and challenges. Multimedia Tools Applications. 2011;**51**:35-76

[4] Zhang X, Liu J, Du Y, Lv T. A novel clustering method on time series data. Expert Systems with Applications. 2011;**38**:11891-11900

[5] Sun J, Chen W, Fang W, Wun X, Xu W. Gene expression data analysis with the clustering method based on an improved quantum-behaved particle swarm optimization. Engineering Applications of Artificial Intelligence. 2012;**25**:376-391

[6] Steinbach M, Ertoz L, Kumar V. The challenges of clustering high dimensional data. In: New Directions in Statistical Physics. Berlin, Germany: Springer; 2004. pp. 273-309

[7] Chen X, Ye Y, Xu X, Huang JZ. A feature group weighting method for subspace clustering of high-dimensional data. Pattern Recognition. 2012;**45**:434-446

[8] Song Q, Ni J, Wang G. A fast clustering-based feature subset selection algorithm for high dimensional data. IEEE Transactions on Knowledge and Data Engineering. 2011;**9**:1-14

[9] Parsons L, Haque E, Liu H. Subspace clustering for high dimensional data: A review. ACM SIGKDD Explorations Newsletter. 2004;**6**:90-105

[10] Von Luxburg U. A tutorial on spectral clustering. Statistics and Computing. 2007;**17**:395-416

[11] Shi J, Malik J. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2000;**22**:888-905

[12] Ng AY, Jordan MI, Weiss Y. On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems. 2002;**2**:849-856

[13] Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation. 2003;**15**:1373-1396

[14] Tenenbaum JB, De Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000;**290**:2319-2323

[15] Donoho DL. Compressed sensing. IEEE Transactions on Information Theory. 2006;**52**: 1289-1306

[16] Wright J, Ma Y, Mairal J, Sapiro G, Huang TS, Yan S. Sparse representation for computer vision and pattern recognition. Proceedings of the IEEE. 2010;**98**:1031-1044

[17] Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999;**401**:788-791

[18] Paatero P, Tapper U. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics. 1994;**5**:111-126

[19] Ding CH, He X, Simon HD. On the equivalence of nonnegative matrix factorization and spectral clustering. In: SIAM International Conference on Data Mining; 2005. pp. 606-610

[20] Lee DD, Seung HS. Algorithms for non-negative matrix factorization. Advances in Neural Information Processing Systems. 2001;**13**:556-562

[21] Sandler R, Lindenbaum M. Nonnegative matrix factorization with Earth mover's distance metric for image analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2011;**33**:1590-1602

[22] Guan N, Tao D, Luo Z, Shawe-Taylor J, MahNMF. Manhattan Non-Negative Matrix Factorization, arXiv preprint arXiv:1207.3438;2012

[23] Kim D, Sra S, Dhillon IS. Fast Newton-type methods for the least squares nonnegative matrix approximation problem. In: SIAM International Conference on Data Mining; 2007

[24] Guan N, Tao D, Luo Z, Yuan B. Online nonnegative matrix factorization with robust stochastic approximation. IEEE Transactions on Neural Networks and Learning Systems. 2012;**23**:1087-1099

[25] Sun M, Hamme HV. Large scale graph regularized non-negative matrix factorization with l1 normalization based on Kullback–Leibler divergence. IEEE Transaction on Signal Processing. 2012;**60**:3876-3880

[26] Hoyer PO. Non-negative matrix factorization with sparseness constraints. The Journal of Machine Learning Research. 2004;**5**:1457-1469

[27] Esser E, Moller M, Osher S, Sapiro G, Xin J. A convex model for nonnegative matrix factorization and dimensionality reduction on physical space. IEEE Transactions on Image Processing. 2012;**21**:3239-3252

[28] Zafeiriou S, Tefas A, Buciu I, Pitas I. Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. IEEE Transactions on Neural Networks. 2006;**17**:683-695

[29] Cai D, He X, Han WJ, Huang TS. Graph regularized nonnegative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2011;**33**:1548-1560

[30] Guan N, Tao D, Luo Z, Yuan B. Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent. IEEE Transactions on Image Processing. 2011;**20**: 2030-2048

[31] Liu H, Wu Z, Li X, Cai D, Huang TS. Constrained nonnegative matrix factorization for image representation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2012;**34**:1299-1311

[32] Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science. 2000;**290**:2323-2326

[33] Luo Y, Tao D, Geng B, Xu C, Maybank S. Manifold regularized multi-task learning for semi-supervised multi-label image classification. 2013;**22**:523-536

[34] Belkin M, Niyogi P, Sindhwani V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. The Journal of Machine Learning Research. 2006;**7**:2399-2434

[35] Gao S, Tsang I, Chia L. Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013;**35**: 92-104

[36] Zhou Y, Barner K. Locality constrained dictionary learning for nonlinear dimensionality reduction. IEEE Signal Processing Letters. 2012;**20**:335-338

[37] Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y. Locality-constrained linear coding for image classification, computer vision and pattern recognition (CVPR). In: 2010 IEEE Conference on, (IEEE, 2010); pp. 3360-3367

[38] Yu J, Tao D, Wang M. Adaptive hypergraph learning and its application in image classification. IEEE Transactions on Image Processing. 2012;**21**:3262-3272

[39] Yu J, Wang M, Tao D. Semi-supervised multiview distance metric learning for cartoon synthesis. IEEE Transactions on Image Processing. 2012;**21**:4636-4648

[40] Deng X, Liu X, Song M, Cheng J, Bu J, Chen C. LF-EME: Local features with elastic manifold embedding for human action recognition. Neurocomputing. 2013;**99**:144-153

[41] Yu J, Liu D, Tao D, Seah HS. Complex object correspondence construction in two-dimensional animation. IEEE Transactions on Image Processing. 2011;**20**:3257-3269

[42] Donoho DL. For most large underdetermined systems of equations, the minimal l1-norm near-solution approximates the sparsest near-solution. Communications on Pure and Applied Mathematics. 2006;**59**:907-934

[43] Candès EJ. The restricted isometry property and its implications for compressed sensing. Comptes Rendus Mathematique. 2008;**346**:589-592

[44] Candès EJ, Compressive sampling. In: Proceedings of the International Congress of Mathematicians; 22-30 August 2006: Invited Lectures; Madrid. 2006. pp. 1433-1452

[45] Chen SS, Donoho DL, Saunders MA. Atomic decomposition by basis pursuit. SIAM Journal on Scientific Computing. 1998;**20**:33-61

[46] Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological). 1996:267-288

[47] Zou H, Hastie T. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2005;**67**:301-320

[48] Zhou T, Tao D. Double shrinking for sparse dimension reduction. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013;**35**:92-104

[49] Zhou T, Tao D, Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11); 2011. pp. 33-40

[50] Zhou T, Tao D, Wu X. Manifold elastic net: A unified framework for sparse dimension reduction. Data Mining and Knowledge Discovery. 2011;**22**:340-371

[51] Zheng M, Bu J, Chen C, Wang C, Zhang L, Qiu G, Cai D. Graph regularized sparse coding for image representation. IEEE Transactions on Image Processing. 2011;**20**:1327-1336

[52] Wright J, Yang AY, Ganesh A, Sastry SS, Ma Y. Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2009;**31**: 210-227

[53] Li C, Guo J, Zhang H. Local sparse representation based classification. In: Pattern Recognition (ICPR), 2010 20th International Conference on, (IEEE, 2010); pp. 649-652

[54] Gao S, Tsang IW, Chia L. Kernel sparse representation for image classification and face recognition. In: Computer Vision–ECCV 2010. Berlin, Germany: Springer; 2010. pp. 1-14

[55] Elhamifar E, Vidal R. Sparse subspace clustering. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, (IEEE, 2009); pp. 2790-2797

[56] Jiao J, Mo X, Shen C. Image clustering via sparse representation. In: Advances in Multimedia Modeling. Springer; 2010. pp. 761-766

[57] Gao Y, Choudhary A, Hua G. A nonnegative sparsity induced similarity measure with application to cluster analysis of spam images. In: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, (IEEE, 2010); pp. 5594-5597

[58] Yan S, Wang H. Semi-supervised learning by sparse representation. In: SIAM International Conference on Data Mining; 2009. pp. 792-801

[59] Cheng H, Liu Z, Yang J. Sparsity induced similarity measure for label propagation. In: Computer Vision, 2009 IEEE 12th International Conference on, (IEEE, 2009); pp. 317-324

[60] UCI Data Sets. http://archive.ics.uci.edu/ml/datasets/ [Accessed: November 10, 2012]

[61] Georghiades A. Yale Face. 2013. http://cvc.yale.edu/projects/yalefaces/yalefaces.html

[62] Georghiades A, Belhumeur P, Kriegman D, Yale Face_B. 2013. http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html

[63] ORL Face. AT&T Lab Cambridge. 2013. http://www.face-rec.org/databases/

[64] Koh K, Kim SJ, Boyd S, l1_ls_matlab. 2013. http://www.stanford.edu/~boyd/l1_ls/l1_ls_matlab.zip

[65] Hartigan JA, Wong MA. Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics). 1979;**28**:100-108

[66] Huang Z. Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery. 1998;**2**:283-304

[67] Jing L, Ng MK, Huang JZ. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. IEEE Transactions on Knowledge and Data Engineering. 2007;**19**:1026-1041

[68] Deng Z, Choi K, Chung F, Wang S. EEW-SC, enhanced entropy-weighting subspace clustering for high dimensional gene expression data clustering analysis. Applied Soft Computing. 2011;**11**:4798-4806

Chapter 10

Performance Assessment of Unsupervised Clustering Algorithms Combined MDL Index

Hadeel K. Aljobouri, Hussain A. Jaber and
Ilyas Çankaya

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74506

Abstract

Best clustering analysis should be resisting the presence of outliers and be less sensitive to initialization as well as the input sequence ordering. This chapter compares the performance among three of the unsupervised clustering algorithms: neural gas (NG), growing neural gas (GNG), and robust growing neural gas (RGNG). A complete explanation of NG and GNG algorithms is presented in the next comparison with RGNG. Another comparison due to the minimum description length (MDL) criterion between RGNG used MDL value as the clustering validity index versus GNG and NG combined with MDL. Statistical estimations are applied to explain the meaning of the output results when these algorithms are fed to the synthetic 2D dataset. The techniques introduced in this chapter are designed and implemented in a simple software package using a MATLAB-based graphical user interface (GUI) tool, which allows users to interact with the clustering techniques and output data easily.

Keywords: clustering techniques, graphical user interface (GUI), growing neural gas (GNG), neural gas (NG), robust growing neural gas (RGNG)

1. Introduction

Cluster analysis [1] is a robust tool for exploring the underlining structures in data and grouping them with similar objects called clusters. Cluster analysis found applications in different fields ranging from the main task of data mining applications [2] such as scientific data exploration, spatial database applications, web analysis, marketing, medical diagnostics, computational biology, etc., to statistical data analysis that is used in many fields including machine

learning, pattern recognition [3], image analysis [4], information retrieval [5], and bioinformatics [6]. There are different algorithms related to neural networks; the most popular are K-means, the self-organizing map (SOM), neural gas (NG), and growing neural gas (GNG) [7, 8].

The goal of this work is to present a comparison among neural gas (NG), growing neural gas (GNG), and robust growing neural gas (RGNG) approaches that are related to neural networks, as well as design a new simulation tool for the purpose of education and scientific research using unsupervised learning methods. Due to the difficulty in introducing these algorithms in literature, the three techniques have been presented using a simple graphical user interface (GUI) model. Alziarjawey et al. [9] introduced an application of Matlab GUI in the medical field using the ECG signal for heart rate monitoring and PQRST detection. They introduced another application by developing a software package based on GUI, which consists of two modules using many important methods derived from linear algebra [10]. Aljobouri et al. [11] designed an educational tool for biosignal processing and medical imaging using a GUI package. The user friendly package explained in this work can be used easily by: choosing any method, changing the predefined parameters for each algorithm and comparing the results. Hence, it can be used without any programming knowledge. The interested reader may find more technical details in our previous reports and publications [12, 13].

The current study is organized as follows: Section 2 provides the unsupervised clustering algorithms. Case studies are described in Section 3. Sections 4 and 5 present the experimental implementation on the synthetic dataset and clustering package design, respectively. Finally, Section 6 concludes the paper and introduces future work.

2. Unsupervised clustering algorithms

In this section, a review of the NG, GNG, and RGNG algorithms are presented. Because of the length and complexity of these algorithms, along with the mathematical model, flowcharts are designed for the three algorithms in this work in order to make it more understandable and easier to write the related codes.

2.1. NG algorithm

The NG network algorithm is a simple artificial neural network algorithm for finding optimal data representations based on reference vectors (prototype vectors). It was first introduced in 1991 [14] and is based on Kohonen's SOM [15]. Because of the dynamics of the reference vectors during the adaptation process, this algorithm was called "neural gas" that spread itself as a gas through the data space. NG is unlike other methods that consider distance as a rank like Euclidean distance, but it proposes a new way of calculating the influence of distance. Nearer prototypes in NG algorithm are more affected, but it does not depend directly on the influence of distance.

NG has been successfully applied to clustering [16], speech recognition [17], image processing [18], vector quantization, pattern recognition, topology representation, etc., [19, 20] especially where there is a problem arriving at vector quantization or data compression.

http://dx.doi.org/10.5772/intechopen.74506

It adapts the reference vectors (prototype vectors) "w_i" without any fixed topological arrangement within the network. NG not just adapts the winner vector for a specific input vector as a single-layered soft competitive learning neural network, but also updates the residual reference vectors according to the input vector nearness using a soft-max updating rule [21]. The main advantages of NG network [22] are: (1) lower distortion error than other clustering algorithms (k-means, maximum-entropy and SOM), (2) faster assemblage due to low distortion errors, (3) submission a stochastic gradient descent on a specific energy surface.

The NG algorithm is represented by the dependence of updating strengths for c reference vectors $w_{ci}(i_0, i_1, \ldots, i_{N-1})$ on their position ranking. If the input vector is presented by x, the definition of the position ranking $(w_{i0}, w_{i1}, \ldots, w_{ik})$ of the reference vectors w_{ci} will be:

w_{i0} is adjacent to x.

w_{i1} is second adjacent to x

for $k = 1, 2, \ldots, N-1$; $\|x - w_j\| < \|x - w_{ik}\|$, where w_{ik} is the reference vector, which has k vectors w_j.

$k_i(x, w)$ is the ranking index associated with each weight w_i.

The updating step of adjusting w_i according to a Hebb-like learning rule is given by:

$$\Delta w_i = \varepsilon(t).h_\lambda\left(k_i(x, w)\right).(x - w_i), \quad i = 1, 2, \ldots, c \tag{1}$$

where:

$h_{(.,.)}$: deterministic function with some regularity condition imposed on it.

$\varepsilon(t) \in [0, 1]$: the learning rate (step size) that characterize the total range of the variation. This extent is represented by $\{\varepsilon(t) = \varepsilon_i.\left(\varepsilon_f/\varepsilon_i\right)^{t/(\text{Max\_ter})}\}$, so Max_iter, so and t denote the maximum number of repetitions and the repetition step respectively.

$h_\lambda\left(k_i(v, w) \in [0, 1]\right.$: considers the w_i within the input extent.

for $h_\lambda(k) \in [0, 1]$, the exponential form $\exp(-k/\lambda)$ was proposed [22] to obtain the best extensive result compared to other options like the Gaussian function.

λ: finds the number of reference vectors that significantly change their positions in the updating steps and usually individually decrease with the iteration step t as: $\lambda(t) = \lambda_i.\left(\lambda_f/\lambda_i\right)^{t/(\text{Max\_iter})}$.

The NG algorithm is widely related to the structure of fuzzy clustering methods [23]. So, NG used the uncertainty of the relationship value $\left(h_\lambda(k_i(x, w))\right)/(C(\lambda))$to set each input vector "x" to all the reference vectors w_i $(i = 1, 2, \ldots, c)$instead of using u_{ij} $(2 \le i \le c, 1 \le j \le N)$ in FCM algorithm. This algorithm is based on solving a cost function using iterative methods plus the familiarity with linear optimization methods, essentially the gradient descent method and Newton's method. Therefore, the NG cost function to optimize [22] is:

$$E_{ng} = \frac{1}{2C(\lambda)} \sum_{i=1}^{c} \int P(x)\, h_\lambda\left(k_i(x, w)\right) \left\|x - w_i\right\|^2 \tag{2}$$

with

$$C(\lambda) = \sum_{i=1}^{c} h_{\lambda}(k_i) = \sum_{k=0}^{c-1} h_{\lambda}(k) \tag{3}$$

Martinetz et al. [22, 26] introduced this cost function and proved that the updating in the Hebb-like learning rule can be derived by a stochastic gradient descent on this function. By starting with a large value of λ and reducing it slowly, a good reference vector can be obtained.

Due to the sequential learning scheme in NG algorithms and the use of the neighborhood dealing rule, NG became less sensitive to various initializations due to the sequential learning scheme and use of neighborhood cooperation rule with comparison to other clustering algorithms like k-means and FCM.

Before feeding the NG algorithm, there are some parameters that have to be defined:

N: maximal number of neurons

$\varepsilon_i, \varepsilon_f$: step size

λ_i, λ_f: decay constant

T_i, T_f: life-time

t_{max} = Max_iter = (maximal number of iterations)

Figure 1 shows the flowchart of the NG algorithm. Although the NG model has many advantages as mentioned earlier, it also has some limitations. It depends on decaying parameters that change over time; it is incapable of finding a network size and structure automatically and continue learning. Hence, based on the NG algorithm, the GNG algorithm was introduced by Fritzke [24, 25], which has an advantage over NG algorithms through its ability to modify the network topology by removing edges with its age variable. Moreover, during the growth process associated with the neighborhood updating rule, there is no need for the neighborhood sorting step [24, 25]. It has the ability to find a network size and structure automatically, and continue learning, adding units and connections, until a performance criterion is fulfilled.

2.2. GNG algorithm

In the GNG algorithm, Fritzke [24, 27] proposed changing the unit numbers (mostly increased) during SOM network with a variable topological structure [24, 25]. This growth mechanism is combined with topology formation rules using the competitive Hebbian learning (CHL) [26] and the earlier proposed growing mechanism inherited from the growing cell structures [27] to form a new model.

The GNG algorithm needs only constant parameters; it is not required to set the amount of prototypes. The main idea behind the GNG is to start with a minimal network size and insert a few new neurons and connections respectively in a growing structure by using a vector quantization until the desired characteristics of the model is fulfilled (e.g., net size, time limit, predefined number of neurons inserted, quality or some performance measure). To determine where to insert new units, local error measures are gathered during the adaptation process.

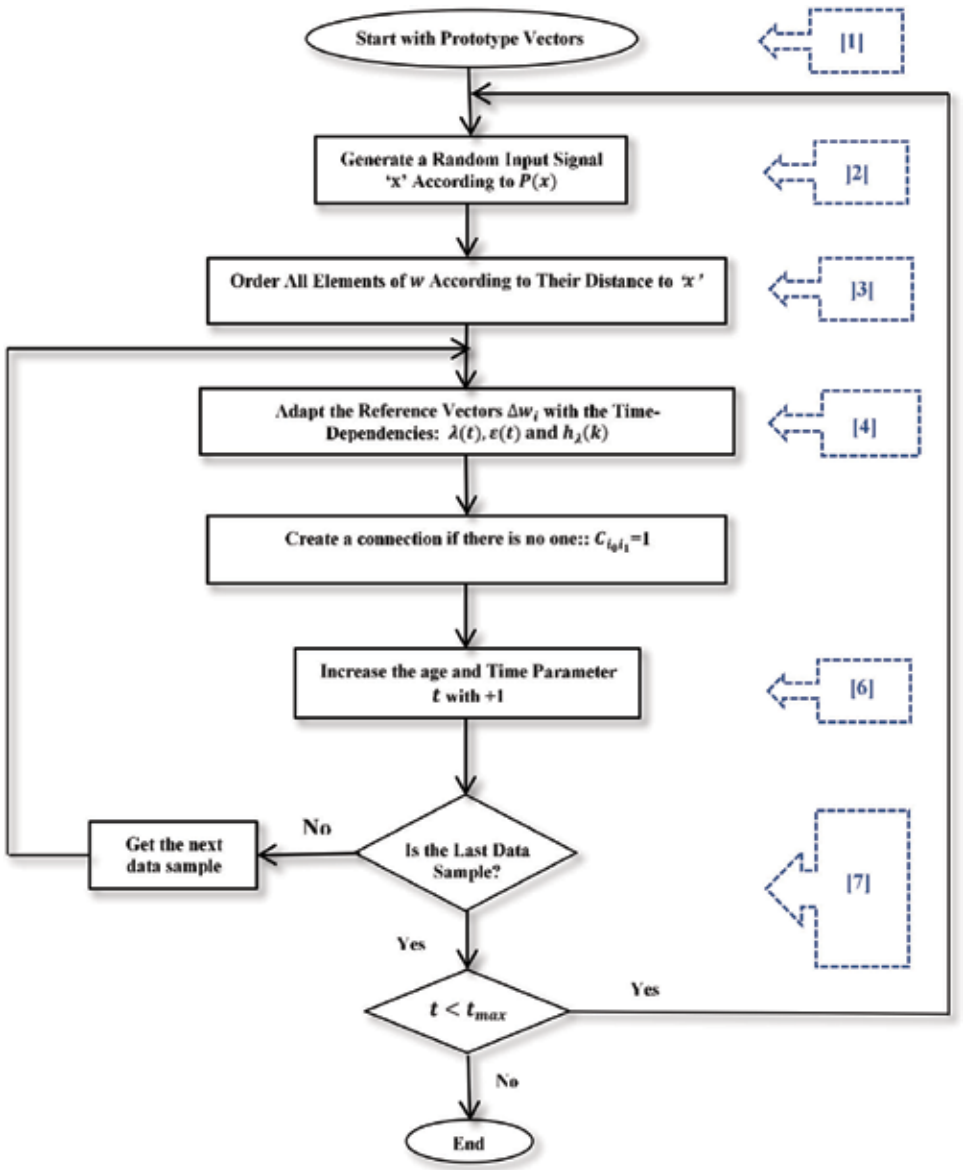

Figure 1. The flowchart of an NG algorithm.

Each new unit is inserted near the unit that has accumulated the highest error, and a connection between the winner and the second nearest neuron is formed using the competitive Hebbian learning algorithm.

Before feeding the GNG algorithm, there are some parameters that have to be defined:

N: maximal number of neurons

$\varepsilon_b, \varepsilon_n$: constant learning rate for the winner and its topological neighbors, respectively

λ: iteration number

α: reduction of error counter by inserting a new neuron

β: value that will reduce the overall value of the error counter every iteration step

Max_iter:: maximal number of iterations

Each reference vector $w_i, i = 1, 2, \ldots, c$, has a set of edges emanating from it. It is defined to connect with its direct topological neighbors. The GNG algorithm starts by initializing a few prototype vectors (usually two) $W = \{w_1, w_2\}$ with reference vectors that are chosen randomly. New prototype vectors are successively inserted. The learning rates $\varepsilon_b, \varepsilon_n$ are used in the training procedure and a connection is formed C, $C \subset w \times w$, to the empty set: $C = \varnothing$.

The pre-specified maximum number of prototypes or neurons is set to grow as pre_numnode; and the maximum predefined training epoch Max_iter is set during each growth stage with the largest local accumulated error measure. The data set used for training is $X = \{x_1, x_2, \ldots, x_N\}$. Then, the initial training epoch number is set as $m = 0$ and the iteration step in the training epoch m is set as: $t = 0$.

Figure 2 presents the flowchart of the GNG algorithm. This figure shows that nonfunctional prototypes that do not win over long time intervals may be detected by tracing the changes of an age variable associated with each edge. Hence, the GNG algorithm has an advantage against the NG algorithm through its ability to modify the network topology by removing edges with their age variable (not being refreshed for a time interval α_max) and the resultant nonfunctional prototypes. In the GNG algorithm, the growth process associated with the neighborhood updating rule used is somewhat similar to the neighborhood, decreasing procedure in NG. However, unlike the NG algorithm, there is no need for the neighborhood sorting step.

2.3. RGNG algorithm

Any robust algorithm should have the following features [28]:

1. It should achieve a good precision for the given model.
2. The performance of the given model may have few deviations from the assumptions made, but these deviations should not weaken the performance, except by a small degree.
3. The presence of large deviations from the model assumptions should not cause disaster.

If classical clustering methods are to be used as prototype based clustering algorithms, the major robustness problems are the sensitivity to initialization, the order of input vectors, and existence of many outliers, but each well executed regarding condition 1. Due to the growth scheme associated with the GNG algorithm, the algorithm faces the "dead nodes" problem. This occurs due to inappropriate initializations that led to some prototypes that may never win through the training process.

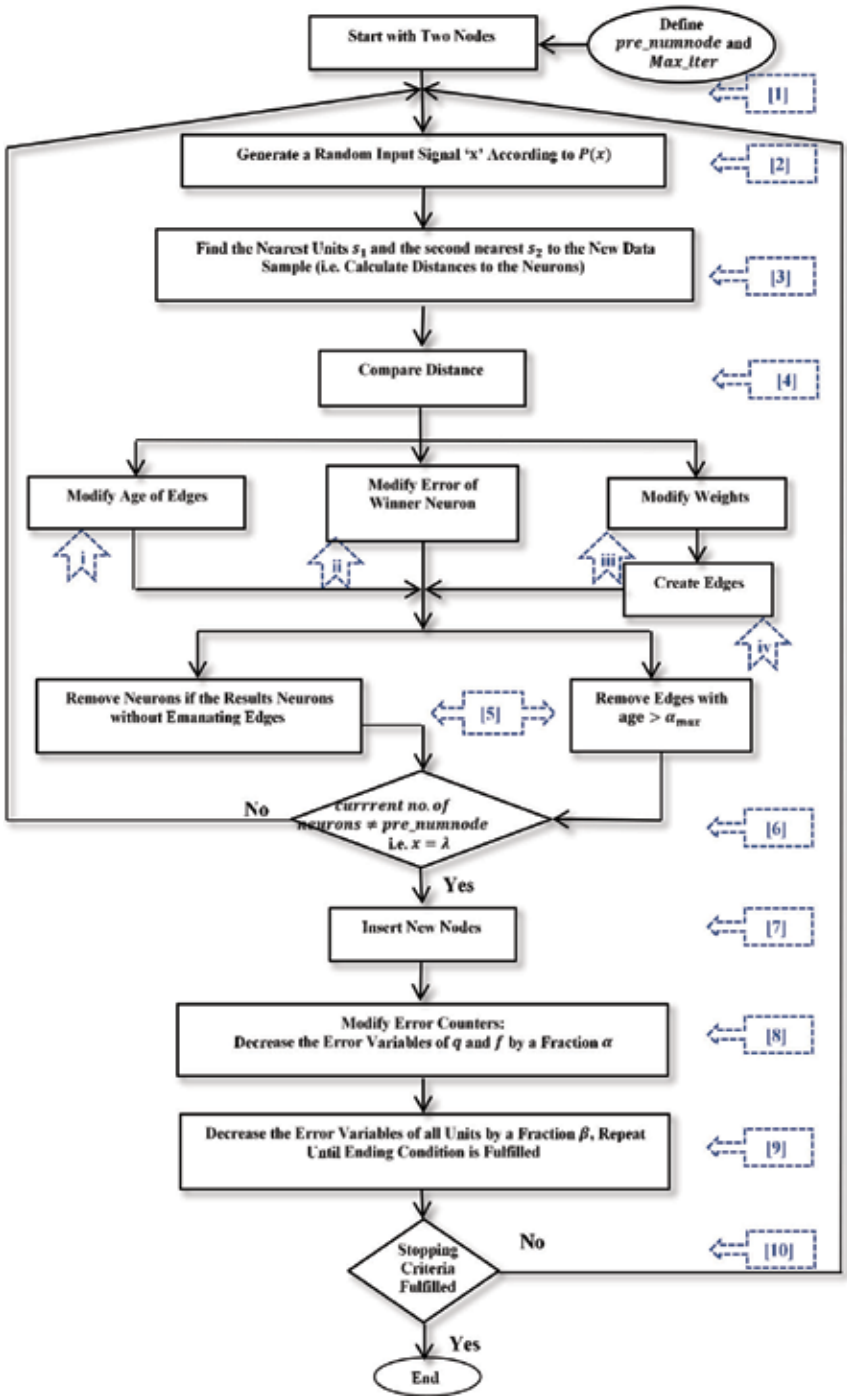

Figure 2. The flowchart of the GNG algorithm.

Even with initialization-insensitive clustering methods, good clustering results may not be obtained if the order of the input sequence is not chosen properly.

Even with the initialization insensitive clustering methods, good clustering results may not be obtained if the order of the input sequence is not chosen properly. As well as the introduced problem gets along with the sensitivity for initialization and the order of input vectors, there also another problem attributable to the existence of many outliers. This implies the GNG network may fail to differentiate the outliers from the inliers through the original prototype updating rule when many of outliers exist in a data set. These outliers can be regarded as input vectors that different from data points belonging to the ordinary clusters (inliers).

For these limitations of the GNG algorithm, a novel robust clustering algorithm was proposed [29] within the GNG structure, namely the robust growing neural gas (RGNG) network. RGNG possesses better robustness than the original GNG algorithm because of its succession properties. It also incorporates with it several robust strategies, such as outlier resistant scheme, adaptive modulation of learning rates, and cluster repulsion method.

Therefore, compared to the GNG network, the RGNG network is insensitive to initialization, input sequence ordering, the presence of outliers, and determination of the optimal number of clusters. The minimum description length (MDL) value was used with RGNG as the clustering validity index [30, 31]. The MDL value is used to find the optimal number of clusters and their center positions corresponding to the smallest MDL. This determined automatically the optimal number of clusters by searching the extreme value of the MDL measure through the network growing process.

Before feeding the RGNG algorithm, there are some parameters that have to be defined:

N: maximal number of neurons

ε_b^l: learning rate of the winner

ε_n^l: learning rate of its topological neighbors

ε_{bf}^l, ε_{bi}^l, ε_{nf}^l, ε_{ni}^l: initial and final values of ε_b^l and ε_n^l

$\alpha_{\max}$: maximal age of a connection

β: mobility of the winner's neighborhood toward the input vector

k, η: parameters used to determine the MDL value

Max_iter: maximal number of iterations

The maximum number of nodes may be set to increase the pre_numnode and Max_iter and during each step with a defined number of nodes. The initial training epoch number ($m = 0$) and the iteration stage in training epoch m at $t = 0$ may also be set. Hence, the total iteration step iter during each increasing step is iter $= m.N + t$, where N is an actual number of the neuron. The dataset used for training is $X = \{x_1, x_2, \ldots, x_N\}$.

Figure 3 presents the flowchart of the RGNG algorithm.

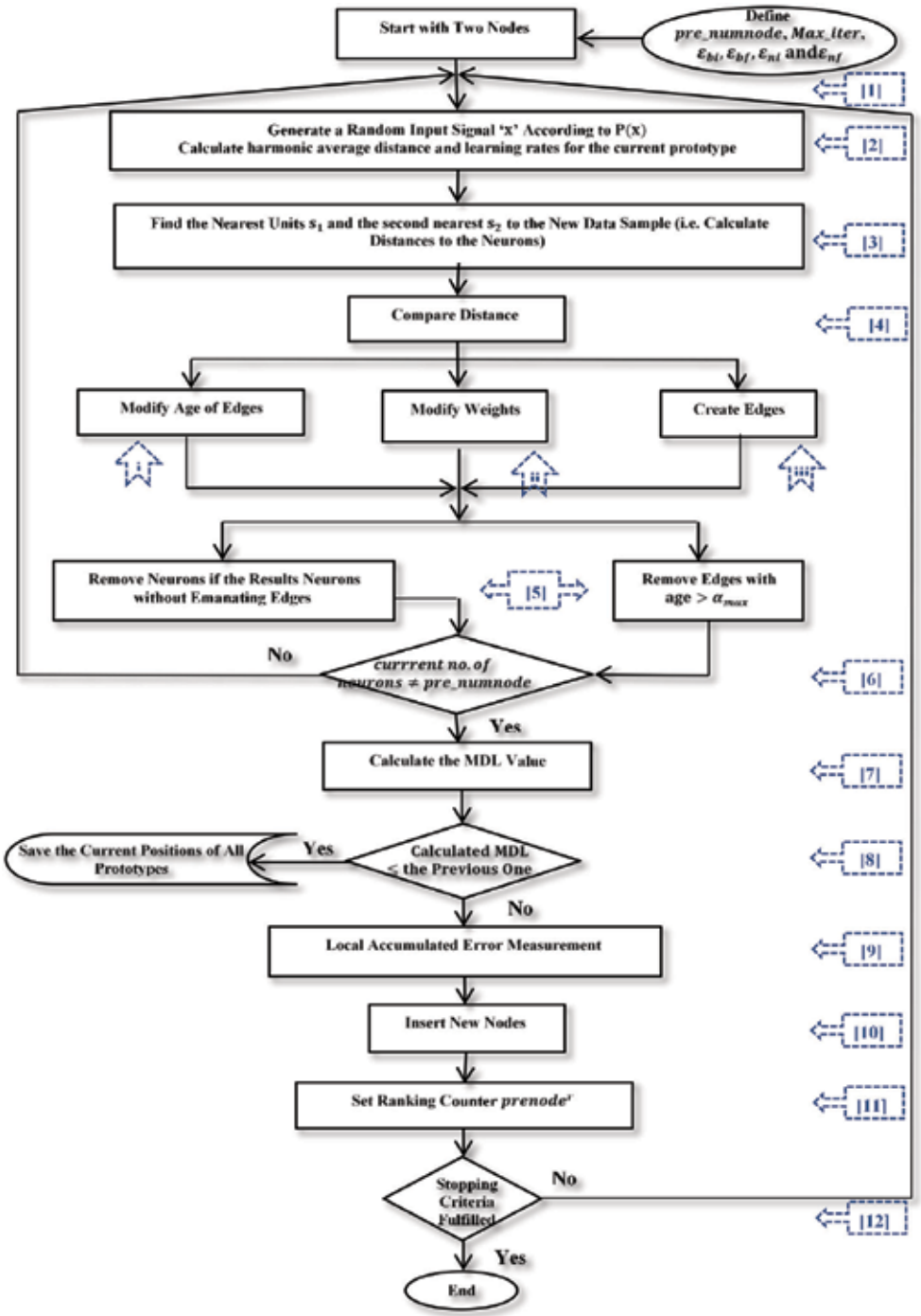

Figure 3. The flowchart of the RGNG algorithm.

3. Case studies

In the presented work, the performance of the NG, GNG, and RGNG algorithms on synthetic data are described. The cases studies are carried out to compare the performance of the three approaches. The experimental results on a public synthetic dataset are presented in the next section. Comparison of different neural networks and the need for such performance parameters using statistical evaluations has been recently highlighted by a number of researchers.

There are four parameters that are used in this work to evaluate the performance of the proposed clustering technique. These performance measures are: classification rate (CR), average partition quality (PQ), minimum cluster number (MCN), and mean square error (MSE). A robust clustering technique should be less sensitive to parameter configurations and give better performance under the same parameter settings in all experiments.

In the following experiments, the parameters are fixed for each technique with typical values suggested in literature. The RGNG technique was set with the typical values provided by Qin and Suganthan [29]: $\varepsilon_{bi} = 0.1$, $\varepsilon_{bf} = 0.01$, $\varepsilon_{ni} = 0.005$, $\varepsilon_{nf} = 0.0005$, $\alpha_{\max} = 100$, $k = 1.3$, $\eta = 1 \times 10^{-4}$. GNG and NG techniques were set with the typical values provided by Fritzke [24]: $\varepsilon_b = 0.05$, $\varepsilon_n = 0.006$, $\alpha_{\max} = 100$, $\beta = 0.0005$, $\lambda = 300$ for GNG; and $\varepsilon_i = 0.5$, $\varepsilon_f = 0.005$, $\lambda_i = 10$, $\lambda_f = 0.01$, $t_{\max} = 40000$ for NG network.

Each index of the performance measures is explained in the following sections.

3.1. Classification rate

This index refers to the classification rate (CR) for the whole dataset so that each data point is classified according to its nearest prototype. CR is based on using a majority voting classifier [32] by labeling all prototypes using a simple voting mechanism. According to the proposed technique, the numbers of prototypes are small, so the resulting CR will not be high.

3.2. Partition quality

This index refers to the average partition quality (PQ) measurement, which is averaged over all the independent runs in the experiments. PQ was defined by Hamerly and Elkan [33], as:

$$\mathrm{PQ} = \frac{\sum_{i=1}^{n_{cs}} \sum_{j=1}^{n_{cl}} p(i,j)^2}{\sum_{i=1}^{n_{cs}} p(i)^2} \tag{4}$$

where:

n_{cs}: true number of classes

m_{cl}: minimum number of clusters found by the technique

$p(i, j)$: probability of a point vector in cluster j belonging to the class i

$p(i)$: class probability

The number of classes n_{cs} should equal the actual number of clusters if each natural cluster is assumed to stand for an individual class. The minimum cluster number m_{ct} can be obtained by running the techniques.

The $p(i, j)$ term represents the frequency based on the probability that a data point is labeled by clusters i and j. The $p(i, j)$ quality is normalized by the sum of true probabilities; then, squared. This statistic is related to the rand statistic for comparing partitions [34]. The PQ index is maximized when the number of clusters m_{ct} is correctly detected and induces the same partition of n_{cs}, i.e., $m_{ct} = n_{cs}$, so that all points in each cluster are the same as those in one of the natural clusters.

3.3. Minimum cluster number

The minimum cluster number (MCN) is the average number of detected clusters by the techniques. The MCN indexes the ability of the techniques to find the underlying natural clusters. During the training of the techniques and according to the MCN value, only the proposed RGNG approach can find the actual number of clusters successfully.

During the growing process, this value is defined as the number of natural clusters in which the algorithm places at least one prototype when the number of prototypes in the network reaches the actual number of clusters. Cluster numbers detected by NG and GNG during the growing process deviate from the actual value of clusters when the number of prototypes is the same as the actual number of clusters.

3.4. Mean square error

Mean square error (MSE) is another criterion used for evaluating the performance of the proposed clustering technique. The MSE value represents the mean distance between the current nearest prototypes' positions resulting from the application of the techniques and the actual cluster centers.

The average MSE value in this experiment is higher for NG and GNG techniques than the RGNG technique. This indicates that the RGNG approach achieves the best accuracy with the strongest stability among the three approaches.

4. Experimental results with synthetic data

There are six different types of 2D synthetic datasets [29, 35] which are used in this work. They are snail, screw, ring, set3, set5, and set25 dataset. **Figures 4–6** show the plots of NG, GNG, and RGNG clustering with three types of 2D synthetic datasets (screw, set5, and snail) as an example. The number of neurons are selected randomly, $N = 7$, 10, and 12.

These figures cannot clearly differentiate between each method. Hence, four parameters are used in this work to evaluate the performance of the proposed clustering techniques: CR, PQ, MCN, and MSE introduced in the previous section. For the best comparison with RGNG, MDL criterion is added to NG and GNG techniques. The training results of these techniques

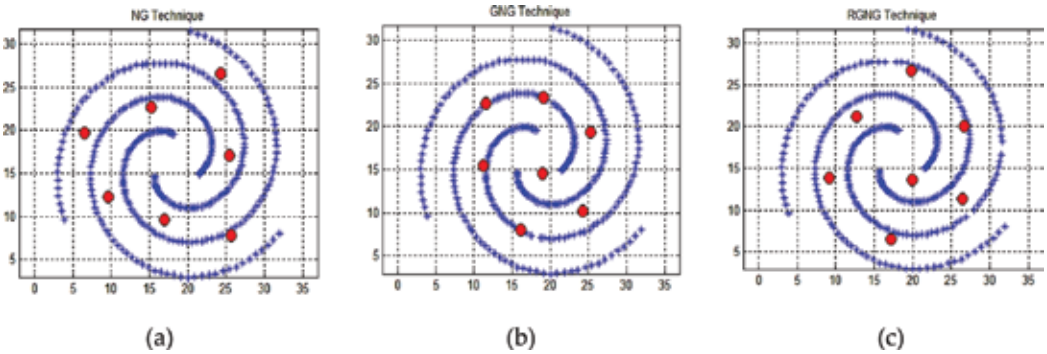

Figure 4. Clustering with screw synthetic dataset for $N = 7$, by running NG, GNG, and RGNG techniques.

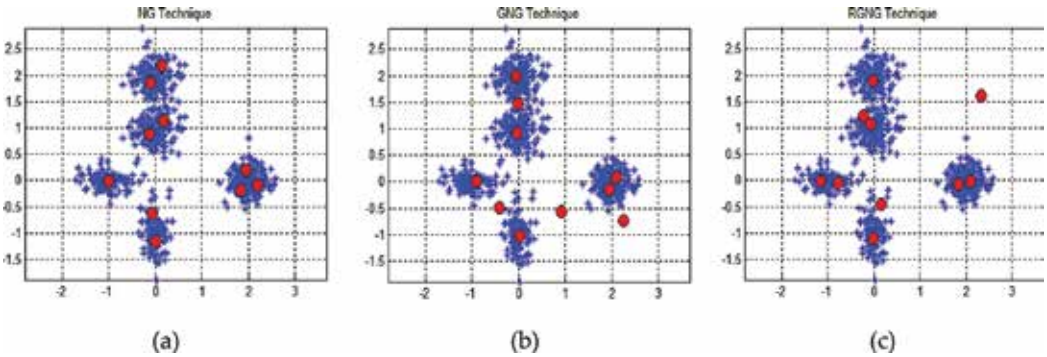

Figure 5. Clustering with set5 synthetic dataset for $N = 10$, by running NG, GNG, and RGNG techniques.

with synthetic data are shown in **Table 1**, where the number of neurons is chosen randomly as $N = 7$, 10, and 12.

According to literature [29, 36], the clustering output results introduced in **Table 1** clarified that RGNG approach is insensitive to different initializations and the presence of outliers. In

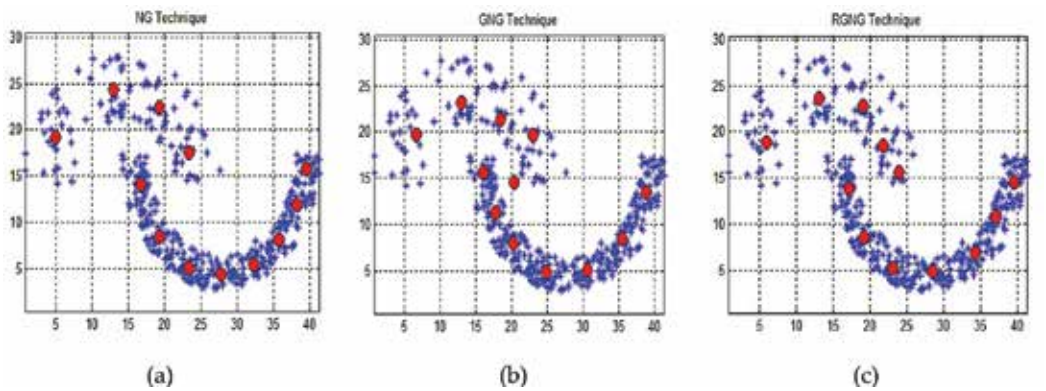

Figure 6. Clustering with snail synthetic dataset for $N = 12$, by running NG, GNG, and RGNG techniques.

Parameters	Number of neurons	NG	GNG	RGNG
CR	N = 7	0.8718	0.9686	0.9929
	N = 10	0.8514	0.9786	0.9843
	N = 12	0.8010	0.9647	0.9759
MCN	N = 7	9	8	7
	N = 10	12	11	10
	N = 12	15	14	12
PQ	N = 7	0.8990	0.9465	0.9869
	N = 10	0.8531	0.9288	0.9841
	N = 12	08279	0.9043	0.9807
MSE	N = 7	2.8032e+004	2.7608e+004	2.6493e+004
	N = 10	2.7913e+004	2.7378e+004	2.6351e+004
	N = 12	2.7703e+004	2.6940e+004	2.6188e+004

Table 1. Clustering results of synthetic data.

these techniques, the number of neurons used is small, so the CR values registered in the table are not high. In all the three clustering techniques, the number of neurons was equal to the actual cluster number. RGNG can effectively locate the actual number of clusters compared to the other two methods; NG and GNG fail with higher cluster numbers in the synthetic case.

The registered values of the MCN show that the number of detected prototypes or clusters in the RGNG technique is less than the others; which means that its ability to group data in actual number of clusters is better than the other two techniques. For example, when N is set to 10, the MCN value for RGNG is 10, which is less than that for NG and GNG values. The MCN value for running RGNG is equal to the number of neurons, 10, and has the same rate when compared with other N values; while the MCN value of running NG and GNG deviated from the actual cluster number.

Regarding the PQ value, it is noticed that the RGNG approach possesses higher PQ values than the NG and GNG techniques. For example, when N is set to 12, the PQ value for RGNG is 0.9807, which is higher than that of NG and GNG values. These high values of PQ indicate that the RGNG technique has a better partitioning quality with respect to the others, and finds more representative clusters.

Moreover, the RGNG method can find all the natural clusters during the growing stage with the correct number of prototypes. Hence, the MSE values are lower, which indicates that the RGNG technique has better robustness. For example, when N is set to 7, the MSE value for RGNG is 2.6493e + 004, which is lower than that for NG and GNG values. NG and GNG techniques may not detect all the actual clusters; hence, they yield higher MSE values.

The MDL value is one of the popular information theory evaluation measures that are used as clustering validity indexes [37]. The MDL criterion gives the ability of finding the optimal number of clusters and their center positions, corresponding to the smallest MDL value.

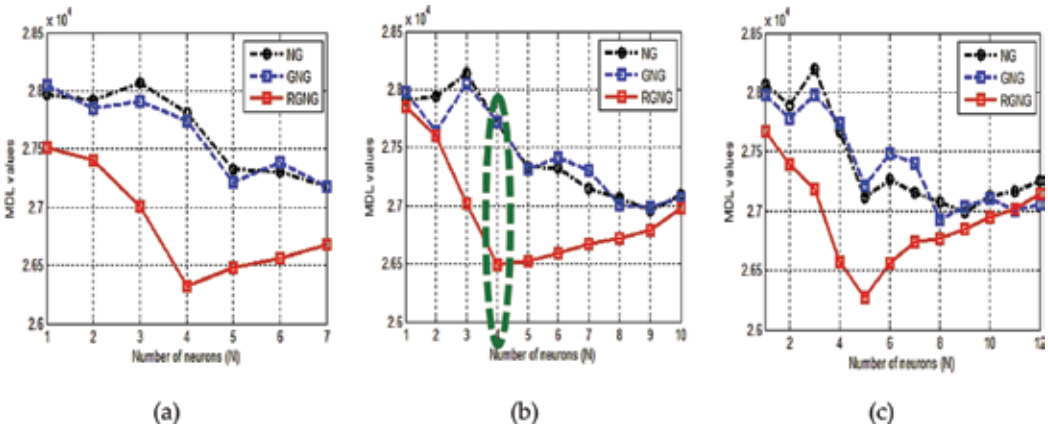

Figure 7. MDL values versus the number of clusters running the NG, GNG, and RGNG techniques on synthetic data, for: (a) $N = 7$; (b) $N = 10$; (c) $N = 12$.

The average MDL values during the growth stages are plotted versus the number of clusters or prototypes. **Figure 7** shows the curves for the NG and GNG techniques combined with the MDL criterion, as well as the RGNG approach on a synthetic dataset for different number of neurons, which are selected randomly as $N = 7$, 10, and 12. Each detected the cluster number corresponding to the MDL value.

In RGNG, the smallest MDL value was recorded on average with respect to NG and GNG combined with the MDL principle. For example, in **Figure 7 (b)**, the smallest MDL value is 2.65 that is obtained from running RGNG when N is equal to 4. While in the same $N = 4$, higher MDL value of 2.77 is recorded from running NG and GNG. From the presented figures, it is concluded that the proposed RGNG approach is insensitive to different initializations and the presence of outliers and can successfully find the actual number of clusters.

5. Prototype-based clustering package

The techniques introduced in this work are designed and implemented in a simple software package tool that allows users to interact with the clustering techniques and output data easily [13]. **Figure 8** shows the main window with the most important features of the designed prototype-based clustering software package.

1. *Selection data*: The user can select any one type of data from the different synthetic 2D datasets in the pop-up menu. Ring data is a 2D synthetic data selected as an example in **Figure 8**.

2. *Load data*: The selected data are loaded and all information related to the selected data ("Dimension," "Name," and "Type of Data") appear in the "info" window. The dimension

of the selected "Ring" data is 400x2 double. The selected data is plotted on sketch1 inside the main clustering window of **Figure 8**.

Figure 9 shows some of selected 2D synthetic datasets from the different datasets that were used in this work. Beside each plot, the information related to it is shown in the "info" window, in the left side of each plot.

3. *Selection technique*: The user can select one of the clustering techniques NG, GNG, or RGNG. The RGNG technique is selected as an example for the training in **Figure 8** with Ring data and $N = 18$, which is selected randomly.

Before clicking on "Apply NG," "Apply GNG," or "Apply RGNG" button, the training parameters related to each technique must be defined. As explained in Section 3, the training parameters must be set carefully within the limited range. The number of neurons (N) as well as the other parameters related to the selected technique must be defined. Another example of using the RGNG technique with Set3 dataset is shown in **Figure 10**. RGNG training parameters are set as the typical values in literature: $\varepsilon_{bi} = 0.1$, $\varepsilon_{bf} = 0.01$, $\varepsilon_{ni} = 0.005$, $\varepsilon_{nf} = 0.0005$, $\alpha_{\max} = 100$, $k = 1.3$, $\eta = 1 \times 10^{-4}$; the number of neurons (N) is chosen randomly as 14. When the algorithm's training is

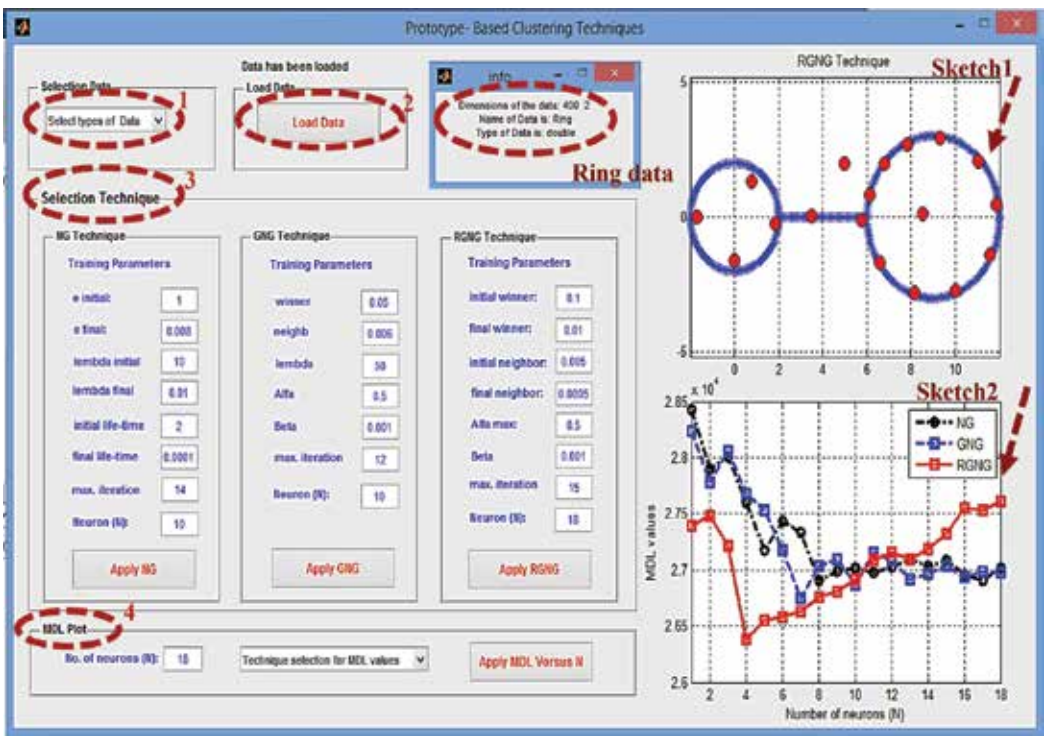

Figure 8. Main window of the prototype-based clustering software package.

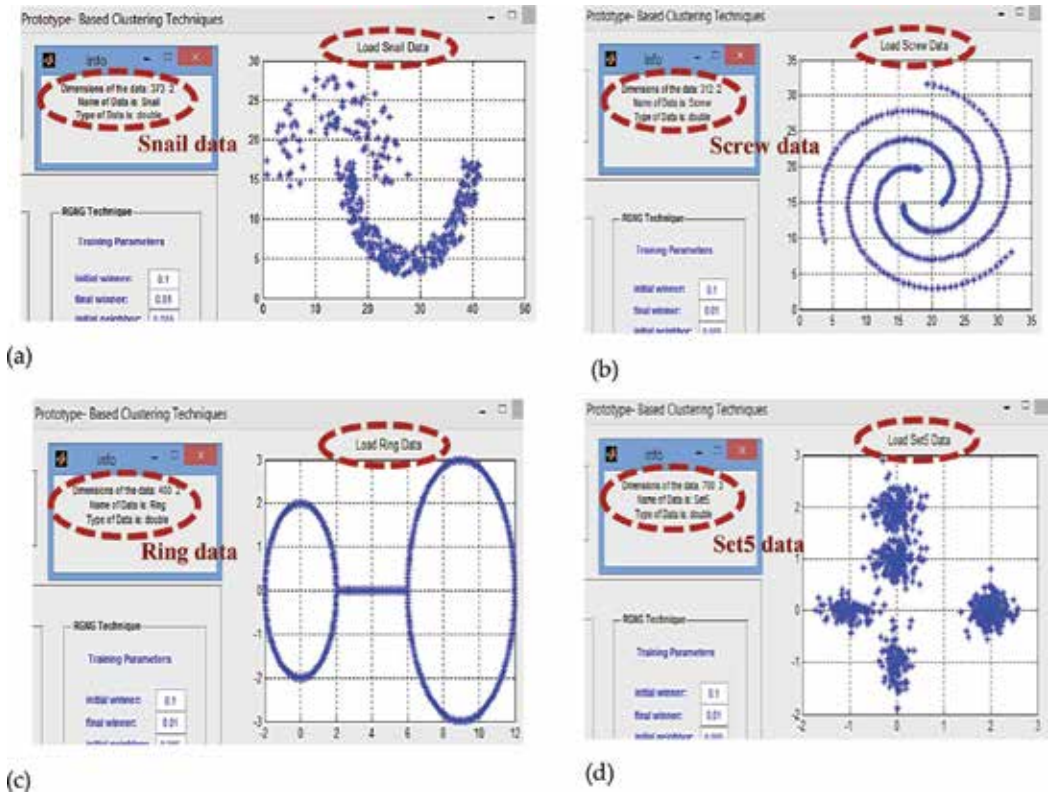

Figure 9. Different datasets with their information: (a) snail data; (b) screw data; (c) ring data; (d) Set5 data.

started, the program sketches the output running of the implemented technique as Sketch1. In Sketch1, a Set3 data is shown with firm red circles, which represent the actual cluster centers.

4. *MDL plot*: This panel is related to plotting MDL values versus the number of neurons (N) running the RGNG, GNG, and NG combined with MDL criterion. This panel includes three main buttons: "No. of neurons (N)," "Technique selection for MDL value," and "Apply MDL versus N" buttons, as shown in **Figure 11**.

After defining the number of neurons (N); one, two, or three of the training techniques have to be selected for comparing the MDL results. In the "Technique selection for MDL value" pop-up menu, there are seven selections—either show the result of each technique alone, two of them, or three of them for easy comparison. After clicking on the "Apply MDL versus N" button, the output results of MDL values are plotted with respect to the number of neurons (N) in Sketch2.

Figure 11 shows an example of the MDL plot, defining N = 16 and choosing "RGNG & GNG & NG" for comparing the results of the three techniques in Sketch2. For easy and best comparison between the MDL values of the three techniques, the output results sketch in the same figure.

http://dx.doi.org/10.5772/intechopen.74506

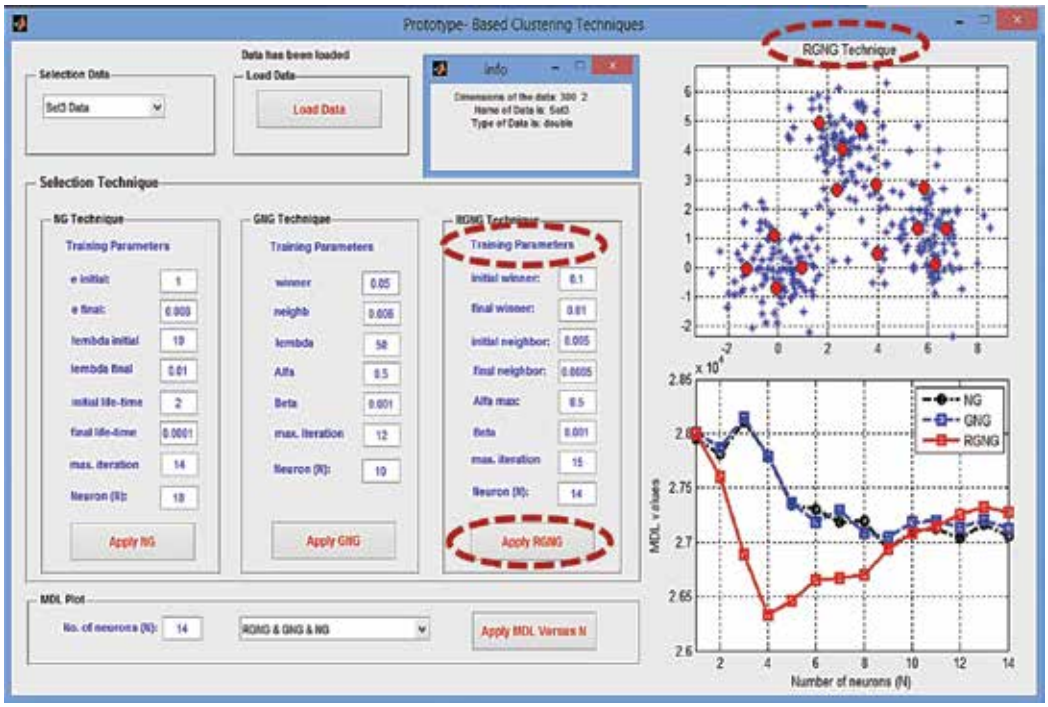

Figure 10. RGNG clustering with Set3 data ($N = 14$).

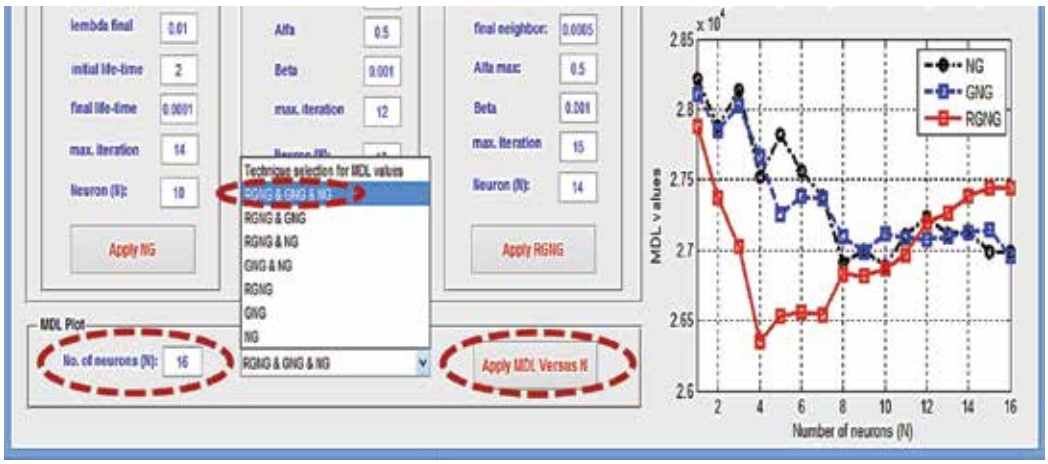

Figure 11. Comparison of MDL values for $N = 16$.

6. Conclusions

A simple user friendly software package is designed and implemented as an automatic clustering model for any dataset to use as part of the neural network course. NG, GNG, and RGNG algorithms are performed in the same package using a MATLAB-based graphical user

interface (GUI) tool. This visual tool lets the students/ researchers visualize the desired results using plots obtained with the click of a few buttons. The performance of these algorithms on 2D synthetic datasets is reported with respect to statistical estimations to explain the meaning of the output results. These results clarified that RGNG is better than NG and GNG when considering insensitivity to initialization as well as the presence of outliers. RGNG enhances GNG to be more robust toward noisy input dataset by using MDL criteria. Hence, RGNG solves the problem of finding the optimal number of clusters with respect to NG and GNG.

For future research directions, other unsupervised or supervised clustering algorithms may be used in the laboratory experiments. Another research direction is to apply the comparison among the three clustering algorithms to real multimodal datasets in medical applications. The package results could also be shared to websites using ASP .NET, which can give facility for users by sharing applications which requires no installation of MATLAB or any special program just a Web browser.

Author details

Hadeel K. Aljobouri[1,2]*, Hussain A. Jaber[2] and Ilyas Çankaya[2]

*Address all correspondence to: hadeel_bme77@yahoo.com

1 Biomedical Engineering Department, College of Engineering, Al-Nahrain University, Baghdad, Iraq

2 Electrical and Electronics Engineering Department, Graduate School of Natural Science, Ankara Yıldırım Beyazıt University, Ankara, Turkey

References

[1] Jain AK, Murty MN, Flynn PJ. Data clustering: A review. ACM Computing Surveys (CSUR). 1999;**31**(3):264-323

[2] Berkhin P. Survey of Clustering Data Mining Techniques. Technical Report. Accrue Software Inc.; 2002

[3] Kato N, Nemoto Y. Large scale handwritten character recognition system using subspace methods. Proceeding of IEEE International Conference on Systems, Man and Cybernetics, Beijing, China, 1996. pp. 1996432-1996437

[4] Ray S, Turi RH. Determination of number of clusters in k-means clustering and application in color image segmentation. Proceeding of the Fourth International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99), Calcutta, India, 1999. pp.137-143

[5] Bhatia SK, Deogun JS. Conceptual clustering in information retrieval. IEEE Transactions on System, Man, Cybernetics, Part B. 1998;**28**(3):427-436

[6] Ressom H, Wang D, Natarajan P. Adaptive double self-organizing maps for clustering gene expression profiles. Neural Networks. 2003;**16**(5-6):633-640

[7] Duda RO, Hart PE, Storck DG. Pattern Classification. New York: Wiley-Interscience; 2000

[8] Ripley BD. Pattern Recognition and Neural Networks. New York: Cambridge University Press; 1996

[9] Alziarjawey HA, Cankaya I. Heart rate monitoring and PQRST detection based on graphical user interface with Matlab. IJIEE. 2015;**5**:311-316

[10] Alziarjawey HAJ, Çamdalı Ü, Çankaya I, Aljobouri H. Design graphical user interface of linear algebra system package by using MATLAB. IJRITCC. 2016;**4**:428-433

[11] AlJobouri HK, Alziarjawey HA, Cankaya I. Biosignal Processing. Medical Imaging and fMRI (BSPMI) Software Package Based on MATLAB GUI for Education and Research. 2015;**1**:2380-8128

[12] Aljobouri HK, Çankaya I, Karal O. From biomedical signal processing techniques to fMRI Parcellation. Biosciences Biotechnology Research Asia. 2015;**12**(2):1115-1138

[13] AlJobouri HK, Jaber HA, Çankaya I. Performance evaluation of prototype-based clustering algorithms combined MDL index. Computer Applications in Engineering Education, Wiley Inc. 2017;**25**(4):642-654

[14] Martinetz T, Schulten KA. "Neural Gas" Network Learns Topologies. Artificial Neural Networks. Elsevier; 1991. pp. 397-402

[15] Kohonen T. Self-Organizing Maps. 3rd ed. Berlin: Springer; 2001

[16] Fernando C, Max C. Modification of the growing neural gas algorithm for cluster analysis. Progress in Pattern Recognition, Image Analysis and Applications, Lecture Notes in Computer Science, Springer. 2007;**4756**:684-693

[17] Curatelli F, Mayora-Iberra O. Competitive learning methods for efficient vector Quantizations in a speech recognition environment. MICAI 2000: Advances in artificial intelligence, lecture notes in computer science. Spring. 2000;**1793**:108-114

[18] Anastassia A, Alexandra P, José GR, Kenneth R. Automatic Landmarking of 2D medical shapes using the growing neural gas network. Computer vision for biomedical image applications, lecture notes in computer science. Spring. 2005;**3765**:210-219

[19] Atukorale AS, Downs T, Suganthan PN. Boosting the HONG network. Neurocomputing. 2003;**51**:75-86

[20] Winter M, Metta G, Sandini G. Neural-gas for Function Approximation: A heuristic for minimizing the local estimation error. Proceeding of International Joint Conference on Neural Network (IJCNN), Italy. 2000:535-538

[21] Haykin S. Neural Networks: A Comprehensive Foundation. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall; 1998

[22] Martinetz TM, Berkovich SG, Schulten KJ. "Neural gas" network for vector quantization and its application to time series prediction. IEEE Transactions on Neural Networks. 1993;**4**(4):558-569

[23] Bezdek JC, Keller JM, Krishnapuram R, Pal NR. Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Norwell, MA: Kluwer; 1999

[24] Fritzke B. Some Competitive Learning Methods (Draft). Technique Report, Institute for Neural Computation. Bochum: Ruhr-University; 1997

[25] Fritzke B. A Growing Neural Gas Network Learns Topologies. Advances in Neural Information Processing Systems 7, Cambridge: MIT Press; 1995:625-632

[26] Martinetz TM Competitive Hebbian learning rule forms perfectly topology preserving maps. Proceedings of International Conference on Artificial Neural Networks (ICANN93), Amsterdam, The Netherlands. 1993. pp. 427-434

[27] Fritzke B. Growing cells structures—A self-organizing network for unsupervised and supervised learning. Neural Networks. 1994;**7**(9):1441-1460

[28] Huber PJ. Robust Statistics. New York: Wiley; 1981

[29] Qin AK, Suganthan PN. Robust growing neural gas algorithm with application in cluster analysis. Neural Networks, Elsevier Ltd. 2004;**17**:1135-1148

[30] Tenmoto H, Kudo M, Shimbo M. MDL-based selection of the number of components in mixture models for pattern classification. Advance in pattern recognition. Lecture Notes in Computer Science. 1998;**1451**:831-836

[31] Zemel RS. A Minimum Description Length Framework for Unsupervised Learning. Ph.D. Thesis. University of Toronto; 1994

[32] Gareth J. Majority Vote Classifiers: Theory and Applications. Ph.D. dissertation submitted to Stanford University; 1998

[33] Hamerly G, Elkan C. Learning the k in k-means. Proceeding of 17th annual conference on neural information processing systems (NIPS2003). Canada; 2003

[34] Hubert L, Arabie P. Comparing partitions. Journal of Classification. 1985;**2**:193-218

[35] Retrieved 2015, from www.yarpiz.com. 2015

[36] Qin AK, Suganthan PN. Enhanced neural gas network for prototype-based clustering. Pattern Recognition, Elsevier Ltd. 2005;**38**:1275-1288

[37] Rissanen J. A universal prior for integers and estimation by minimum description length. Annals of Statistics. 1983;**11**:416-431

Chapter 11

New Approaches in Multi-View Clustering

Fanghua Ye, Zitai Chen, Hui Qian, Rui Li,
Chuan Chen and Zibin Zheng

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75598

Abstract

Many real-world datasets can be naturally described by multiple views. Due to this, multi-view learning has drawn much attention from both academia and industry. Compared to single-view learning, multi-view learning has demonstrated plenty of advantages. Clustering has long been serving as a critical technique in data mining and machine learning. Recently, multi-view clustering has achieved great success in various applications. To provide a comprehensive review of the typical multi-view clustering methods and their corresponding recent developments, this chapter summarizes five kinds of popular clustering methods and their multi-view learning versions, which include *k*-means, spectral clustering, matrix factorization, tensor decomposition, and deep learning. These clustering methods are the most widely employed algorithms for single-view data, and lots of efforts have been devoted to extending them for multi-view clustering. Besides, many other multi-view clustering methods can be unified into the frameworks of these five methods. To promote further research and development of multi-view clustering, some popular and open datasets are summarized in two categories. Furthermore, several open issues that deserve more exploration are pointed out in the end.

Keywords: clustering, multi-view clustering, multi-view *k*-means, multi-view spectral clustering, multi-view matrix factorization, tensor decomposition, deep learning

1. Introduction

Clustering is one of the most critical unsupervised learning techniques, which has been widely applied for data analysis, such as social network analysis, gene expression analysis, heterogeneous data analysis, and market analysis. The goal of clustering is to partition a dataset into several groups such that data samples in the same group are more similar than those in

different groups. Clustering plays an important role in mining the hidden patterns. However, most of the existing clustering algorithms are designed for single-view data.

With the rapid development of Internet and communication technology (ICT), the accesses to extract data are dramatically extended. That is, data can be collected from multiple sources or multiple facets. In such setting, each datum is associated with much richer information, which results in the requirement that to mine the intrinsic and valuable patterns hidden in the data, it is a necessity to take full advantage of the information contained in multiple sources. This issue is formally referred to as *multi-view learning*. To be more specific, each view corresponds to one source of information. For example, web pages can be described by both the page-contents (one view) and the hyperlink information (another view). Besides, different facets of a datum can also be treated as different views. For instance, an image can be characterized by its shape, color, and location.

Obviously, integrating the information contained in multiple views can bring great benefits for data clustering. The most straightforward way to utilize the information of all views is to concatenate the data features of each view together and then perform the traditional clustering methods such as *k*-means. However, such a method lacks the ability to distinguish the different significance of different views. That is, the important views and less important views are treated equally, which may degrade the ultimate performance severely. To take better advantage of the multi-view information, the ideal approach is to simultaneously perform the clustering using each view of data features and integrate their results based on their importance to the clustering task. Formally, this approach is known as *multi-view clustering*.

As an emerging and effective paradigm in data mining and machine learning, multi-view clustering refers to the clustering of the same class of data samples with multi-view representations, either from various information sources or from different feature generators. It is clear that if the clustering method cannot cope appropriately with multi-views, these views may even degrade the performance of multi-view clustering. To make use of multi-view information to improve clustering results, there are two main challenges to overcome. The first one is how to naturally ensemble the multiple clustering results of all the views. The second one is how to learn the importance of different views to the clustering task. In addition, these two issues should be figured out simultaneously. Thus, to achieve these goals, new clustering objective function should be designed, followed by the new solving method.

Multi-view clustering was first studied by Bickel and Scheffer [1] in 2004. They extended the classic *k*-means and expectation maximization (EM) clustering methods to the multi-view environment to deal with text data with two conditionally independent views. Based on this seminal work, a variety of multi-view clustering methods have been proposed over the past two decades [2–4]. Since covering all the proposed methods in one chapter is hard, to provide a comprehensive review of the typical multi-view clustering methods and their corresponding recent developments, we summarize five kinds of popular clustering methods and their multi-view learning versions, which include *k*-means, spectral clustering, matrix factorization, tensor decomposition, and deep learning. This is based on the consideration that these clustering methods are the most widely employed algorithms for single-view data, and lots of efforts have been devoted to extending them for multi-view clustering. Besides, many other multi-

view clustering methods can be unified into the frameworks of these five methods. Therefore, when readers become familiar with these five multi-view clustering methods, they can capture the core ideas of other multi-view clustering methods easily. This chapter is self-contained, which follows a line of introduction from the preliminaries of these clustering methods for single-view data to their variant forms for multi-view clustering.

The remainder of this chapter is organized as follows. Section 2 describes the benefits of multi-view clustering. Section 3 details the aforementioned five multi-view clustering methods. Section 4 summarizes two kinds of popular open datasets. Several open issues are illustrated in Section 5. Section 6 concludes this chapter.

2. Benefits of multi-view clustering

Compared with the clustering methods that are implemented on single-view data, multi-view clustering is expected to obtain more robust and novel partitioning results by exploiting the redundant and complementary information in different views [5], as stated in the following sections.

2.1. Benefit one: accurate description of data

It is obvious that single-view data may contain incomplete knowledge, while multi-view data usually contains complementary and redundant information, which results in a more accurate description of the data. For example, it may fail to identify the intrinsic community structures of a social network via just leveraging the friendships. However, if more information such as users' demographics can be obtained, it is more inclined to find out the implicit relationships between users.

2.2. Benefit two: reducing noises of data

Even when the information contained in single-view data is complete, there may exist some unavoidable noises. It is apparent that data cleaning is one critical issue in data analysis, which can tremendously affect the performance of clustering algorithms. It is quite hard and costly to remove all the noises of data, and thus single-view noisy data usually leads to unsatisfactory clustering results. On the other hand, multi-view clustering is able to circumvent the side effect of noises or corrupted data in each view and emphasize the common patterns shared by multi-view data.

2.3. Benefit three: wider range of applications

There is no doubt that all the multi-view clustering methods can be applied to single-view data. However, many clustering tasks are impossible to implement by single-view clustering due to its limitations. For example, data with multiple modalities is becoming more and more common and heterogeneous information networks are gaining increasing popularity as well.

These types of data naturally fit into multi-view learning, while cannot be settled by single-view learning methods appropriately. In all, the complementary property among multi-view data can overcome the limitations of single-view data and expand their application areas.

3. Multi-view clustering methods

Due to the widespread use of multi-view datasets in practice, many realistic applications are accomplished by multi-view learning methods, such as community detection in social networks, image annotation in computer vision, and cross-domain user modeling in recommendation systems [6]. Meanwhile, based on the seminal work of Bickel and Scheffer [1], plenty of multi-view clustering methods have been proposed [2, 3, 5]. As explained in Section 1, this chapter seeks to review five kinds of typical clustering methods and their multi-view versions, which include *k*-means, spectral clustering, matrix factorization, tensor decomposition, and deep learning. All of these five methods are popular methods for single-view clustering. Although there are some other multi-view clustering methods not contained in this chapter, such as the canonical correlation analysis (CCA)-based multi-view clustering methods [7], the DBSCAN-based multi-view clustering methods [8], and the lower dimensional subspace-based multi-view clustering methods [9], most of them can be unified into the frameworks of these five involved methods. For instance, a pair-wise sparse subspace representation model for multi-view clustering proposed in [10] can be unified into the framework of matrix factorization.

3.1. Multi-view clustering via *k*-means

k-means is one of the most popular clustering algorithms with a history of more than 50 years [11]. Except for its simplicity, *k*-means has a good potential to deal with large-scale datasets. Owing to these properties, *k*-means has been successfully used in various topics, including computer vision, social network analysis, and market segmentation, to name but a few. Although it has been studied deeply over the past few decades, many variants of *k*-means are put forward continuously [12–15].

3.1.1. Preliminaries of k-means

As a classic clustering algorithm, *k*-means employs K prototype vectors (i.e., centers or centroids of the K clusters) to characterize each data sample and minimizes a sum of squared loss function to find these prototypes. Consider a dataset denoted by $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{M \times N}$, where $\mathbf{x}_i \in \mathbb{R}^M$ represents the attribute (feature) vector of the i-th data sample $\mathbf{x}_i$. In order to partition the dataset $\mathbf{X}$ into K disjoint clusters, denoted by $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$, *k*-means tries to optimize the following objective function:

$$\varepsilon = \sum_{i=1}^{N} \sum_{k=1}^{K} \delta_{ik} \|\mathbf{x}_i - \mathbf{v}_k\|_2^2, \quad \mathbf{v}_k = \frac{\sum_{i=1}^{N} \delta_{ik} \mathbf{x}_i}{\sum_{i=1}^{N} \delta_{ik}} = \frac{1}{|C_k|} \sum_{\mathbf{x} \in C_k} \mathbf{x}, \tag{1}$$

where δ_{ik} is an indicator variable with $\delta_{ik} = 1$ if $\mathbf{x}_i \in C_k$ and 0 otherwise and $\mathbf{v}_k$ is the k-th prototype vector, i.e., the k-th cluster center.

As can be seen, Eq. (1) adopts the Euclidean distance to measure the similarities between data samples. However, there are many data structures or data distributions in real world. Thus, it is not always suitable to apply this basic form of *k*-means to accurately identify the hidden patterns of datasets. What is more, some datasets may be not separable in the low-dimensional space. Recently, kernel method has been of wide concern in the field of machine learning. By introducing a kernel function, the original nonlinear datasets are mapped to a higher dimensional reproducing kernel Hilbert space. In the new space, the datasets become linearly separable. For this reason, the kernel *k*-means algorithm [16, 17] has been proposed. It is just a generalization of the standard *k*-means algorithm and has the following objective function:

$$\varepsilon = \sum_{i=1}^{N}\sum_{k=1}^{K}\delta_{ik}\|\phi(\mathbf{x}_i) - \mathbf{v}'_k\|_2^2, \quad \mathbf{v}'_k = \frac{\sum_{i=1}^{N}\delta_{ik}\phi(\mathbf{x}_i)}{\sum_{i=1}^{N}\delta_{ik}}, \tag{2}$$

where $\phi : \mathbf{X} \to \mathbf{H}$ is a nonlinear transformation function. Define a kernel function $\mathcal{K} : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$ with $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Then, Eq. (2) can be rewritten into the kernel form as below:

$$\varepsilon = \sum_{i=1}^{N}\sum_{k=1}^{K}\delta_{ik}\left(\mathcal{K}(\mathbf{x}_i, \mathbf{x}_i) - 2\frac{\sum_{j=1}^{N}\delta_{jk}\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j=1}^{N}\delta_{jk}} + \frac{\sum_{j=1}^{N}\sum_{l=1}^{N}\delta_{jk}\delta_{lk}\mathcal{K}(\mathbf{x}_j, \mathbf{x}_l)}{\sum_{j=1}^{N}\sum_{l=1}^{N}\delta_{jk}\delta_{lk}}\right). \tag{3}$$

With the aid of the kernel function, there is no need to explicitly provide the transformation function ϕ. This is because, for certain kernel function, the corresponding transformation function is intractable. However, the inner products in the kernel space can be easily obtained according to the kernel function.

3.1.2. Basic form of multi-view k-means

Both the *k*-means and the kernel *k*-means described above are designed for single-view data. To solve the multi-view clustering problem, some new objective functions should be developed. Assume that there are V views in total. Let $\mathcal{X} = \left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(V)}\right\}$ denote the data of all the views. It is obvious that different views should have different contributions according to their conveyed information. To achieve this goal, it is straightforward to modify the standard *k*-means to make it applicable in the multi-view environment with a new objective function as follows:

$$\varepsilon = \sum_{v=1}^{V}\mu_v^{\gamma}\varepsilon_v, \text{ s.t.}\mu_v \geq 0, \sum_{v=1}^{V}\mu_v = 1, \gamma > 1, \tag{4}$$

where μ_v is the weight factor for the v-th view, γ is a parameter used to control the weight distribution, and ε_v corresponds to the objective function (i.e., loss function) of the v-th view:

$$\varepsilon_v = \sum_{i=1}^{N}\sum_{k=1}^{K}\delta_{ik}\|\mathbf{x}_i^{(v)} - \mathbf{v}_k^{(v)}\|_2^2, \mathbf{v}_k^{(v)} = \frac{\sum_{i=1}^{N}\delta_{ik}\mathbf{x}_i^{(v)}}{\sum_{i=1}^{N}\delta_{ik}}. \tag{5}$$

Similarly, the objective function of the multi-view kernel *k*-means can be obtained, which is omitted here. Note that finding the optimal solution of Eq. (4) is an NP-hard problem; thus,

some iterative algorithms are developed according to the greedy strategy. One basic iterative algorithm works in a two-stage manner: (1) updating the clustering for given weights and (2) updating the weights for given clusters; see [18] for details.

Denote $\|\mathbf{X}\|_F$ as the Frobenius norm of a given matrix $\mathbf{X}$, i.e., $\|\mathbf{X}\|_F = \sqrt{\sum_{i,j=1}^{N} x_{ij}^2}$. Then, Eq. (4) can be easily transformed into a matrix form as shown in the following:

$$\min_{\mathbf{V}^{(v)},\mathbf{U},\mu_v} \sum_{v=1}^{V} \mu_v^{\gamma} \|\mathbf{X}^{(v)} - \mathbf{V}^{(v)}\mathbf{U}^T\|_F^2, \text{s.t.} u_{ik} \in \{0,1\}, \sum_{k=1}^{K} u_{ik} = 1, \mu_v \geq 0, \sum_{v=1}^{V} \mu_v = 1, \gamma > 1, \tag{6}$$

where $\mathbf{V}^{(v)} \in \mathbb{R}^{M_v \times K}$ denotes the centroid matrix for the v-th view and $\mathbf{U} \in \mathbb{R}^{N \times K}$ denotes the clustering indicator matrix with the (i,k) element being δ_{ik}. Note that all the views share a common clustering indicator matrix $\mathbf{U}$.

3.1.3. Variants of multi-view k-means

The basic formulations of multi-view *k*-means shown in Eqs. (4) and (6) do have some drawbacks. For example, it assumes that all the views are sharing a common clustering indicator matrix $\mathbf{U}$. However, the structure information contained may be very limited or even lost in some views. In such case, the performance will be severely affected if all the views share a common clustering indicator matrix. To tackle the issues, many variants of multi-view *k*-means clustering algorithms have been proposed in recent years. Instead of the ℓ_2-norm, the structured sparsity-inducing norm, i.e., the $\ell_{2,1}$-norm, is adopted to strengthen the basic multi-view *k*-means, in the hope that the effect of outlier data samples will be reduced [19]. In [20], a *k*-means-based dual-regularized multi-view outlier detection method (DMOD) is proposed to identify the cluster outliers and the attribute outliers simultaneously, which is based on a novel cross-view outlier measurement criterion. Moreover, in the DMOD model, each view is associated with a particular clustering indicator matrix, and another alignment matrix is introduced to enforce the consistency between different views. An automated two-level variable weighting clustering algorithm, called TW-*k*-means, is developed in [21]. TW-*k*-means is able to compute weights for each view and each individual attribute simultaneously. More specifically, in this algorithm, to identify the compactness of the view, a view weight is assigned to each view, and an attribute weight is assigned to each attribute in the view to identify the importance of the attribute. Both view weights and attribute weights are employed in the distance function to determine the cluster structures of data samples. Similar strategies have also been taken in [22, 23] to learn more robust multi-view *k*-means models.

As aforementioned, it is NP-hard to find the optimal solution of the multi-view *k*-means clustering problem. The greedy iterative algorithm has a high risk of getting stuck in local optima during the optimization. Recently, the self-paced learning has been used to alleviate this problem. The general self-paced learning model consists of a weighted loss function on all data samples and a regularizer term imposed on the weights of data samples. By gradually increasing the penalty on the regularizer, more data samples are automatically added into consideration from "easy" to "complex" via a pure self-paced approach. In this, Xu et al. [24]

present a new multi-view self-paced learning (MSPL) algorithm for clustering based on multi-view *k*-means. MSPL learns the multi-view model by not only progressing from "easy" to "complex" data samples but also from "easy" to "complex" views. The objective function of MSPL is quite succinct, which is shown in Eq. (7).

$$\min_{\mathbf{V}^{(v)},\mathbf{U},u} \sum_{v=1}^{V} \|\left(\mathbf{X}^{(v)} - \mathbf{V}^{(v)}\mathbf{U}^T\right)\mathrm{diag}\left(\sqrt{\mu^{(v)}}\right)\|_F^2 + f(\mathcal{U}), \text{s.t.} u_{ik} \in \{0,1\}, \sum_{k=1}^{K} u_{ik} = 1, \tag{7}$$

where $\mu^{(v)} = \left[\mu_1^{(v)}, \mu_2^{(v)}, \ldots, \mu_N^{(v)}\right] \in [0,1]^N$ denotes the weights of data samples in the v-th view, $\mathcal{U} = \left[\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(V)}\right]$, and $f(\mathcal{U})$ denotes the regularization term on demand.

3.2. Multi-view clustering via spectral clustering

Spectral clustering is built upon the spectral graph theory. In recent years, spectral clustering has become one of the most popular clustering algorithms and shown its effectiveness in various real-world applications ranging from statistics, computer sciences to bioinformatics. Due to its adaptation in data distribution, spectral clustering often outperforms traditional clustering algorithms such as *k*-means. In addition, spectral clustering is simple to implement and can be solved efficiently by standard linear algebra.

3.2.1. Preliminaries of spectral clustering

Spectral clustering is closely related to the minimum cut problem of graphs. It first performs dimensionality reduction on the original data space by leveraging the spectrum of the similarity matrix of data samples and then performs *k*-means on the low-dimensional space to partition data into different clusters. Therefore, for a set of data samples, a similarity matrix should be constructed at first. Typically, each data sample is treated as a node of a graph and each relationship between data samples is regarded as an edge in the graph. Besides, each edge is associated with a weight. It is obvious that the value of the edge weight between two far-away data samples should be low and the value between two close data samples should be high. For a given dataset $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{M \times N}$, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$ be the generated undirected weighted graph, where $\mathcal{V}$ denotes the set of nodes representing the data samples and $\mathcal{E}$ denotes the set of edges representing the relationships between data samples. The similarity matrix $\mathbf{S}$ is a symmetric matrix with each element s_{ij} representing the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. There are three popular approaches to construct graph $\mathcal{G}$, that is, the ε-neighborhood graph, the *k*-nearest neighbor graph, and the fully connected graph (see details in [25]). To partition $\mathcal{G}$ into disjoint subgraphs (clusters), the minimum cut problem requires that the edge weights across different clusters are as small as possible, while the total edge weights within each cluster are as high as possible.

According to the above graph cut theory, two popular versions of spectral clustering are developed, i.e., the ratio cut (RatioCut) and the normalized cut (Ncut). The classical relaxed form of the RatioCut [26] is shown as below:

$$\min\ tr\left(\mathbf{U}^T\mathbf{L}\mathbf{U}\right), \text{s.t.}\mathbf{U}^T\mathbf{U} = \mathbf{I}, \tag{8}$$

where tr computes the trace of a matrix, $\mathbf{U} \in \mathbb{R}^{N\times K}$ is the clustering indicator matrix, $\mathbf{I}$ is an identity matrix, and $\mathbf{L}$ is the graph Laplacian matrix defined as $\mathbf{L} = \mathbf{D} - \mathbf{S}$. Here, $\mathbf{D}$ is a diagonal matrix with $d_{ii} = \sum_{j=1}^{N} s_{ij}$. The objective function of Ncut [27] is similar to Eq. (8) by replacing $\mathbf{L}$ by the normalized Laplacian matrix $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$. Both RatioCut and NCut can be solved efficiently by the eigenvalue decomposition (EVD) of $\mathbf{L}$ or $\tilde{\mathbf{L}}$.

3.2.2. Basic form of multi-view spectral clustering

Multi-view spectral clustering is able to learn the latent cluster structures by fusing the information contained in multiple graphs. Similar to multi-view k-means, it is not hard to extend the basic spectral clustering to the multi-view environment. Given a dataset $\mathcal{X} = \left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(\mathbf{V})}\right\}$ with V views, V graphs $\left\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(\mathbf{V})}\right\}$ and the corresponding Laplacian matrices $\left\{\mathbf{L}^{(1)}, \mathbf{L}^{(2)}, \ldots, \mathbf{L}^{(\mathbf{V})}\right\}$ can be constructed.

Kumar et al. [28] firstly present a multi-view spectral clustering approach, which has a flavor of co-training idea widely used in semi-supervised learning. It follows the consistency of multi-view learning that each view gives the same labels for all data samples. So it can use the eigenvector of one view to "label" another view and vice versa. For example, via computing two views' eigenvectors, say $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$, the clustering result of $\mathbf{U}^{(1)}$ can be used to modify the graph similarity matrix $\mathbf{S}^{(2)}$, and then the clustering result of $\mathbf{U}^{(2)}$ can be used to modify the graph similarity matrix $\mathbf{S}^{(1)}$. For more than two views, the same strategy can be applied. Kumar et al. [29] further propose a multi-view spectral clustering approach using co-regularization idea that makes the clustering results of different views agree with each other. The co-regularization form is stated as the disagreement between clustering results of two views: $\Phi\left(\mathbf{U}^{(p)}, \mathbf{U}^{(q)}\right) = -tr\left(\mathbf{U}^{(p)}\mathbf{U}^{(p)T}\mathbf{U}^{(q)}\mathbf{U}^{(q)T}\right)$. Then the goal is to minimize the disagreement to achieve the consistency between views with the following objective function:

$$\min\sum_{v=1}^{V} tr\left(\mathbf{U}^{(v)T}\mathbf{L}^{(v)}\mathbf{U}^{(v)}\right) - \sum_{p,\,q=1}^{V}\lambda_{pq}tr\left(\mathbf{U}^{(p)}\mathbf{U}^{(p)T}\mathbf{U}^{(q)}\mathbf{U}^{(q)T}\right), \text{s.t.}\mathbf{U}^{(v)T}\mathbf{U}^{(v)} = \mathbf{I}, \tag{9}$$

where λ_{pq} represents the degree of disagreement between the p-th view and the q-th view. From another perspective, all the views sharing a common indicator matrix $\mathbf{U}^*$ is also rational according to the consistency requirement. So the model in Eq. (9) can be rewritten as

$$\min\sum_{v=1}^{V} tr\left(\mathbf{U}^{(v)T}\mathbf{L}^{(v)}\mathbf{U}^{(v)}\right) - \sum_{v=1}^{V}\lambda_{v}tr\left(\mathbf{U}^{(v)}\mathbf{U}^{(v)T}\mathbf{U}^{*}\mathbf{U}^{*T}\right), \text{s.t.}\mathbf{U}^{(v)T}\mathbf{U}^{(v)} = \mathbf{I}, \tag{10}$$

where λ_v controls the degree of disagreement between $\mathbf{U}^{(v)}$ and $\mathbf{U}^*$.

3.2.3. Variants of multi-view spectral clustering

The basic form of multi-view spectral clustering achieves the basic goals of multi-view learning. However, some issues have not yet been considered. For instance, the weight parameter λ in Eq. (9) needs to be set manually. To make up this issue, it is necessary to adaptively compute the weight of each view. Xia et al. [30] assume that each view has a weight μ_v representing its importance and the weight distribution should be sufficiently smooth. They further consider a unified indicator matrix $\mathbf{U}$ across all views, which can be fulfilled via exploring the complementary property of different views. To this end, they develop a novel model as follows:

$$\min \sum_{v=1}^{V} \mu_v^{\gamma} tr\left(\mathbf{U}^T \mathbf{L}^{(v)} \mathbf{U}\right), \text{s.t.} \sum_{v=1}^{V} \mu_v = 1, \mu_v > 0. \tag{11}$$

The model above needs a manually specified parameter γ to adjust the weights of different views, which is sometimes intractable. Thus, Nie et al. [31] propose a parameter-free auto-weighted multiple graph learning method (AMGL), wherein the weight parameter μ_v is replaced by $\alpha_v = \frac{1}{2}\sqrt{tr\left(\mathbf{U}^T \mathbf{L}^{(v)} \mathbf{U}\right)}$. Thus, AMGL does not require additional parameters, and α_v can be self-updated. To avoid the considerable noise in each view which often degrades the performance severely, Xia et al. [32] propose a robust multi-view spectral clustering (RMSC) method via low-rank and sparse decomposition. In RMSC, a novel Markov chain is designed for dealing with the noise. First, the similarity matrix $\mathbf{S}^{(v)}$ and the corresponding transition probability matrix $\mathbf{P}^{(v)} = \left(\mathbf{D}^{(v)}\right)^{-1} \mathbf{S}^{(v)}$ are computed. Then, the row-rank latent transition probability matrix $\widehat{\mathbf{P}}$ and the deviation error matrix $\mathbf{E}^{(v)}$ are constructed via low-rank and sparse decomposition. Finally, based on the transition probability matrix $\widehat{\mathbf{P}}$, the standard Markov chain method is applied for partitioning data into K clusters. Note that the methods above have a high cost in optimization computation. There are numerous variables that need to be updated and the derivation process is also extremely complex during the optimization. To overcome this limitation, Chen et al. [33] present a novel variant of the Laplacian matrix named block intra-normalized Laplacian defined as follows, without the linear combination of multiple Laplacian matrices.

$$\mathbf{B} = \mathbf{B}_w + \beta \mathbf{B}_a = \begin{pmatrix} \mathbf{L}^{(1)} & 0 & \cdots & 0 \\ 0 & \mathbf{L}^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{L}^{(V)} \end{pmatrix} + \beta \begin{pmatrix} (V-1)\mathbf{I} & -\mathbf{I} & \cdots & -\mathbf{I} \\ -\mathbf{I} & (V-1)\mathbf{I} & \cdots & -\mathbf{I} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{I} & -\mathbf{I} & \cdots & (V-1)\mathbf{I} \end{pmatrix}, \tag{12}$$

where $\mathbf{B}_w$ denotes the within Laplacian matrix of V views and $\mathbf{B}_a$ denotes the across Laplacian matrix between different views. Based on $\mathbf{B}$, the block intra-normalized Laplacian matrix is then defined as $\widehat{\mathbf{B}} = \mathbf{D}^{-1/2} \mathbf{B}_w \mathbf{D}^{-1/2} + \beta \mathbf{B}_a$, where $\mathbf{D}$ is a block diagonal matrix with the v-th block being $\mathbf{D}^{(v)}$. By proving that the multiplicity of the zero eigenvalue of the constructed block Laplacian matrix is equal to the number of clusters K, the eigenvectors of the block

Laplacian matrix can be used for clustering via the classical form of spectral clustering. At the end, the lower and upper bounds of the optimal solution are also established. See [33] for more details.

3.3. Multi-view clustering via matrix factorization

In the fields of data mining and machine learning, matrix factorization (MF) is an effective latent factor learning model. Given a data matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$, MF tries to find two low-rank factor matrices $\mathbf{V} \in \mathbb{R}^{M \times K}$ and $\mathbf{U} \in \mathbb{R}^{N \times K}$ whose multiplication can well approximate it, i.e., $\mathbf{X} \approx \mathbf{V}\mathbf{U}^T$. MF has shown many promising applications in real world, such as information retrieval, recommendation system, signal processing, document analysis, and so on. Usually, the nonnegativity constraints are enforced to the factor matrices to promote the interpretability of the MF models. Therefore, in this part, we focus on the introduction of the nonnegative MF (NMF)-related clustering models. For a comprehensive review of NMF-based models and applications, please refer to [34].

3.3.1. Preliminaries of matrix factorization

As is well known, there are many matrix factorization models, including the singular value decomposition, Cholesky decomposition, LU decomposition, QR decomposition, and Schur decomposition. These factorization models either have too strict restrictions on the factor matrices or lack the ability to be applied to data analysis. Due to the wide applications of NMF in recommending systems, NMF has drawn much attention in both academia and industry. In fact, NMF can be regarded as an extension of the standard k-means algorithm by relaxing the constraints imposed on the clustering indicator matrix. For a given dataset $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}_+^{M \times N}$, NMF seeks to learn a basis matrix $\mathbf{V}$ and a coefficient matrix $\mathbf{U}$ via optimizing the following objective function:

$$\min_{\mathbf{V},\,\mathbf{U}} \|\mathbf{X} - \mathbf{V}\mathbf{U}^T\|_F^2, \text{ s.t.} \mathbf{V} \geq 0, \mathbf{U} \geq 0, \tag{13}$$

where $\mathbf{V} \in \mathbb{R}_+^{M \times K}$ can be considered as the cluster centroid matrix and $\mathbf{U} \in \mathbb{R}_+^{N \times K}$ can be treated as a "soft" clustering indicator matrix. The objective function above is not convex in $\mathbf{U}$ and $\mathbf{V}$; therefore, it is impractical to find the global optima. Typically, there are two methods to solve Eq. (13). The first one is the gradient descent method [35]. The other one is the multiplicative method [36] where the iterative updating rules are as follows:

$$\mathbf{V} \leftarrow \mathbf{V} \odot \frac{\mathbf{X}\mathbf{U}}{\mathbf{V}\mathbf{U}^T\mathbf{U}}, \quad \mathbf{U} \leftarrow \mathbf{U} \odot \frac{\mathbf{X}^T\mathbf{V}}{\mathbf{U}\mathbf{V}^T\mathbf{V}}, \tag{14}$$

where $\odot$ and $\frac{[\cdot]}{[\cdot]}$ denote the element-wise multiplication and division, respectively. It is noteworthy that there are many other criterions to measure the difference between $\mathbf{X}$ and $\mathbf{V}\mathbf{U}^T$, such as the ℓ_1-norm, the $\ell_{2,1}$-norm, and the Kullback-Leibler divergence (a.k.a. relative entropy). For these criterions, the updating rules can be derived similarly.

3.3.2. Basic form of multi-view matrix factorization

The hypothesis behind multi-view clustering is that different views should admit the same underlying clustering structures of the datasets. That is, the coefficient matrices learned from different views should be as consistent as possible. To this end, a soft regularization term is introduced to enforce the coefficient matrices of different views toward a common consensus [37]. For a given dataset $\mathfrak{X} = \left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(V)}\right\}$ with V views, the following objective function can be derived to partition $\mathfrak{X}$ into K clusters:

$$\min_{\mathbf{V}^{(v)},\, \mathbf{U}^{(v)}} \sum_{v=1}^{V} \|\mathbf{X}^{(v)} - \mathbf{V}^{(v)}\mathbf{U}^{(v)T}\|_F^2 + \sum_{v=1}^{V} \lambda_v \|\mathbf{U}^{(v)} - \mathbf{U}^*\|_F^2, \text{s.t.} \mathbf{V}^{(v)} \geq 0, \mathbf{U}^{(v)} \geq 0, \mathbf{U}^* \geq 0, \tag{15}$$

where $\mathbf{U}^*$ is a consensus matrix that characterizes the intrinsic clustering structures of datasets among all views and λ_v is the parameter used to tune both the relative importance of different views and the contribution between the first reconstruction error term and the second disagreement term. Note that Eq. (15) does not require that all the views share a common $\mathbf{U}^*$; thus, this model is more robust to low-quality views, i.e., the effect of low-quality views is reduced by setting the corresponding λ_v to be small enough.

Instead of enforcing a rigid common consensus constraint on all the views as in Eq. (15), another form of basic multi-view NMF for clustering is the pair-wise CoNMF model [38], which imposes similarity constraints on each pair of views. Through the pair-wise co-regularization, it is expected that the coefficient matrices learned from two views can complement with each other during the factorization process. And therefore, high-quality clustering results can be yielded. The co-regularization objective function of the pair-wise CoNMF model is defined intuitively as follows:

$$\min_{\mathbf{V}^{(v)},\, \mathbf{U}^{(v)}} \sum_{v=1}^{V} \lambda_v \|\mathbf{X}^{(v)} - \mathbf{V}^{(v)}\mathbf{U}^{(v)T}\|_F^2 + \sum_{p,\, q=1}^{V} \lambda_{pq} \|\mathbf{U}^{(p)} - \mathbf{U}^{(q)}\|_F^2, \text{s.t.} \mathbf{V}^{(v)} \geq 0, \mathbf{U}^{(v)} \geq 0, \tag{16}$$

where λ_v is the parameter employed to combine the factorization of different views and λ_{pq} is the parameter used to denote the weight of similarity constraint on $\mathbf{U}^{(p)}$ and $\mathbf{U}^{(q)}$. As the column vector of the coefficient matrix $\mathbf{U}$ represents a cluster, when adopting the vector-based ℓ_2-norm, each element of $\mathbf{U}^T\mathbf{U}$ gives the cosine similarity between two clusters. Obviously, in the multi-view environment, the cluster similarity between different views should also be consistent, which results in the cluster-wise CoNMF model. Cluster-wise CoNMF replaces the pair-wise regularization term in Eq. (16) by the following cluster-wise regularization term:

$$\sum_{p,\, q=1}^{V} \lambda_{pq} \|\mathbf{U}^{(p)T}\mathbf{U}^{(p)} - \mathbf{U}^{(q)T}\mathbf{U}^{(q)}\|_F^2. \tag{17}$$

Similar to the optimization of the standard single-view NMF model, all the three basic multi-view NMF clustering models can be optimized via the multiplicative updating rules.

3.3.3. Variants of multi-view matrix factorization

As the locality preserving learning and the manifold learning have been shown very important to promote the performance of clustering algorithms, Cai et al. [39] propose a graph (or manifold) regularized NMF model GNMF for single-view clustering with satisfying performance. Note that the aforementioned multi-view NMF models cannot preserve the local geometrical structures of the samples. To overcome this limitation, a multi-manifold regularized NMF model (MMNMF) is proposed in [40]. MMNMF incorporates consensus manifold and consensus coefficient matrix with multi-manifold regularization to preserve the local geometrical structures of the multi-view data space. The multi-manifold regularization has also been considered in [41]. Moreover, the correntropy-induced metric (CIM) is adapted to measure the reconstruction error, since CIM has achieved excellent performance in many applications. CIM is also insensitive to large errors that are mainly introduced from heavy noises. A much simpler formulation of the manifold regularized multi-view NMF model is developed in [42]. Without the explicit constraint that enforces a rigid common manifold consensus, an auxiliary matrix is involved to add constraints on the column sums of the basis matrix $\mathbf{V}^{(v)}$ such that the coefficient matrix $\mathbf{U}^{(v)}$ is comparable. A weighted extension of multi-view NMF is presented in [43] to address the image annotation problem. In this model, two weight matrices are introduced. One weight matrix is used to bias the factorization toward improved reconstruction for rare tags. The other weight matrix gives more weight to images containing rare tags and is applied to all views. A weighted extension of the pair-wise CoNMF model has also been developed in [44] to handle those attributes that are unobserved in each data sample so as to resolve the sparseness problem in all views' matrices. For the realistic cases that many views suffer from missing of some data samples resulting in many partial examples, Li et al. [45] firstly devise a partial multi-view clustering method to handle this problem. A multi-incomplete-view clustering method MIC [46] is also designed to deal with the incompleteness of the views. MIC is built upon the weighted NMF model with a $\ell_{2,1}$-norm regularization. Zhang et al. [47] further propose a constrained multi-view clustering algorithm for unmapped data in the framework of NMF. The proposed algorithm uses inter-view constraints to establish the connections between different views.

Due to its great interpretability and high efficacy, NMF has been widely employed for graph clustering [48]. In such setting, the data matrix $\mathbf{X}$ is replaced by the adjacency matrix $\mathbf{A}$. In many applications, graph data may be collected from heterogeneous domains or sources. Integrating multiple graphs has been shown to be a promising approach to improve the graph clustering accuracy. Clearly, multi-view NMF is suitable for multiple graph processing. In [49], a flexible and robust NMF-based framework, named co-regularized graph clustering (CGC), is developed to address the multi-domain graph clustering problem. CGC supports many-to-many cross-domain node relationships, and it also incorporates weights on cross-domain relationships. Besides, CGC allows partial cross-domain mapping so that graphs in different domains may have different sizes. Considering the fact that in many real-world applications, different graphs have different node distributions, the assumption that the multiple graphs share a common clustering structure does not hold. Given this, Ni et al. [50] develop a novel two-phase clustering method NoNClus, based on the NMF framework. At first, a main graph

is constructed via modeling the similarity between different domains. Then, the main graph is utilized to regularize the clustering structures in different domain-specific graphs. In the NonClus model, multiple underlying clustering structures can co-exist among domain-specific graphs, while for similar domains, the corresponding clustering structures should be as consistent as possible.

3.4. Multi-view clustering via tensor decomposition

In this part, we analyze multi-view clustering from a multilinear algebra perspective and present several novel multi-view clustering algorithms (note that the notations used in this part are self-contained). Tensor is known as a multidimensional matrix or multiway array [51]. In multi-view research field, data can be naturally modeled as a third-order tensor with objects, features, and view dimensions. An intuitive way is to compact different views along the view dimension of the tensor (see **Figure 1**). Another widely adopted way is to transform each feature matrix to a similarity matrix before compacting them.

3.4.1. Preliminaries of tensor decomposition

In the field of data mining and machine learning, tensor decomposition is an emerging and effective tool for processing multi-view data. In this section, some basic knowledge on tensors and tensor decomposition methods is provided. We refer the readers to [51, 52] for a comprehensive understanding of these topics.

3.4.1.1. Notations

Let $\mathcal{X}$ be an m-order tensor of size $I_1 \times I_2 \times \cdots \times I_m$. The mode-$p$ matricization of $\mathcal{X}$ is denoted as an $I_p \times \left(I_1 \cdots I_{p-1} I_{p+1} \cdots I_m\right)$ matrix $\mathbf{X}_{(p)}$, which is obtained by arranging the mode-p fibers to be the columns of the matrix $\mathbf{X}_{(p)}$. The p-mode multiplication $\mathcal{Y} = \mathcal{X} \times_p \mathbf{U}$ can be manipulated as matrix multiplication $\mathbf{Y}_{(p)} = \mathbf{U}\mathbf{X}_{(p)}$, where $\mathbf{U} \in \mathbb{R}^{J_p \times I_p}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \cdots I_{p-1} J_p I_{p+1} \cdots I_m}$. The Frobenius norm of a tensor $\mathcal{X}$ is the sum of the squares of all its elements $x_{i_1 i_2 \ldots i_m}$. The tensor $\mathcal{X}$ is a rank-one tensor if it can be written as the outer product of m vectors, i.e., $\mathcal{X} = \mathbf{x}^{(1)} \circ \mathbf{x}^{(2)} \circ \ldots \circ \mathbf{x}^{(m)}$, where $\circ$ represents the vector outer product.

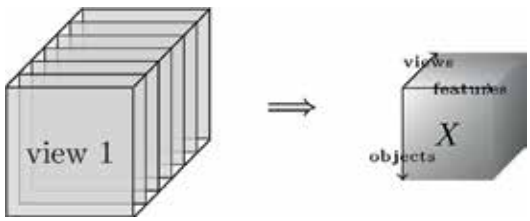

Figure 1. Visualization of the process of transforming the feature matrices to a third-order tensor.

3.4.1.2. CP decomposition

The idea of expressing tensor as the sum of a number of rank-one tensors comes from the study of Hitchcock [53]. Then, Cattell [54] proposed the idea of parallel proportional analysis. The popular CP decomposition comes from the ideas of Carroll and Chang [55] (canonical decomposition) and Harshman [56] (parallel factors). Taking a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ as an example, the CP decomposition tries to approximate tensor $\mathcal{X}$ with R components of rank-one tensor, i.e.,

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r, \tag{18}$$

where $\mathbf{u}_r \in \mathbb{R}^I$, $\mathbf{v}_r \in \mathbb{R}^J$, and $\mathbf{w}_r \in \mathbb{R}^K$. For simplicity, we denote $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_R]$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_R]$, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_R]$, and $[\![\mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$ as the CP decomposition of $\mathcal{X}$.

3.4.1.3. Tucker decomposition

The idea of Tucker decomposition is introduced by Tucker [57]. The Tucker decomposition is a form of higher-order singular value decomposition (HOSVD) [58]. It decomposes a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ into a core tensor $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ multiplied by several orthogonal matrices along each mode, i.e.,

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \mathbf{u}_p \circ \mathbf{v}_q \circ \mathbf{w}_r. \tag{19}$$

The cutting-edge technique for calculating the factor matrices is proposed in [59].

3.4.2. Tensor decomposition-based multi-view clustering

In multi-view clustering, the goal is to find out some meaningful group of objects from the data. The above CP decomposition naturally divides the multi-view data into several components, which can be seen as the clusters. Thus, it can be directly applied to solve multi-view clustering problems. For a given dataset $\mathcal{X} = \left\{ \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(V)} \right\}$ with V views, where $\mathbf{X}^{(v)}$ of each view takes value from $\mathbb{R}^{N \times M}$, $\mathcal{X}$ can be formulated as a third-order tensor $\mathcal{X} \in \mathbb{R}^{N \times M \times V}$. In this part, a variant of CP decomposition is introduced first, which is quite straightforward. Then we shed light on the relations between several classic multi-view spectral clustering methods and the Tucker decomposition.

3.4.2.1. Total variation based CP (TVCP)

In some clustering problems, a consecutive range of time points is non-negligible. For example, in the dataset with authors, publications, and a sequence of time points, we are interested in figuring out which group of authors work in the same topics during a period of time. Chen et al. [60] propose a total variation based tensor decomposition method (TVCP) for the

constraint on a period of consecutive time points. The total variation regularizes the time factor to obtain a piece-wise constant function w.r.t. time points. Owing to the piece-wise constant function, the decomposition can be relatively consistent in a cluster and separated between clusters. The TVCP model is formulated as follows:

$$\min_{[\![\mathbf{U},\mathbf{V},\mathbf{W}]\!]} \frac{1}{2}\|\mathcal{X} - \sum_{r=1}^{R} \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r\|_F^2 + \tau \sum_{r=1}^{R} \|\mathbf{F}\mathbf{w}_r\|_1, \tag{20}$$

where $\mathbf{F}$ is the first-order difference $(V-1)\times V$ matrix such that $f_{ii}=1$ and $f_{i(i+1)}=-1$ for $i=1,2,\cdots,V-1$, and the other elements are zeros, τ is a positive regularization parameter, and $\|\cdot\|_1$ denotes the ℓ_1-norm. The first term corresponds to the CP decomposition of $\mathcal{X}$, and the second term constrains the time mode ($\mathbf{w}$) to be a piece-wise constant function.

3.4.2.2. Relations between Tucker decomposition and spectral clustering

Liu et al. [61] propose a framework of multi-view clustering via tensor decomposition, mainly the Tucker decomposition. According to the framework, the common type of multi-view spectral clustering is equivalent to a Tucker decomposition problem as follows:

$$\min_{\mathbf{U}} \quad \sum_{v=1}^{V} tr\left(\mathbf{U}^T\mathbf{L}^{(v)}\mathbf{U}\right), \text{s.t.}\mathbf{U}^T\mathbf{U}=\mathbf{I}, \quad \Leftrightarrow \quad \max_{\mathbf{U}} \|\mathcal{X}\times_1\mathbf{U}^T\times_2\mathbf{U}^T\times_3\mathbf{I}^T\|_F^2. \tag{21}$$

Another form of multi-view spectral clustering can also be written as a Tucker problem:

$$\begin{aligned} &\min_{\mathbf{U},\,\mu} \; tr\left(\mathbf{U}^T\left(\sum_{v=1}^{V}\mu_v\mathbf{L}^{(v)}\right)\mathbf{U}\right), \\ &\text{s.t.}\mathbf{U}^T\mathbf{U}=\mathbf{I}, \mu_v\geq 0, \sum_{v=1}^{V}\mu_v=1, \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} &\max_{\mathbf{U},\,\mu} \|\mathcal{X}\times_1\mathbf{U}^T\times_2\mathbf{U}^T\times_3\mu^T\|_F^2, \\ &\text{s.t.}\mathbf{U}^T\mathbf{U}=\mathbf{I}, \mu_v\geq 0, \sum_{v=1}^{V}\mu_v=1. \end{aligned} \tag{22}$$

With this framework, variety of spectral clustering problems can be solved by a tensor decomposition algorithm. We can see the strong connection between them as well as the strong capability of tensor methodology.

Canonical correlation analysis is designed to inspect the linear relationship between two sets of variables [62]. In multi-view learning, a typical approach is to maximize the sum of pair-wise correlations between different views [63]. Without loss of high-order correlations, Luo et al. [64] propose a tensor canonical correlation analysis (TCCA), which is equivalent to CP decomposition of the correlation tensor. Khan et al. [65] propose a Bayesian extension of CP decomposition for multiple coupled tensors sharing common latent factors.

3.5. Multi-view clustering via deep learning

With the third wave of artificial intelligence, deep learning is gaining increasing popularity in recent years. Deep learning has demonstrated excellent performance in many real-world

applications, such as face recognition, image annotation, natural language processing, object detection, customer relationship management, and mobile advertising. Typically, deep learning models are composed of multiple nonlinear transformations and thus can learn a better feature representation than traditional shallow models [66]. However, deep learning requires labeled training data to learn the models, which limits its application in data clustering for the reason that training data with cluster labels are not available in many cases. Despite the hardness, there are some works devoted to adjusting shallow clustering models for deep learning. Here, we introduce two popular deep clustering models and their extensions to the multi-view environment.

3.5.1. Deep auto-encoder

An auto-encoder [67] is an artificial neural network adopted for unsupervised learning, the goal of which is to learn a representation for each data sample. An auto-encoder always consists of two parts: the encoder and the decoder. The encoder plays the role of a nonlinear mapping function that can map each data sample to a representation space. The decoder demands accurate data reconstruction from the representation generated by the encoder. Auto-encoder has been shown to be similar to spectral clustering in theory; however, it is more efficient and flexible in practice. The auto-encoder can be easily deepened via adding more encoder layers and corresponding decoder layers. **Figure 2 (a)** gives an example of the framework of the deep auto-encoder.

Although auto-encoder can learn a compact representation for each data sample, it contributes little to clustering since it does not require that the representation vectors of similar data samples should also be similar. To make the learned feature representation better capture the cluster structures, many variants of deep auto-encoder models have been proposed. In [68], a novel regularization term that is similar to the objective function of k-means is introduced to guide the learning of the mapping function. In this way, the learned feature representation is more stable and suitable for clustering. In [69], a deep embedded clustering method is proposed to simultaneously learn feature representations and cluster assignments using deep auto-encoders. These deep clustering models are designed for single-view data. For deep multi-view clustering, the learned feature representations should not only capture the cluster structure of each single view but also implement a consensus between different views. To this end, a common encoder is utilized to extract the shared feature representation for all views,

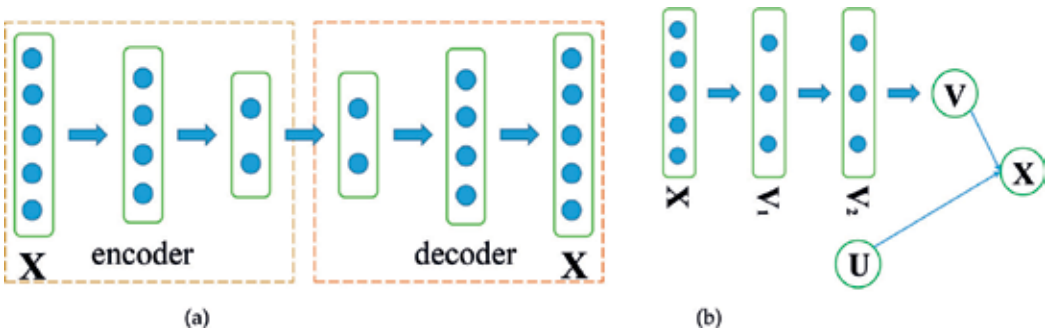

Figure 2. Frameworks of deep auto-encoder and deep matrix factorization (depth is 2).

and different decoders are used to reconstruct view-specific input data samples [70]. In [71], an extension of CCA based on deep neural networks is proposed to learn a shared representation of two views. In fact, the feature representations of the two views are not exactly the same, but their correlations are maximized. Following this line, the deep canonically correlated auto-encoder (DCCAE) is developed in [72]. DCCAE simultaneously optimizes the canonical correlation between the learned feature representations and the reconstruction errors of the auto-encoders. Benton et al. [73] further extend the deep CCA model for multiple views.

3.5.2. Deep matrix factorization

Another line of developing deep clustering models is deepening the MF models. As shown earlier, MF, especially NMF, has demonstrated outstanding performance in many applications. Thus, it is worth building a deep structure for MF in the hope that better feature representations can be obtained to facilitate clustering. **Figure 2(b)** illustrates an example of the framework of the deep MF models. Compared to the deep auto-encoders, both deep MF and deep auto-encoders are trying to minimize the reconstruction errors. However, unlike deep auto-encoders, the mapping function of deep MF is linear.

The first nonnegative deep network based on NMF is proposed in [74] for speech separation. This architecture can be discriminatively trained for optimal separation performance. Then Li et al. [75] propose a novel weakly supervised deep MF model to uncover the latent image representations and tag representations embedded in the latent subspace by collaboratively exploring the weakly supervised tagging information, the visual structure, and the semantic structure. In [76], a deep semi-NMF model is further developed for learning latent attribute representations. Semi-NMF is a popular variant of NMF by relaxing the factorized basis matrix to be real-valued. This practice makes semi-NMF have much wider applications than NMF since the datasets in real world may contain complex information, for instance, the attributes may be mix-signed. Considering the fact that these deep MF models are trying to factorize the basis matrix hierarchically alone, Qiu et al. [77] further propose a deep orthogonal NMF model which can decompose the coefficient matrix hierarchically. This model is able to learn higher-level representations for clusters. These deep MF models have achieved great success in data clustering for single-view data. However, they are seldom utilized for multi-view clustering. A recent work [78] attempts to extend the deep semi-NMF model for multi-view clustering, which can dissemble unimportant factors layer by layer and generate an effective consensus representation in the last layer. Another work [79] proposes to address the incomplete multi-view clustering problem via deep semantic mapping. The proposed model first projects all incomplete multi-view data to a unified representation in a common subspace, which is further executed by standard shallow NMF for clustering.

4. Open datasets

No one can make bricks without straw. In this section we will first list two kinds of open datasets that can be used in multi-view clustering, i.e., feature-based and graph-based datasets. Then we will discuss the performance of multi-view clustering on them briefly.

4.1. Feature-based datasets

Audio genre [80] consists of 1886 audio tracks classified into 9 music genres, which are Blues, Electronic, Jazz, Pop, Rap/HipHop, Rock, Folk/Country, Alternative, and Funk/Soul. Forty-nine low-level audio features have been extracted and they are grouped into 15 vector spaces.

NUS-WIDE [81] is a web image dataset composed of 269,648 images, 5018 related tags, and 81 ground-truth concepts. Six types of low-level features have been extracted: 64-D color histogram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture, 225-D block-wise color moments extracted over 5×5 fixed grid partitions, and 500-D bag of words based on SIFT descriptions.

UCF101 [82] consists of 101 human action classes. These actions can be divided into five types: human-object interaction, body-motion only, human-human interaction, playing musical instruments, and sports. There are over 13,000 clips and 27 hours of video data in it.

Handwritten numerals [83] is composed of 2000 handwritten digits which are divided into 10 classes. Four types of feature sets have been extracted: Zernike moments, Karhunen-Loeve features, Fourier descriptors, and image vectors. For Zernike set, it has 47 rotation invariant Zernike moments and 6 morphological features. For Fourier set, it has 76 two-dimensional shape descriptors. Both Zernike and Fourier feature sets are rotation invariant. For Karhunen-Loeve set, it has 64 Karhunen-Loeve transform which corresponds to the projection of images onto the eigenvectors of a covariance matrix.

4.2. Graph-based datasets

DBLP coauthorship [84] is a coauthorship network composed of 10,305 authors. There are 617 layers in it, each layer representing different publication categories.

Facebook [85] is a three-layer social network composed of 1640 users with multiple types of ties. The first layer shows whether two users are friends. The second layer shows whether users are in a same group. The third layer shows whether users are in the same photos uploaded by users.

CiteSeer [86] consists of 3312 scientific publications classified into 6 classes, which are Agents, AI, DB, IR, ML, and HCI. It can be represented as an annotated network, where nodes represent scientific publications and links represent the citation relationships. For each node, there is a 3703-dimensional one-hot encoding vector representing the absence/presence of key words.

Enron e-mail [87] consists of 184 users and 44 layers. Although it is a temporal network, it can be considered as a multi-layer network. Each layer represents communication in different months.

4.3. Performance on different datasets

For feature-based datasets, when confronted with the situation where we need to reconstruct the views, the performance of classical methods, like deep learning, is not promising. But

http://dx.doi.org/10.5772/intechopen.75598

multi-view clustering can give satisfactory results under this condition. In some cases, classical methods can also give good performance for feature-based datasets where all features are descriptions of the same object from different perspectives. For graph-based datasets, multi-view clustering naturally fits into them since different graphs can be processed by different views.

For both feature-based and graph-based datasets, when the scale of datasets becomes significantly large, most multi-view clustering methods have the potential to outperform other clustering methods on speed. For example, multi-view matrix factorization is quite suitable to parallel process.

5. Open issues

Although multi-view clustering has demonstrated its superiority over single-view clustering in many applications, there are still many open issues deserving much more attention from both academia and industry. Several vital open issues are summarized in this part.

5.1. View construction

Although there are many typical methods to construct views, they all have their own drawbacks. It is well known that if we cannot extract valuable information from the original data and put it into different views appropriately, the performance will be highly limited no matter how delicate the algorithm is. So it is important to find efficient ways of constructing and evaluating multiple views.

5.2. Incomplete view

When constructing different views, we may find that for some views, the information is not complete. In other words, even though we know how to construct views appropriately, we do not have enough information to do it, which is very common in practical problems. In real world, it is very difficult to ensure the completeness of data. This unbalanced relationship between complete views and incomplete views could cause huge problems. Moreover, these incomplete views may influence views with complete information. To solve it, one possible way is to construct these lost information from other views.

5.3. Single-view to multi-view

In multi-view learning, sometimes researchers will convert single-view data into multiple views and apply relevant algorithms on them. In practice, it may give good performance, but there are few theoretical researches on the proof of its reliability. Since the original data is single view, it is important to make it clear: is it necessary to complicate a simple task? We should not only focus on the final performance, the trade-off between cost and benefit is also important.

5.4. Deep leaning in multi-view

Deep learning has shown remarkable performance in many fields. One common way to deal with data composed of different types of sources is to combine them together and then feed them into a deep learning model. It often works well. Although multi-view learning seems to be a more reasonable way to deal with data composed of different types of sources, there is no evidence showing that multi-view learning has an obvious advantage over deep learning. Another issue is that when using deep learning in multi-view learning, we need to train different neural networks for different views separately. This method has two drawbacks. One is that the number of neural networks depends on the number of views. When there are many views, the calculation is huge. The other is that it fails to unify different views during training.

6. Conclusion

Multi-view clustering has demonstrated variety of real-world applications, such as community detection in social networks, image annotation in computer vision, cross-domain user modeling in recommendation systems, and protein interaction analysis in bioinformatics. This chapter provides a comprehensive review of the typical multi-view clustering methods and their corresponding recent developments by focusing on five most typical and popular clustering methods, which include *k*-means, spectral clustering, matrix factorization, tensor decomposition, and deep learning. The basic forms of these five clustering methods are introduced in detail, followed by a substantial overview of their recent developments. Several open datasets and open issues are discussed in the end, which deserves more attention to facilitate the future research of multi-view clustering.

In the field of multi-view clustering, there are many algorithms whose source codes are exposed by their authors. For example, the co-training[1] and co-regularization[2] methods of classical multi-view spectral clustering are open in GitHub with MATLAB. The variants MSE[3] and AMGL[4] are also implemented by MATLAB.

Author details

Fanghua Ye[1], Zitai Chen[1], Hui Qian[1], Rui Li[2], Chuan Chen[1]* and Zibin Zheng[1]

*Address all correspondence to: chenchuan@mail.sysu.edu.cn

1 School of Data and Computer Science, Sun Yat-sen University, China

2 School of Physics, Sun Yat-sen University, China

---

[1] https://github.com/areslp/matlab/tree/master/code_cospectral

[2] https://github.com/areslp/matlab/tree/master/code_coregspectral

[3] https://github.com/rciszek/mse

[4] http://www.escience.cn/people/fpnie/index.html;jsessionid = 253C211B5AEDB8C09865FFEAEAACFB73-n1

References

[1] Bickel S, Scheffer T. Multi-view clustering. ICDM. 2004;**4**:19-26

[2] Chang X, Tao D, Xu C. A survey on multi-view learning. arXiv preprint arXiv. 2013;**1304**: 5634

[3] Sun S. A survey of multi-view machine learning. Neural Computing and Applications. Feb 2013;**23**(7–8):2031-2038

[4] Zhao J, Xie X, Xin X, Sun S. Multi-view learning overview: Recent progress and new challenges. Information Fusion. 2017;**38**:43-54

[5] Liu X. Learning from Multi-View Data: Clustering Algorithm and Text Mining Application. Leuven, Belgium: KU Leuven; 2011

[6] Ali Mamdouh E, Yang S, Xiaodong H. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In: WWW, International World Wide Web Conferences Steering Committee; 2015. pp. 278-288

[7] Blaschko MB, Lampert CH. Correlational spectral clustering. In: CVPR. IEEE; 2008. pp. 1-8

[8] Kailing K, Kriegel H-P, Pryakhin A, Schubert M. Clustering multi-represented objects with noise. In: PAKDD. Springer Berlin Heidelberg: Springer; 2004. pp. 394-403

[9] Chaudhuri K, Kakade SM, Livescu K, Sridharan K. Multi-view clustering via canonical correlation analysis. In: ICML. ACM; 2009. pp. 129-136

[10] Yin Q, Shu W, He R, Wang L. Multi-view clustering via pairwise sparse subspace representation. Neurocomputing. 2015;**156**:12-21

[11] Jain AK. Data clustering: 50 years beyond k-means. PRL. Jun 2010;**31**(8):651-666

[12] Maldonado S, Carrizosa E, Weber R. Kernel penalized k-means: A feature selection method based on kernel k-means. Information Sciences. 2015;**322**:150-160

[13] Liang D, Zhou P, Shi L, Wang H, Fan M, Wang W, Shen Y-D. Robust multiple kernel k-means using l21-norm. In: IJCAI; 2015

[14] Liu X, Li M, Wang L, Dou Y, Yin J, Zhu E. Multiple kernel k-means with incomplete kernels. In: AAAI: 2017. pp. 2259-2265

[15] Wang S, Gittens A, Mahoney MW. Scalable kernel k-means clustering with nystrom approximation: Relative-error bounds. arXiv preprint arXiv. 2017;**1706**:02803

[16] Zhang R, Rudnicky AI. A large scale clustering scheme for kernel k-means. In: ICPR. Vol. 4. IEEE; 2002. pp. 289-292

[17] Dhillon IS, Guan Y, Kulis B. Kernel k-means. In: SIGKDD. ACM, ACM Press; 2004. pp. 551-556

[18] Tzortzis G, Likas A. Kernel-based weighted multi-view clustering. In: ICDM. IEEE; 2012. pp. 675-684

[19] Cai X, Nie F, Huang H. Multi-view k-means clustering on big data. In: IJCAI; 2013. pp. 2598-2604

[20] Zhao H, Yun F. Dual-regularized multi-view outlier detection. In: IJCAI; 2015. pp. 4077-4083

[21] Chen X, Xiaofei X, Huang JZ, Ye Y. Tw-k-means: Automated two-level variable weighting clustering algorithm for multiview data. TKDE. 2013;**25**(4):932-944

[22] Yu-Meng X, Wang C-D, Lai J-H. Weighted multi-view clustering with feature selection. Pattern Recognition. 2016;**53**:25-35

[23] Bo J, Qiu F, Wang L. Multi-view clustering via simultaneous weighting on views and features. Applied Soft Computing. 2016;**47**:304-315

[24] Xu C, Tao D, Xu C. Multi-view self-paced learning for clustering. In: IJCAI; 2015. pp. 3974-3980

[25] von Luxburg U. A tutorial on spectral clustering. Statistics and Computing. 2007;**17**(4): 395-416

[26] Hagen L, Kahng AB. New spectral methods for ratio cut partitioning and clustering. TCAD. 1992;**11**(9):1074-1085

[27] Shi J, Malik J. Normalized cuts and image segmentation. TCAD. 2000;**22**(8):888-905

[28] Kumar A, Daumé H. A co-training approach for multi-view spectral clustering. In: ICML; 2011. pp. 393-400

[29] Kumar A, Rai P, Daume H. Co-regularized multi-view spectral clustering. In: NIPS; 2011. pp. 1413-1421

[30] Xia T, Tao D, Mei T, Zhang Y. Multiview spectral embedding. SMCB. 2010;**40**(6):1438-1446

[31] Nie F, Li J, Li X et al. Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification. In: IJCAI; 2016. pp. 1881-1887

[32] Xia R, Pan Y, Lei D, Yin J. Robust multi-view spectral clustering via low-rank and sparse decomposition. In: AAAI; 2014, pp. 2149-2155

[33] Chen C, Ng MK, Zhang S. Block spectral clustering methods for multiple graphs. Numerical Linear Algebra with Applications. 2017;**24**:e2075. DOI: 10.1002/nla.2075

[34] Wang Y-X, Zhang Y-J. Nonnegative matrix factorization: A comprehensive review. TKDE. 2013;**25**(6):1336-1353

[35] Kivinen J, Warmuth MK. Additive versus exponentiated gradient updates for linear prediction. In Proceedings of the twenty-seventh annual ACM symposium on Theory of computing

(STOC '95). ACM, New York, NY, USA.1995. pp. 209-218. http://dx.doi.org/10.1145/225058.225121

[36] Lee DD, Sebastian Seung H. Learning the parts of objects by non-negative matrix factorization. Nature. 1999;**401**(6755):788-791

[37] Liu J, Wang C, Gao J, Han J. Multi-view clustering via joint nonnegative matrix factorization. In: ICDM. SIAM; 2013. pp. 252-260

[38] He X, Kan M-Y, Xie P, Chen X. Comment-based multi-view clustering of web 2.0 items. In: WWW. ACM, ACM Press; 2014. pp. 771-782

[39] Cai D, He X, Han J, Huang TS. Graph regularized nonnegative matrix factorization for data representation. PAMI. 2011;**33**(8):1548-1560

[40] Zong L, Zhang X, Zhao L, Hong Y, Zhao Q. Multi-view clustering via multi-manifold regularized non-negative matrix factorization. Neural Networks. 2017;**88**:74-89

[41] Weihua O, Shujian Y, Li G, Jian L, Zhang K, Xie G. Multi-view non-negative matrix factorization by patch alignment framework with view consistency. Neurocomputing. 2016;**204**:116-124

[42] Hidru D, Goldenberg A. Equinmf: Graph regularized multiview nonnegative matrix factorization. arXiv preprint arXiv. 2014;**1409**:4018

[43] Kalayeh MM, Idrees H, Shah M. Nmf-knn: Image annotation using weighted multi-view non-negative matrix factorization. In: CVPR; 2014. pp. 184-191

[44] Gong X, Wang F, Huang L. Weighted nmf-based multiple sparse views clustering for web items. In: PAKDD. Springer; 2017. pp. 416-428

[45] Li S-Y, Jiang Y, Zhou Z-H. Partial multi-view clustering. In: AAAI; 2014

[46] Shao W, He L, Philip SY. Multiple incomplete views clustering via weighted nonnegative matrix factorization with $l_{2,1}$ regularization. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer; 2015. pp. 318-334

[47] Zhang X, Zong L, Liu X, Yu H. Constrained nmf-based multi-view clustering on unmapped data. In: AAAI; 2015. pp. 3174-3180

[48] Wang F, Li T, Wang X, Zhu S, Ding C. Community discovery using nonnegative matrix factorization. DMKD. 2011;**22**(3):493-521

[49] Cheng W, Zhang X, Guo Z, Yubao W, Sullivan PF, Wang W. Flexible and robust co-regularized multi-domain graph clustering. In: SIGKDD. ACM; 2013. pp. 320-328

[50] Ni J, Tong H, Fan W, Zhang X. Flexible and robust multi-network clustering. In: SIGKDD. ACM; 2015. pp. 835-844

[51] Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Review. Aug 2009; **51**(3):455-500

[52] Sidiropoulos ND, De Lathauwer L, Xiao F, Huang K, Papalexakis EE, Faloutsos C. Tensor decomposition for signal processing and machine learning. SP. 2017;**65**(13):3551-3582

[53] Hitchcock FL. The expression of a tensor or a polyadic as a sum of products. Journal of Mathematical Physics. Apr 1927;**6**(1–4):164-189

[54] Cattell RB. "Parallel proportional profiles" and other principles for determining the choice of factors by rotation. Psychometrika. Dec 1944;**9**(4):267-283

[55] Douglas Carroll J, Chang J-J. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-young" decomposition. Psychometrika. Sep 1970;**35**(3):283-319

[56] Harshman RA. Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis. UCLA Working Papers in Phonetics. 1970;**16**: 1-84

[57] Tucker LR. Some mathematical notes on three-mode factor analysis. Psychometrika. 1966; **31**(3):279-311

[58] De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applications. July 2000;**21**:1253-1278

[59] De Lathauwer L, De Moor B, Vandewalle J. On the best rank-1 and rank-(r1 ,r2 ,…,rn) approximation of higher-order tensors. SIAM Journal on Matrix Analysis and Applications. 2000;**21**(4):1324-1342

[60] Chen C, Li X, Ng MK, Yuan X. Total variation based tensor decomposition for multi-dimensional data with time dimension. Numerical Linear Algebra with Applications. May 2015;**22**(6):999-1019

[61] Liu X, Ji S, Glänzel W, De Moor B. Multiview partitioning via tensor methods. TKDE. 2013; **25**(5):1056-1069

[62] Hardoon DR, Szedmak S, Shawe-Taylor J. Canonical correlation analysis: An overview with application to learning methods. Neural Computation. Dec 2004;**16**(12):2639-2664

[63] Vía J, Santamaría I, Pérez J. A learning algorithm for adaptive canonical correlation analysis of several data sets. Neural Networks. 2007;**20**(1):139-152

[64] Luo Y, Tao D, Ramamohanarao K, Xu C, Wen Y. Tensor canonical correlation analysis for multi-view dimension reduction. TKDE. Nov 2015;**27**(11):3111-3124

[65] Khan SA, Kaski S. Bayesian multi-view tensor factorization. In: Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg: Springer; 2014. pp. 656-671

[66] Zhao L, Chen Z, Yang Z, Yueming H, Obaidat MS. Local similarity imputation based on fast clustering for incomplete data in cyber-physical systems. IEEE Systems Journal. 2016; **PP**(99):1-11

[67] Bengio Y. Learning deep architectures for Al. Foundations and trends® in Machine Learning. 2009;**2**(1):1-127

[68] Song C, Huang Y, Liu F, Wang Z, Liang W. Deep auto-encoder based clustering. Intelligent Data Analysis. 2014;**18**(6S):S65-S76

[69] Xie J, Girshick R, Farhadi A. Unsupervised deep embedding for clustering analysis. In: ICML; 2016. pp. 478-487

[70] Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY. Multimodal deep learning. In: ICML; 2011. pp. 689-696

[71] Andrew G, Arora R, Bilmes J, Livescu K. Deep canonical correlation analysis. In: ICML; 2013. pp. 1247-1255

[72] Wang W, Arora R, Livescu K, Bilmes J. On deep multi-view representation learning: Objectives and optimization. arXiv preprint arXiv. 2016;**1602**:01024

[73] Benton A, Khayrallah H, Gujral B, Reisinger D, Zhang S, Arora R. Deep generalized canonical correlation analysis. arXiv preprint arXiv. 2017;**1702**:02519

[74] Le Roux J, Hershey JR, Weninger F. Deep nmf for speech separation. In: ICASSP. IEEE; 2015, pp. 66-70

[75] Li Z, Tang J. Weakly supervised deep matrix factorization for social image understanding. IP. Jan 2017;**26**(1):276-288

[76] Trigeorgis G, Bousmalis K, Zafeiriou S, Schuller BW. A deep matrix factorization method for learning attribute representations. PAMI. 2017;**39**(3):417-429

[77] Qiu Y, Zhou G, Xie K. Deep approximately orthogonal nonnegative matrix factorization for clustering. arXiv preprint arXiv. 2017;**1711**:07437

[78] Zhao H, Ding Z, Fu Y. Multi-view clustering via deep matrix factorization. In: AAAI; 2017. pp. 2921-2927

[79] Zhao L, Chen Z, Yi Y, Jane Wang Z, Leung VCM. Incomplete multi-view clustering via deep semantic mapping. Neurocomputing. 2018;**275**:1053-1062

[80] Homburg H, Mierswa I, Moller B, Morik K, Wurst M. A benchmark dataset for audio classification and clustering. In: Ismir 2005, Proceedings of the International Conference on Music Information Retrieval, 11–15 September 2005; London. Uk; 2005. pp. 528-531

[81] Chua TS, Tang J, Hong R, Li H, Luo Z, Zheng Y. Nus-wide: a real-world web image database from national university of singapore. In: ACM International Conference on Image and Video Retrieval; 2009. p. 48

[82] Soomro K, Zamir AR, Shah M. Ucf101: A dataset of 101 human actions classes from videos in the wild. CRCV-TR-12-01, November, 2012

[83] Van Breukelen M, Duin RPW, Tax DMJ, Den Hartog JE. Handwritten digit recognition by combined classifiers. Kybernetika. 1998;**34**(4):381-386

[84] Ng KP, Li X, Ye Y. Multirank: co-ranking for objects and relations in multi-relational data. In: SIGKDD; 2011. pp. 1217-1225

[85] Lewis K, Kaufman J, Gonzalez M, Wimmer A, Christakis N. Tastes, ties, and time: A new social network dataset using facebook.Com. Social Networks. 2008;**30**(4):330-342

[86] Sen P, Namata G, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T. Collective classification in network data articles. AI Magazine. 2008;**29**(3):93-106

[87] Bader BW, Harshman RA, Kolda TG. Temporal analysis of semantic graphs using asalsan. In: ICDM; 2007. pp. 33-42

Chapter 12

Collective Solutions on Sets of Stable Clusterings

Vladimir Vasilevich Ryazanov

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.76189

Abstract

Two clustering problems are considered. We consider a lot of different clusters of the same data for a given number of clusters. Data clustering is understood as their stable partition into a given number of sets. Clustering is considered stable if the corresponding partitioning remains unchanged with its minimum change. How to create a new cluster based on ensemble clusterings? The second problem is the following. A definition of the committee synthesis as ensemble clustering is introduced. The sets of best and worst matrices of estimates are considered. Optimum clustering is built on the basis of the clusterings obtained as being closest to the set of the best estimation matrices or as the most distant from the set of worst-case matrices of estimates. As a result, the problem of finding the best committee clustering is formulated as a discrete optimization problem on permutations.

Keywords: clustering, algorithm, ensemble, collective, stability, optimality, construction

1. Introduction

There are many different approaches to solving the problems of clustering multidimensional data: based on the optimization of internal criteria (indices) [1, 2], hierarchical clustering [3], centroid-based clustering [4], density-based clustering [5], distribution-based clustering [6], and many others. There are well-known books and papers on clustering [7–10].

This section is devoted to one approach to the creation of stable clusterings and the processing of their sets. A natural criterion is considered, which is applicable to any clustering method. In work [11], various criteria (indices) are proposed, optimizing which clustering is built with a definite look "what is clustering?" In this chapter, we use a criterion based on stability. If we really got clustering, that is, a solution for the whole sample, the partitioning should not

change with a small change in the data. Criteria are introduced for the quality of the partition obtained. If the criterion value is less than one, then the partition is unstable. Let us obtain for the same data N clusterings. How to create a new ensemble clustering based on the N partitions? Previously, a committee method for building ensemble clusterings was proposed [12–15]. Let there be N results of cluster analysis of the same data for l clusters. The committee method of building ensemble clustering makes it possible to build such l clusters, each of which is the intersection of "many" initial clusters. In other words, we find such l clusters whose objects are "equivalent" to each other according to several principles. As initial N clusterings, one can take stable ones. Finally, we consider a video-logical approach to building the initial N coarse clusterings.

2. Criteria for stability of clustering

Let the sample of objects $\mathrm{X} = \{\mathbf{x}_i, i = 1, 2, \ldots, m\}$, $\mathbf{x}_i \in R^n$ be given and $\mathrm{K} = \{K_1, K_2, \ldots, K_l\}$ is the clustering of the sample into l clusters obtained by some method, $K_i \subseteq \mathrm{X}$, $i = 1, 2, \ldots, l$, $\cup_1^l K_i = \mathrm{X}$, $K_i \cap K_j = \varnothing$, $i \neq j$. Speaking of clustering, we mean applying a method to a sample without focusing on the method itself. Is partition K of a sample by this method clustering or here some kind of stopping criterion is satisfied? For example, an extremum of some functional is obtained or the maximum number of operations in the iterative process is fulfilled. We will use the following thesis as the main one. If the resulting partition K is indeed clustering, then it must be the same clustering for any minimal change in the sample X. Let $\mathbf{x}_i$ be arbitrary, $\mathbf{x}_i \in K_\alpha$ ensemble then the sample $\mathrm{X}\backslash\{\mathbf{x}_i\}$ partition $\mathrm{K}^*(\mathbf{x}_i) = \{K_1^*, K_2^*, \ldots, K_l^*\}$, $K_j^* = K_j$, $j = 1, 2, \ldots, l$, $j \neq \alpha$, $K_\alpha^* = K_\alpha \backslash \{\mathbf{x}_i\}$, $i = 1, 2, \ldots, m$ must be clustering. The fact of "coincidence" of clusterings $\mathrm{K} = \{K_1, K_2, \ldots, K_l\}$ and $\mathrm{K}^*(\mathbf{x}_i) = \{K_1^*, K_2^*, \ldots, K_l^*\}$ will be called identity, the clusterings themselves are identical and denoted it as $\mathrm{K}^*(\mathbf{x}_i) \approx \mathrm{K}$. In this case, it is natural to call a partition K as stable clustering if the partitions $\mathrm{K}^*(\mathbf{x}_i)$ and K coincide for all $\mathbf{x}_i$, $i = 1, 2, \ldots, m$. In the case of non-identity of some individual $\mathrm{K}^*(\mathbf{x}_i)$ with K, we will call K as quasi-clustering.

Definition 1. The quality of quasi-clustering (of unstable clustering) is the quantity $\Phi(\mathrm{K}) = |\{x_i, i = 1, 2, \ldots, m : \mathrm{K}^*(\mathbf{x}_i) \approx \mathrm{K}\}|/m$.

If $\Phi(\mathrm{K}) = 1$, then in this case, we will talk about stable clustering K or simply clustering. Suppose that for some i, $i = 1, 2, \ldots, m$ the condition $\mathrm{K}^*(\mathbf{x}_i) \approx \mathrm{K}$ is not satisfied, and $\mathrm{K}^\circ(\mathbf{x}_i) = \{K_1^\circ, K_2^\circ, \ldots, K_l^\circ\}$ is the clustering of the sample $X\backslash\{\mathbf{x}_i\}$ obtained from the partition $\mathrm{K}^*(\mathbf{x}_i)$ using $\mathrm{K}^*(\mathbf{x}_i)$ as the initial approximation. Then $\mathrm{K}^\circ(\mathbf{x}_i)$ can significantly differ from $\mathrm{K}^*(\mathbf{x}_i)$. We will use as a function of the proximity between clustering $\mathrm{K}^\circ(\mathbf{x}_i)$ and partitioning K the value $d(\mathrm{K}^\circ(\mathbf{x}_i), \mathrm{K}) = \max_{\boldsymbol{\alpha}} \sum_{i=1}^l \left|K_i^\circ \cap K_{\alpha_i}\right|/(m-1)$. Note that to calculate proximity it is required to find the maximum matching in a bipartite graph, for which there is a polynomial algorithm [16]. If $\mathrm{K}^\circ(\mathbf{x}_i)$ does not exist, we will assume that $d(\mathrm{K}^\circ(\mathbf{x}_i), \mathrm{K}) = 0$.

Definition 2. The quality $F_{\min}(\mathrm{K})$ of the quasi-clustering K will be called the quantity $F_{\min}(\mathrm{K}) = \min_i d(\mathrm{K}^\circ(\mathbf{x}_i), \mathrm{K})$.

Definition 3. The quality $F_{avr}(\mathrm{K})$ of the quasi-clustering K will be called the quantity $F_{avr}(\mathrm{K}) = \sum_{i=1}^{m} d(\mathrm{K}^{\circ}(\mathbf{x}_i), \mathrm{K})/m$.

For some clustering algorithms, there are simple economical rules for computing $\Phi(\mathrm{K})$. Let us bring them (see also in [3, 17, 18]).

2.1. Method of minimizing the dispersion criterion

It is known that in order to minimize the dispersion criterion, it suffices to satisfy inequalities

$$\frac{n_j}{(n_j-1)}\|\mathbf{x}^{\times}-\mathbf{m}_j\|^2 - \frac{n_k}{(n_k+1)}\|\mathbf{x}^{\times}-\mathbf{m}_k\|^2 \le 0 \tag{1}$$

for any clusters K_j and K_k, arbitrary $\mathbf{x}^{\times} \in K_j$, where $n_j = \left|K_j\right|$, $\mathbf{m}_j = \frac{1}{n_j}\sum_{\mathbf{x}_t \in K_j} \mathbf{x}_t$.

We establish the conditions for the identity $\mathrm{K}^*(\mathbf{x}_i) \approx \mathrm{K}$ of the partitions $\mathrm{K}^*(\mathbf{x}_i)$ and K. In the case $\mathbf{x}^{\times} \in K_j$ [considering (Eq. (1))] to satisfy the condition $\mathrm{K}^*(\mathbf{x}_i) \approx \mathrm{K}$ inequalities.

$\frac{(n_j-1)}{(n_j-2)}\|\mathbf{x}^{\times}-\mathbf{m}_j\|^2 + \frac{2}{(n_j-2)}(\mathbf{x}^{\times}-\mathbf{m}_j, \mathbf{x}_i-\mathbf{m}_j) + \frac{1}{(n_j-1)(n_j-2)}\|\mathbf{x}_i-\mathbf{m}_j\|^2 - \frac{n_k}{n_k+1}\|\mathbf{x}^{\times}-\mathbf{m}_k\|^2 \le 0$ must be satisfied. In the case $\mathbf{x}^{\times} \in K_k$ inequalities $\frac{n_k}{(n_k-1)}\|\mathbf{x}^{\times}-\mathbf{m}_k\|^2 - \frac{(n_j-1)}{n_j}\|\mathbf{x}^{\times}-\mathbf{m}_j\|^2 - \frac{2}{n_j}(\mathbf{x}^{\times}-\mathbf{m}_j, \mathbf{x}_i-\mathbf{m}_j) - \frac{1}{n_j(n_j-1)}\|\mathbf{x}_i-\mathbf{m}_j\|^2 \le 0$ must be satisfied.

2.2. *k*-means method

Let the clustering K be obtained by *k*-means method, that is, $\|\mathbf{x}^{\times}-\mathbf{m}_j\| \le \|\mathbf{x}^{\times}-\mathbf{m}_k\|, \forall j \ne k$, $\forall \mathbf{x}^{\times} \in K_j$. In the case of equality, the object is considered to belong to a cluster with a lower number. Then, $\mathrm{K}^*(\mathbf{x}_i) \approx \mathrm{K}$ is satisfied if $\|\mathbf{x}^{\times}-\mathbf{m}_j\|^2 + \frac{2}{(n_j-1)}(\mathbf{x}^{\times}-\mathbf{m}_j, \mathbf{x}_i-\mathbf{m}_j) + \frac{1}{(n_j-1)^2}\|\mathbf{x}_i-\mathbf{m}_j\|^2 \le \|\mathbf{x}^{\times}-\mathbf{m}_k\|^2$ under $\mathbf{x}^{\times} \in K_j, \mathbf{x}^{\times} \ne \mathbf{x}_i$ and $\|\mathbf{x}^{\times}-\mathbf{m}_k\|^2 \le \|\mathbf{x}^{\times}-\mathbf{m}_j\|^2 + \frac{2}{(n_j-1)}(\mathbf{x}^{\times}-\mathbf{m}_j, \mathbf{x}_i-\mathbf{m}_j) + \frac{1}{(n_j-1)^2}\|\mathbf{x}_i-\mathbf{m}_j\|^2$ under $\mathbf{x}^{\times} \in K_k$.

2.3. Method of hierarchical agglomeration grouping

We confine ourselves to the case of an agglomeration hierarchical grouping. To find the value of the criterion $\Phi(\mathrm{K})$, you can calculate the partitioning K, partitions $\mathrm{K}^{\circ}(\mathbf{x}_i)$, $i = 1, 2, \ldots, m$, and compare K with each $\mathrm{K}^{\circ}(\mathbf{x}_i)$, $i = 1, 2, \ldots, m$. Here it is possible to save in the calculation of $\Phi(\mathrm{K})$ without carrying through the clustering for some of "i". Indeed, let there $\mathrm{K}^t(\mathbf{x}_i) = \{K_1^t, K_2^t, \ldots, K_{m-t}^t\}$ be clustering of the sample $X\backslash\{\mathbf{x}_i\}$ into $m-t$ clusters, $t \le m-l$. K is a partition obtained by the clustering algorithm X. The main property of the hierarchical grouping is that for any $k = 1, 2, \ldots, m-t$ there is $j = 1, 2, \ldots, m-t-1$ for which $K_k^t \subseteq K_j^{t+1}$. In this case, if at some step t, $t \le m-l$ for some k the condition $K_k^t \subseteq K_j$ does not hold for all $j = 1, 2, \ldots, l$, then the condition $\mathrm{K}^*(\mathbf{x}_i) \approx \mathrm{K}$ will not be fulfilled.

2.4. Examples

We give some examples illustrating the stability criteria introduced.

1. Below are the results obtained for model samples. The method of clustering based on the minimization of the dispersion criterion [3] has been used. As the initial data, we used samples of a mixture of two two-dimensional normal distributions with independent features, different **a**, and **σ**. Examples are shown in **Figures 1–3** (images of the samples in question) and in **Tables 1** and **2**. **Figure 1** represents a sample of 200 objects for which all the criteria $\Phi(\mathrm{K})$, $F_{\min}(\mathrm{K})$, $F_{avr}(\mathrm{K})$ are equal to 1, and the resulting clustering into two clusters is stable clustering. Here we used distributions with parameters $\mathbf{a}_1 = (0,0)$, $\mathbf{a}_2 = (9,9)$, and $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 = (3,3)$.

Further, with the same parameters $\mathbf{a}_1$, $\mathbf{a}_2$, experiments were carried out for $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 = (5,5)$.

Then, we used distributions with parameters $\mathbf{a}_1 = (0,0)$, $\mathbf{a}_2 = (9,9)$, $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 = (10,10)$, $m = 200$. In this case, we have the case of strongly intersecting distributions. Formally, the clustering method gives a quasi-clustering, approximately corresponding to the partitioning of the original sample (**Figure 3**) into two sets by a diagonal from the upper left corner of the picture to the lower right. The values of the criteria in **Table 2** were obtained.

2. Data clustering of [19] and criteria values $\Phi(\mathrm{K})$, $F_{\min}(\mathrm{K})$, $F_{avr}(\mathrm{K})$. The following data from classification problem of electromagnetic signals were considered: $n = 34$, $m_1 = 225$, $m_2 = 126$, $l = 2$. We give the values of the stability criteria obtained. **Figure 4** shows the visualization [3] of the sample. The accuracy of the supervised classification methods was about 87% of the correct answers. However, the clustering of data turned out to be only quasi-clustering (**Table 3**).

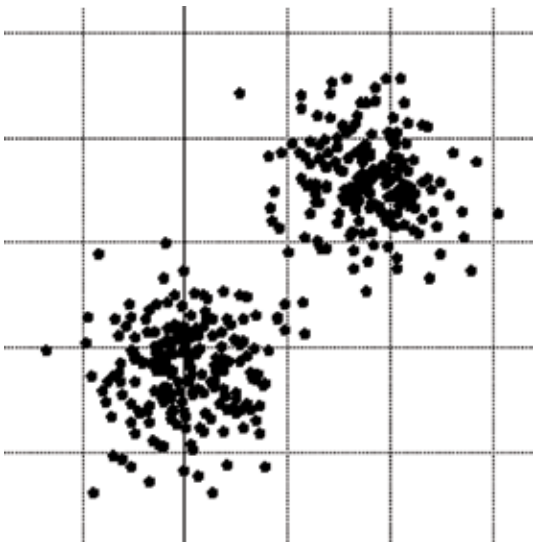

Figure 1. Clustering in a task with parameters $\mathbf{a}_1 = (0,0)$, $\mathbf{a}_2 = (9,9)$, $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 = (3,3)$, $m = 200$.

http://dx.doi.org/10.5772/intechopen.76189

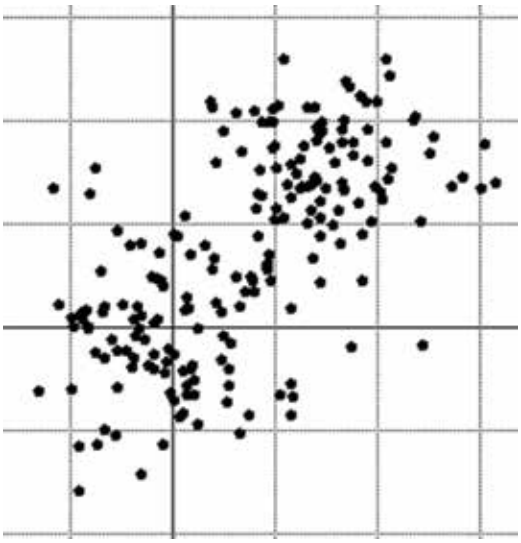

Figure 2. Clustering in a task with parameters $\mathbf{a}_1 = (0,0)$, $\mathbf{a}_2 = (9,9)$, $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 = (5,5)$, $m = 200$.

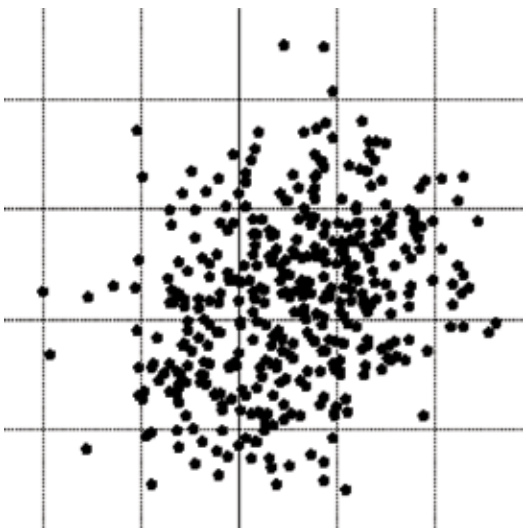

Figure 3. Data with parameters $\mathbf{a}_1 = (0,0)$, $\mathbf{a}_2 = (9,9)$, $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 = (10,10)$, $m = 200$.

$\Phi(\mathrm{K})$	0.995
$F_{\min}(\mathrm{K})$	0.995
$F_{avr}(\mathrm{K})$	0.999

Table 1. Values of quasi-clustering criteria.

Φ(K)	0.770
$F_{\min}$(K)	0.995
F_{avr}(K)	0.998

Table 2. Values of quasi-clustering criteria. Case of very intersecting distributions

Figure 4. Data visualization.

Φ(K)	0.966
$F_{\min}$(K)	0.997
F_{avr}(K)	0.999

Table 3. The values of the criteria in the problem "ionosphere" Φ(K), $F_{\min}$(K), F_{avr}(K).

3. Committee synthesis of ensemble clustering

The problem is as follows. There are N clusterings for the same number of clusters. How to choose from them the only one or build a new clustering from the available ones? In the supervised classification problem (with the help of a collective solution of a set of algorithms) there is a criterion according to which one can choose an algorithm from existing ones or build a new algorithm. This is a supervised classification error. This direction in the theory of classification appeared in the early 1970s of the last century [20, 21], then was created an algebraic approach [22], various correctors were appeared. The key in the algebraic approach is the creation in the form of special algebraic polynomials of a correct (error-free) algorithm based on a set of supervised classification algorithms. Some algebraic operations on matrices of "degrees of belonging" of recognized objects are used. Various types of correctors were also created [22–25], when the problem of constructing (and applying) the best algorithm is also solved in two stages. First, the supervised classification algorithms are determined, and then the corrector. This can be, for example, the problem of approximating a given partial Boolean function by some monotonic function. In recent decades, there are conferences on multiple classifier systems, these issues are reflected in the books [21, 10]. How to choose or create the

best clustering using a finite set of given solutions? Here, all problems are connected primarily with the absence of a single generally accepted criterion. Each clustering algorithm finds such "source" clusters of objects that are "equivalent" to each other. In this chapter, it is proposed to build such a clustering of the initial data, the cluster solutions of which have a large intersection with the initial clusters.

Let the sample of objects $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, $\mathbf{x}_i \in R^n$ for supervised classification and l classes are given. In the theory of supervised classification, the following definition of the supervised classification algorithm exists [21]. Let $\alpha_{ij} \in \{0, 1\}$ be equal to 1 when the object $\mathbf{x}_i$, $i = 1, 2, \ldots, m$ is classified by the algorithm A^r as $\mathbf{x}_i \in K_j$ and 0 otherwise: $A^r(X) = \|\alpha_{ij}\|_{m\times l}$. Here the intersection of classes is allowed. Unlike the supervised classification problem, when clustering a sample, we have freedom in the designation of clusters.

Definition 4. The matrices $I = \|\alpha_{ij}\|_{m\times l}$, $\alpha_{ij} \in \{0, 1\}$ and $I' = \|\alpha'_{ij}\|_{m\times l}$, $\alpha'_{ij} \in \{0, 1\}$ are said to be equivalent if they are equals to within a permutation of the columns.

It is clear that this definition defines a class of equivalent matrices for some matrix.

Definition 5. A clustering algorithm is an algorithm that maps a sample X to a set of equivalent information matrices $A^c(X) = K(\|\alpha_{ij}\|_{m\times l})$.

The number of clusters and the length of the control sample are considered to be given. This definition emphasizes the fact that in an arbitrary partition of a sample into l clusters, we have complete freedom in the numbering of clusters. In what follows we shall always consider matrices of dimension $m \times l$.

Let there be given N algorithms $A_1^c, A_2^c, \ldots, A_N^c$ for clustering and their solutions $A_v^c(X) = K\left(\|\alpha_{ij}^v\|_{m\times l}\right)$ for sample X. We denote $I_v = \|\alpha_{ij}^v\|_{m\times l}$ an arbitrary element of the clustering $K\left(\|\alpha_{ij}^v\|_{m\times l}\right)$.

Therefore, we have $I = K(I_1) \times K(I_2) \times \ldots \times K(I_N)$ or set $I = \left\{\left(I'_1, I'_2, \ldots, I'_N\right), I'_v \in K(I_v)\right\}$, $I'_v = \|\alpha'^v_{ij}\|_{m\times l}$.

There are two problems.

1. Construction of the mapping I on, K_c, $I \to K_c = \left\{K\|c_{ij}\|_{m\times l}\right\}$, $c_{ij} \in \{0, 1\}$ (that is, the construction of some kind of clustering).
2. Finding the optimal element in K_c (i.e. finding the best clustering in K_c).

Definition 6. An operator $B\left(I'_1, I'_2, \ldots, I'_N\right) = B = \|b_{ij}\|_{m\times l}$ is called an adder if $b_{ij} = \sum_{v=1}^{N} \alpha'^v_{ij}$.

It is clear that $0 \le b_{ij} \le N$, $b_{ij} \in \{0, 1, 2, \ldots, N\}$.

Definition 7. An operator r is called a threshold decision rule, if $r(B) = C = \|c_{ij}\|_{m\times l}$, $c_{ij} = \begin{cases} 1, & b_{ij} \geq \delta_i, \\ 0, & \text{otherwise}, \end{cases}$ where $\delta_i \in R$.

Definition 8. By the committee synthesis of an information matrix C on an element $\tilde{I}' = \left(I_1', I_2', \ldots, I_N'\right)$ let us call it a computation by the formula $C = r\mathrm{B}\left(\tilde{I}'\right)$, provided that B is the adder and r is the threshold decision rule.

The general scheme of collective synthesis is shown in **Figure 5**.

We note that the total number of possible values B is bounded from above by a quantity $(l!)^N$. Let s be the operator that performs permutation of columns of matrices $m \times l$ with the help of a substitution $< j_1, j_2, \ldots, j_l >$, $S = \{s\}$ is the set of all operators s. We believe that $rs = sr, \forall s \in S$. We continue $s \in S$ to the n-dimensional case $\sigma\left(\tilde{I}'\right) = \left(s\left(I_1'\right), s\left(I_2'\right), \ldots, s\left(I_n'\right)\right)$. We denote $\Sigma = \{\sigma\}, \sigma$ is the extension of s. From the definition of the adder it follows that $\sigma\mathrm{B} = \mathrm{B}\sigma, \forall\sigma \in \Sigma$. Further, $\forall\tilde{I}' \in \mathrm{I}, \forall\sigma \in \Sigma$ we have $r\mathrm{B}\left(\sigma\left(\tilde{I}'\right)\right) = r\sigma\left(\mathrm{B}\left(\tilde{I}'\right)\right) = s\left(r\mathrm{B}\left(\tilde{I}'\right)\right)$ and finally $\left\{\sigma\left(\tilde{I}'\right), \sigma \in \Sigma\right\} \overset{r\mathrm{B}}{\to} \left\{s\left(r\mathrm{B}\left(\tilde{I}'\right)\right),\ s \in S\right\} = \mathrm{K}\left(r\mathrm{B}\left(\tilde{I}'\right)\right) = \mathrm{K}(\|c_{ij}\|_{m\times l})$. Therefore, the product rB defines the desired mapping and specifies some ensemble clustering. It is necessary to determine the optimal element from K_c, find it and $\tilde{I}'$.

$$\mathrm{I} \overset{r\mathrm{B}}{\to} \mathrm{K}_c,\ A^c_{\tilde{I}'}(\mathrm{X}) = \mathrm{K}\left(r\mathrm{B}\left(\tilde{I}'\right)\right).$$

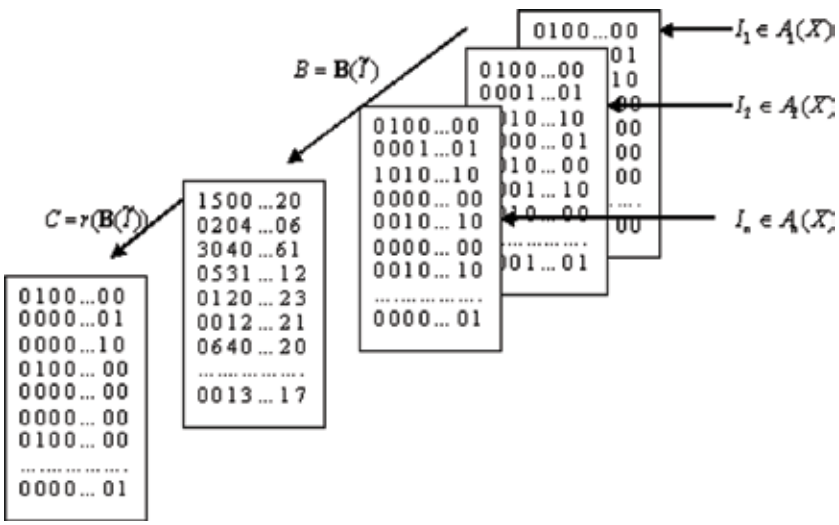

Figure 5. Scheme of committee synthesis.

We introduce definitions of potentially best and worst-case solutions. As the "ideal" of the collective solution, we will consider the case when all algorithms give us essentially the same partitions or coverings.

Definition 9. A numerical matrix $\|b_{ij}\|_{m\times l}$ is called contrasting if $b_{ij} \in \{0, N\}$. A numeric matrix $\|b_{ij}\|_{m\times l}$ is called blurred if $b_{ij} = \delta_i \in R$.

As the distance between two numerical matrices, we consider the function

$$\rho\left(B^1, B^2\right) = \sum_{i=1}^{m}\sum_{j=1}^{l}\left|b_{ij}^1 - b_{ij}^2\right|.$$

Denote by M the set of all contrast matrices, and by $\tilde{M}$ the set of all blurred matrices. We introduce definitions for estimating the quality of matrices.

Definition 10.

$$\Phi(B) = \rho(B, \mathrm{M}) \underset{B}{\rightarrow} \min. \tag{2}$$

Definition 11.

$$\tilde{\Phi}(B) = \rho\left(B, \tilde{\mathrm{M}}\right) \underset{B}{\rightarrow} \max. \tag{3}$$

The set $\tilde{\mathrm{M}}' = \left\{\tilde{B}\right\}$ (where $\tilde{B} = \|\tilde{b}_{ij}\|_{m\times l}$, $\tilde{b}_{ij} = \frac{N}{2}$) is called the mean blurred matrix.

Definition 12.

$$\tilde{\Phi}'(B) = \rho\left(B, \tilde{B}\right) \underset{B}{\rightarrow} \max \tag{4}$$

We note that the optimums according to the criteria (Eq. (2)) and (Eq. (3)) do not have to coincide. The sets M and $\tilde{\mathrm{M}}$ intersect.

Figure 6 illustrates the sets of contrasting and blurred matrices. Arrows indicate some elements of sets.

Theorem 1. The sets of optimal solutions by criteria Eqs. (2) and (4) coincide.

Let us show that $\Phi(B)+\tilde{\Phi}'(B) = \frac{Nml}{2}$ for any B. We write $\tilde{\Phi}'(B) = \sum_{i=1}^{m}\sum_{j=1}^{l}\tilde{\alpha}_{ij}$, $\tilde{\alpha}_{ij} = \left|b_{ij} - \frac{N}{2}\right|$, $\Phi(B)$ $= \sum_{i=1}^{m}\sum_{j=1}^{l}\alpha_{ij}^*$, $\alpha_{ij}^* = \min\left(b_{ij}, N - b_{ij}\right)$. If $b_{ij} \geq \frac{N}{2}$ then $\tilde{\alpha}_{ij} = b_{ij} - \frac{N}{2}$, $\alpha_{ij}^* = N - b_{ij}$, and $\tilde{\alpha}_{ij}+\alpha_{ij}^* = \frac{N}{2}$. If $b_{ij} < \frac{N}{2}$ then $\tilde{\alpha}_{ij} = \frac{N}{2} - b_{ij}$, $\alpha_{ij}^* = b_{ij}$, and $\tilde{\alpha}_{ij}+\alpha_{ij}^* = \frac{N}{2}$.

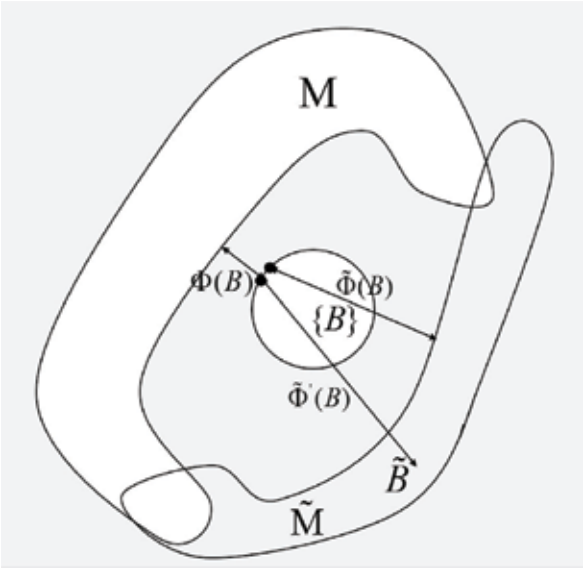

Figure 6. The sets of contrasting M, blurred $\tilde{M}$ matrices, and the set of matrices $\{B\}$.

Summing over all the set of values of pairs of indices i, j, we get that $\Phi(B)+\tilde{\Phi}'(B) = \frac{Nml}{2}$.

We consider the problem of finding optimal ensemble clusterings for the criterion (2). It is clear that $\Phi(B) = \sum_{i=1}^{m}\sum_{j=1}^{l}\min\left(b_{ij}, N - b_{ij}\right)$.

We introduce the notations $M = \{1, 2, \ldots, m\}$, $X_j = \left\{i | b_{ij} \geq \frac{N}{2}, i = 1, 2, \ldots, m\right\}$, $Y_j = M \backslash X_j$, $j = 1, 2, \ldots, l$. Let $\pi^{\nu} =< \mu_1^{\nu}, \mu_2^{\nu}, \ldots, \mu_l^{\nu} >$, $\nu = 1, 2, \ldots, N$ be some permutation of the set $\pi^0 =< 1, 2, \ldots, l >$. A set of permutations $\pi =< \pi^1, \pi^2, \ldots, \pi^N >$ uniquely determines the matrix of estimates.

$$B' = \left\|b'_{ij}\right\|_{m\times l}, b'_{ij} = b_{ij}(\pi) = \sum_{\nu=1}^{N}\alpha'^{\nu}_{ij}.$$

We will further assume that the "initial" matrix $\left\|\alpha_{ij}^{\nu}\right\|_{m\times l}$ of the algorithm A_{ν}^{c} corresponds to the permutation π^0. $\left\|\alpha'^{\nu}_{ij}\right\|_{m\times l}$ is the matrix of the algorithm A_{ν}^{c} corresponding to some permutation π^{ν}. Then $\alpha'^{\nu}_{ij} = \alpha^{\nu}_{i\mu_j^{\nu}}$.

Consider $\tilde{\Delta}_{\nu} = \sum_{j=1}^{l}\left(\sum_{i\in X_j}\overline{\alpha}_{ij}^{\nu} + \sum_{i\in Y_j}\alpha_{ij}^{\nu}\right)$, $\tilde{\Delta}'_{\nu} = \sum_{j=1}^{l}\left(\sum_{i\in X_j}\overline{\alpha}'^{\nu}_{ij} + \sum_{i\in Y_j}\alpha'^{\nu}_{ij}\right)$.

Then $\Delta_{\nu} = \tilde{\Delta}'_{\nu} - \tilde{\Delta}_{\nu} = \sum_{j=1}^{l}\left(\sum_{i\in X_j}\left(\alpha^{\nu}_{ij} - \alpha'^{\nu}_{ij}\right) + \sum_{i\in Y_j}\left(\alpha'^{\nu}_{ij} - \alpha^{\nu}_{ij}\right)\right)$. We convert this expression.

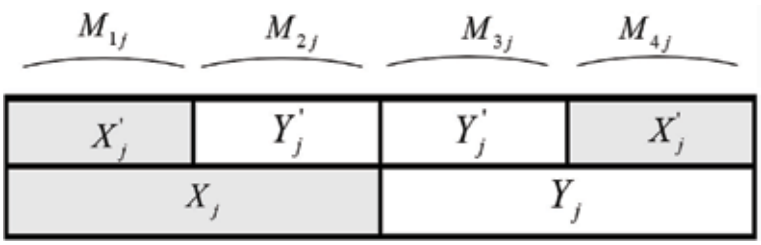

Figure 7. Sets X_j, Y_j, $j = 1, 2, \ldots, l$ are changed.

The identity $\sum_{j=1}^{l}\left(\sum_{i\in X_j}\alpha_{ij}^{\nu}+\sum_{i\in Y_j}\alpha_{ij}^{\nu}\right)=\sum_{j=1}^{l}\left(\sum_{i\in X_j}\alpha_{i\mu_j^{\nu}}^{\nu}+\sum_{i\in Y_j}\alpha_{i\mu_j^{\nu}}^{\nu}\right)$ is valid. Get $\Delta_{\nu}=\sum_{j=1}^{l}\left(\sum_{i\in X_j}\left(\alpha_{ij}^{\nu}-\alpha_{i\mu_j^{\nu}}^{\nu}\right)+\sum_{i\in Y_j}\left(\alpha_{i\mu_j^{\nu}}^{\nu}-\alpha_{ij}^{\nu}\right)\right)=2\sum_{j=1}^{l}\sum_{i\in X_j}\left(\alpha_{ij}^{\nu}-\alpha_{i\mu_j^{\nu}}^{\nu}\right)=2\sum_{j=1}^{l}\sum_{i\in X_j}\alpha_{ij}^{\nu}-2\sum_{j=1}^{l}\sum_{i\in X_j}\alpha_{i\mu_j^{\nu}}^{\nu}$.

Thus, minimizing a function is equivalent to maximizing the second sum of the expression.

After applying the permutations $\pi=<\pi^1,\pi^2,\ldots,\pi^N>$, the sets X_j, Y_j, $j = 1, 2, \ldots, l$ change. We introduce the notations $M_{1j}=X_j\backslash\left(Y_j'\backslash Y_j\right)$, $M_{2j}=Y_j'\backslash Y_j$, $M_{3j}=Y_j\backslash\left(X_j'\backslash X_j\right)$, $M_{4j}=X_j'\backslash X_j$.

Figure 7 schematically shows the changes in sets X_j, Y_j, $j = 1, 2, \ldots, l$.

Theorem 2

$$\Delta\Phi=\Phi\left(B'\right)-\Phi(B)\le\sum_{\nu=1}^{N}\Delta_{\nu}+\sum_{\nu=1}^{N}\left(|M_{2j}|\begin{cases}-2, & N-even,\\ -1, & N-odd\end{cases}+|M_{4j}|\begin{cases}0, & N-even,\\ -1, & N-odd\end{cases}\right)$$

The proof is given in [12, 13]. Theorem 2 is the basis for creating an effective minimization algorithm of Φ.

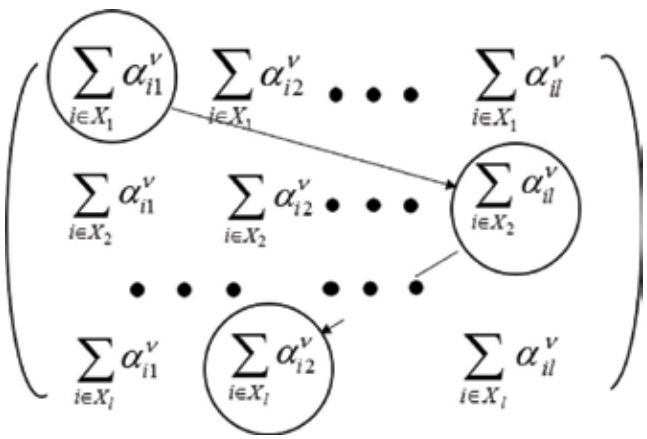

Figure 8. All possible variants of $\sum_{j=1}^{l}\sum_{i\in X_j}\alpha_{i\mu_j^{\nu}}^{\nu}$ for all admissible j and i.

Since the second sum is always not positive, we have an upper bound. We consider the problem of minimizing a function Δ_ν. We write out all possible variants of the function $\sum_{j=1}^{l}\sum_{i\in X_j}\alpha^{\nu}_{i\mu^{\nu}_j}$ in the form of a table in **Figure 8**. Then the minimum of this function is reduced to finding the maximum matching of the bipartite graph, for finding which we can use the polynomial Hungarian algorithm [16].

It is clear that $\min\limits_{\pi^\nu}\Delta_\nu \le 0$. Now we can propose the following heuristic algorithm for steepest descent.

Algorithm.

1. We calculate X_j, $j = 1, 2, \ldots, l$.

2. We find $\Delta^*_\nu = \min\limits_{\pi^\nu}\Delta_\nu$ for each ν.

If $\sum_{\nu=1}^{N}\Delta^*_\nu < 0$, then apply the found permutations $\pi^\nu =< \mu^\nu_1, \mu^\nu_2, \ldots, \mu^\nu_l >$, $\nu = 1, 2, \ldots, N$ and go to step 1).

If $\sum_{\nu=1}^{N}\Delta^*_\nu = 0$ then the END of algorithm.

NOTE. We note that our algorithm does not even find a local minimum of the criterion $\Phi(B)$. Nevertheless, this algorithm is very fast, its complexity at each iteration is estimated as $O(l^5 mN)$.

4. The algorithm of collective k-means

Results of clustering by N algorithms of sampling of m objects to l clusters solutions are obtained, which we can write in the form of a binary matrix $\|\alpha^v_{ij}\|$, $\nu = 1, 2, \ldots, N$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, l$. We assume that the cluster numbers in each algorithm are fixed. Then any horizontal layer number i of this three-dimensional matrix will denote the results of object $\mathbf{x}_i$ clustering. As an ensemble clustering of the sample X, we can take the result of clustering the "new" descriptions—the layers of the original matrix $\|\alpha^v_{ij}\|$, $\nu = 1, 2, \ldots, n$. As a method of clustering, we take the method of minimizing the dispersion criterion. Let there be a lot of N clusterings $\|\alpha^v_{i_1 j}\|, \|\alpha^v_{i_2 j}\|, \ldots, \|\alpha^v_{i_N j}\|$ with heuristic clustering algorithms, then we calculate their sample mean $\|\alpha^{*v}_j\|$ as the solution of the problem $\sum_{\mu=1}^{t}\left(\alpha^{*v}_j - \alpha^v_{i_\mu j}\right)^2 \to \min\limits_{\alpha^{*v}_j}$. Where do we obtain $\alpha^{*v}_j = \frac{1}{N}\sum_{\mu=1}^{N}\alpha^v_{i_\mu j}$. Note that this method makes it possible to calculate such ensemble clusterings $\mathrm{K} = \left\{K^*_1, K^*_2, \ldots, K^*_l\right\}$ that the sets of heuristic clustering of the objects of some cluster of the collective solution will be close to each other in the Euclidean metric. The committee synthesis of collective decisions provides more interpretable solutions. Indeed, if $\mathrm{K}^\nu = \left\{K^\nu_1, K^\nu_2, \ldots, K^\nu_l\right\}$, $\nu = 1, 2, \ldots, N$ are separate solutions of heuristic clustering algorithms, then the cluster of collective solution will be the "intersection" of many some original clusters $K^1_{i_1}, K^2_{i_2}, \ldots, K^N_{i_l}$.

5. Man-machine (video-logical) clustering method

In the problems of ensemble clustering synthesis considered earlier, we did not consider the number of initial clustering algorithms, their quality and their proximity. Ensemble clustering was built and reflected only the opinion of the collective decisions that we used. "Internal" indices [9] reflect the person's ideas about clustering. You can think up examples of data when known internal criteria lead to degenerate solutions.

At the same time, a person has the ability to cluster visual sets on a plane without using any proximity functions, criteria and indices. The following idea was realized. A person can personally cluster projections of sets of points from R^n into R^2. Having made such clusterings under different projections, we can construct generally speaking various N clusterings, which we submit to the input of the construction of the collective solution. The person himself "does not see" the objects in R^n, but can exactly solve the clustering tasks on the plane. Thus, here we use N precise solutions, but of various partial information about the data. Consider this video-logical method on one model example.

A sample of two normal distributions with independent characteristics was considered. The first feature of the first distribution (200 objects) had zero expectation and the standard deviation, the first attribute of the second distribution (200 objects) had these values equal to 5. All the other 49 attributes for all objects had $\mathbf{a}_i = 5$, $\boldsymbol{\sigma}_i = 5$, $i = 2, 3, \ldots, 50$. That is, the two sets had equal distributions for 49 features and one informative feature. Clustering of the entire sample by minimizing dispersion is shown in **Figure 9**. Black and gray points on sample visualization represent the objects of the first and second clusters. Here the fact of informative character of the first feature is lost.

The program of the video-logical approach worked as follows. With the help of a single heuristic approach, all C_n^2 projections are automatically ordered according to the descending criteria of the presence of two clusters. Next we as experts consider some projections and with the help of the mouse we select in each of them two clusters. **Figure 10** shows two such examples. Note that the first feature was present in all projections. It was used "manually" as the defining area for the dense location of objects. Then 10 "manual" clustering went to the program entrance for the committee synthesis of the collective solution. Note that only two objects were erroneously clustered.

Figure 9. Clustering of a sample of model objects by the method of minimizing variance.

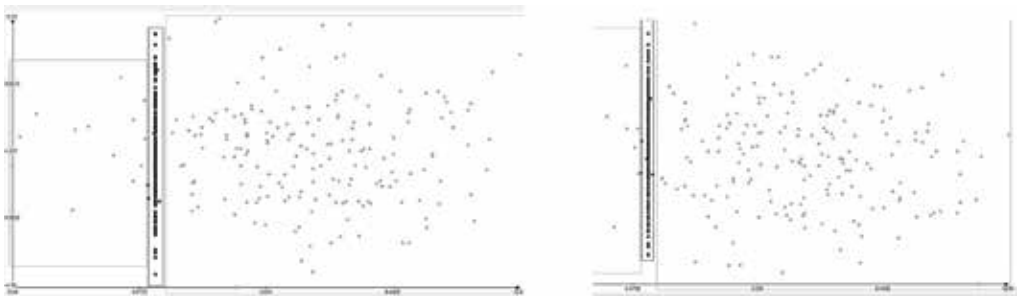

Figure 10. Allocation of clusters by mouse on the (1.4) and (1.6) features.

6. Conclusion

This chapter consists of two parts. First, clustering criteria based on sustainability are introduced. Next, we propose an approach to processing the sets of obtained partitions of the same sample. As the initial clustering, it is better to use stable clustering. It is shown how a person can be used in the construction of the committee synthesis of ensemble clustering.

Acknowledgements

The reported study was funded by RFBR according to the research project No 17-01-00634 and No 18-01-00557.

Author details

Vladimir Vasilevich Ryazanov

Address all correspondence to: rvvccas@mail.ru

Dorodnicyn Computing Centre, Federal Research Center, "Computer Science and Control" of Russian Academy of Sciences, Moscow, Russia

References

[1] Halkidi M, Batistakis Y, Vazirgiannis M. Cluster validity methods: Part 1. SIGMOD Record. 2002;**31**(2):40-45. DOI: 10.1145/601858.601862

[2] Aggarwal C, Reddy C. Data Clustering: Algorithms and Applications. CRC Press; 2014

[3] Duda R, Hart P, Stork D. Pattern Classification. 2nd ed. New York: Wiley; 2000

[4] Lloyd S. Least squares quantization in PCM (PDF). IEEE Transactions on Information Theory. 1982;**28**(2):129-137. DOI: 10.1109/TIT.1982.1056489

[5] Kriegel H, Kröger P, Sander J, Zimek A. Density-based clustering. WIREs Data Mining and Knowledge Discovery. 2011;**1**(3):231-240. DOI: 10.1002/widm.30

[6] Dempster A, Laird N, Rubin D. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B. 1977;**39**(1):1-38 JSTOR 2984875. MR 0501537

[7] Jain A, Dubes R. Algorithms for Clustering Data. Englewood Cliffs: Prentice-Hall, Inc.; 1998

[8] Kaufman L, Rousseeuw P. Finding Groups in data: An Introduction to Cluster Analysis. New York: Wiley; 2009

[9] Aggarwal C. Data Mining: The Textbook. Yorktown Heights/New York: IBM T.J. Watson Research Center; 2015. 771 p. DOI: 10.1007/978-3-319-14142-8

[10] Kuncheva L. Combining Pattern Classifiers: Methods and Algorithms. Hoboken: Wiley; 2004. DOI: 10.1002/9781118914564

[11] Desgraupes B. Clustering indices. University Paris Ouest. Lab Modal'X; 2013

[12] Ryazanov V. Commitee synthesis of algorithms for recognition and classification. Journal of Computational Mathematics and Mathematical Physics. 1981;**21**(6):1533-1543. DOI: 10.1016/0041-5553(81)90161-0

[13] Ryazanov V. On the synthesis of classification algorithms on finite sets of classification algorithms (taxonomy). Journal of Computational Mathematics and Mathematical Physics. 1982;**22**(2):429-440. DOI: 10.1016/0041-5553(82)90049-0

[14] Ryazanov V. One approach for classification (taxonomy) problem solution by sets of heuristic algorithms. In: Proceedings of the 9-th Scandinavian Conference on Image Analysis; 6–9 June 1995; Uppsala; 1995(2). pp. 997-1002

[15] Biryukov A, Shmakov A, Ryazanov V. Solving the problems of cluster analysis by collectives of algorithms. Journal of Computational Mathematics and Mathematical Physics. 2008;**48**(1):176-192. DOI: 10.1134/S0965542508010132

[16] Kuhn H. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly. 1955;**2**:83-97. DOI: 10.1002/nav.3800020109

[17] Ryazanov V. Estimations of clustering quality via evaluation of its stability. In: Bayro-Corrochano E, Hancock E, editors. CIARP 2014. LNCS. Vol. 8827; 2014. pp. 432-439. DOI: 10.1007/978-3-319-12568-8_53

[18] Ryazanov V. About estimation of quality of clustering results via its stability. Intelligent Data Analysis. 2016;**20**:S5-S15. DOI: 10.3233/IDA-160842

[19] Sigillito V, Wing S, Hutton L, Baker K. Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest. 1989;**10**:262-266

[20] Rastrigin L, Erenstein R. Collective decision-making in pattern recognition. Avtomatica i telemehanika. 1975;**9**:133-144

[21] Method of committees in pattern recognition. Sverdlovsk, IMM AN USSR; 1974

[22] Yu Z. On the algebraic approach to solving problems of recognition or classification. Problems of Cybernetics. 1978;**33**:5-68

[23] Zuev Y. The method of increasing the reliability of classification in the presence of several classifiers, based on the principle of monotony. Journal of Computational Mathematics and Mathematical Physics. 1981;**21**(1):157-167

[24] Krasnoproshin V. About the optimal corrector of the set of recognition algorithms. Journal of Computational Mathematics and Mathematical Physics. 1979;**19**(1):204-215

[25] Zhuravlev Y. Selected Scientific Works. Moscow: Publishing House Magister; 1998. p. 420

Edited by Harun Pirim

Clustering has emerged as one of the more fertile fields within data analytics, widely adopted by companies, research institutions, and educational entities as a tool to describe similar/different groups.

The book *Recent Applications in Data Clustering* aims to provide an outlook of recent contributions to the vast clustering literature that offers useful insights within the context of modern applications for professionals, academics, and students. The book spans the domains of clustering in image analysis, lexical analysis of texts, replacement of missing values in data, temporal clustering in smart cities, comparison of artificial neural network variations, graph theoretical approaches, spectral clustering, multiview clustering, and model-based clustering in an R package. Applications of image, text, face recognition, speech (synthetic and simulated), and smart city datasets are presented.

Published in London, UK

IntechOpen

ISBN 978-1-83881-560-8