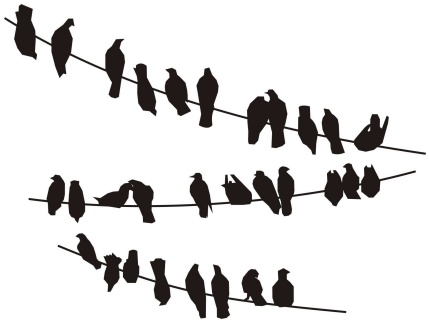
QINPEI ZHAO

Cluster Validity in Clustering Methods



Publications of the University of Eastern Finland
Dissertations in Forestry and Natural Sciences
No 77

Academic Dissertation
To be presented by permission of the Faculty of Science and Forestry for public examination in Louhela Auditorium in Science Park at the University of Eastern Finland, Joensuu, on June 25, 2012,
at 12 o'clock noon.

School of Computing

Kopijyvä Oy
Joensuu, 2012
Editor: Prof. Pertti Pasanen and Prof. Pekka Kilpeläinen

Distribution:
University of Eastern Finland Library / Sales of publications
P.O. Box 107, FI-80101 Joensuu, Finland
tel. +358-50-3058396
http://www.uef.fi/kirjasto

ISBN: 978-952-61-0840-7 (printed)
ISSNL: 1798-5668
ISSN: 1798-5668
ISBN: 978-952-61-0841-4 (pdf)
ISSNL: 1798-5676
ISSN: 1798-5676

Author's address:
University of Eastern Finland
School of Computing
P.O.Box 111
80101 JOENSUU
FINLAND
email: qinpei.zhao@uef.fi

Supervisors:
Professor Pasi Fränti, Ph.D.
University of Eastern Finland
School of Computing
P.O.Box 111
80101 JOENSUU
FINLAND
email: pasi.franti@uef.fi

Mantao Xu, Ph.D.
Shanghai Dianji University
School of Electrical Engineering
JiangChuan Road, Minhang District
200240 SHANGHAI
CHINA
email: xumt@sdju.edu.cn

Reviewers:
Professor Olli Nevalainen, Ph.D.
University of Turku
Department of Computer Science
ICT-talo Joukahaisenkatu 3-5B
20014 TURKU
FINLAND
email: olli.nevalainen@utu.fi

Yi Da (Richard) Xu, Ph.D.
University of Technology, Sydney
School of Computing and Communications
P.O.Box 123
2070 SYDNEY
AUSTRALIA
email: yida.xu@uts.edu.au

Opponent:
Professor Tapio Pahikkala, Ph.D.
University of Turku
Department of Information Technology
ICT-talo, 6th floor, Joukahaisenkatu 3-5 B
20520 TURKU
FINLAND
email: tapio.pahikkala@it.utu.fi

ABSTRACT

Cluster analysis plays an important role in many areas of science, and clustering algorithms and cluster validation are two essential elements. Before clustering, the number of clusters is an essential parameter for the clustering algorithm, while after clustering, the validity of the clustering is performed.

Internal indexes such as the *Bayesian information criterion* (BIC) and *sum-of-squares* have difficulties in finding a knee point of the indexes, so detection methods through BIC in partition-based clustering are proposed in the present study. A new sum-of-squares based index is also proposed, where the minimal value is considered the optimal number of clusters. External indexes, on the other hand, need a reference clustering or ground-truth information of data and therefore cannot be used in cluster validity. Consequently, we extend the external index into an internal index in order to determine the number of clusters by introducing a re-sampling method.

Iterative algorithms, such as the K-means and EM algorithms, suffer from an initialisation problem, so a random swap strategy is employed to overcome this issue. In the present thesis, we extend this approach to the optimisation of the EM algorithm for learning Gaussian mixture model from multivariate data. The EM variant is known as the *random swap EM* (RSEM) algorithm. It provided in our practical tests better results than split-and-merge (SMEM) and is more efficient than repeated EM (REM).

We propose a cluster-level validity criterion called a *centroid ratio*. It has low time complexity and is applicable for detecting unstable or incorrectly located centroids. Employing the centroid ratio in swap-based clustering, we further suggest a *pairwise random swap* clustering algorithm, for which no stopping criterion is required.

AMS Classification: 62H30, 68Q25, 68W40
Universal Decimal Classification: 004.93, 519.237.8
Library of Congress Subject Headings: Data mining; Machine learning; Image processing; Parameter estimation; Cluster analysis; Algorithms

Yleinen suomalainen asiasanasto: tiedonlouhinta; koneoppiminen; kuvankäsittely; ryhmittelyanalyysi; algoritmit; parametrit; estimointi; validointi

Preface

According to *six degrees of separation*, everyone is on average approximately six steps away, by way of introduction, from any other person on Earth. I would like to firstly thank Dr. Mantao Xu, who provided me a connection from Shanghai to Joensuu and introduced me to Prof. Pasi Fränti, my supervisor. He is the person, who gave me the chance to be here. As a supervisor, he provided a lot of support and gave direction on my research. I have learned more than knowledge from him, also attitude on work. I would like to thank my colleagues in SIPU lab and department, who make me work smoothly. Many thanks to my friends in Joensuu for bringing me happiness.

I would like to thank Prof. Olli Nevalainen and Dr. Richard Xu, the reviewers of the thesis for their time and useful comments and Prof. Tapio Pahikkala for acting as my opponent. I am grateful to all people who worked with me on the publications, especially Dr. Ville Hautamäki and Dr. Ismo Kärkkäinen. I would like to acknowledge Dr. Congli Yang and Zhitao Wen in mathematics for the discussion we have conducted. I would also like to thank East Finland graduate school in Computer Science and Engineering (ECSE) for seven months' financial support in 2009. The work of the thesis was also finally supported by the School of Computing, Center for International Mobility (CIMO) and Nokia Foundation.

At last, I would like to denote my deepest love and gratitude to my fiance Jinhua Chen for his endless support on my work and life. I am also deeply grateful to my mother Jingfang Lu for her understanding and support.

Joensuu March 8, 2012 *Qinpei Zhao*

LIST OF PUBLICATIONS

This thesis consists of the present review of the author's work in the field of cluster analysis and the following selection of the author's publications:

P1 Q. Zhao, V. Hautamäki and P. Fränti, "Knee Point Detection in BIC for Detecting the Number of Clusters", *Advanced Concepts for Intelligent Vision Systems (ACIVS'08)*, 664–673, 2008.

P2 Q. Zhao, M. Xu and P. Fränti, "Knee Point Detection on Bayesian Information Criterion", *IEEE Int. Conf. Tools with Artificial Intelligence (ICTAI'08)*, 431–438, 2008.

P3 Q. Zhao, M. Xu and P. Fränti, "Sum-of-Square Based Cluster Validity Index and Significance Analysis", *Int. Conf. on Adaptive and Natural Computing Algorithms (ICANNGA'09)*, 313–322, 2009.

P4 Q. Zhao, M. Xu and P. Fränti, "Expanding external validity measures for determining the number of clusters", *Int. Conf. on Intelligent Systems Design and Applications (ISDA'11)*, 931–936, 2011.

P5 Q. Zhao, P. Fränti, "Centroid Ratio for Pairwise Random Swap Clustering Algorithm", *Manuscript*.

P6 Q. Zhao, V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Random swap EM algorithm for Gaussian mixtures models", *Manuscript*.

P7 Q. Zhao, V. Hautamäki, I. Kärkkäinen and P. Fränti, "Random swap EM algorithm for finite mixtures models in image segmentation", *IEEE Int. Conf. on Image Processing (ICIP'09)*, 2397–2400, 2009.

Throughout the overview, these papers will be referred to as [P1]–[P7]. The papers have been included in this thesis with permission of their copyright holders.

AUTHOR'S CONTRIBUTION

In [P1–P2], Prof. Pasi Fränti gave direction on the knee point detection of BIC, while the author proposed the methods for solving the problem. The WB-index in [P3] was a continuous work of Dr. Ismo Kärkkäinen on the index *F-ratio* [1]. The F-ratio came from the F-test in ANOVA, although the equation was not exactly the same. The current WB-index formula, on which the author performed a systematic study, originated from Prof. Fränti. The idea, implementation and experiments in [P4] originated from the author, while the idea of proposing a cluster-level validity index in [P5] originated from Fränti. The author constructed the way of calculating the centroid ratio and proposed pairwise random swap clustering, employing this within the algorithm, and carried out the experiments and analysis of the results. In [P6–P7], employing a random swap in the EM algorithm was proposed by Fränti. Dr. Ville Hautamäki and Kärkkäinen followed up on the idea and made the main implementation, and then the author took over the development and experiments. The analysis of the results was performed by all of the authors together.

In [P1–P7], the author was responsible for most of the writing.

List of Symbols

X	set of data points	7
$\mathbb{R}^D$	D-dimensional space	7
D	dimension	7
N	number of data points	7
M	number of clusters	7
C	Clustering result in centroids	8
P	Clustering result in partitions	8
p_i	partition label of $x_i \in X$	8
c_j	jth cluster	8
$\overline{x}$	average of data points in X	16
n_i	size of cluster i	19
$\overline{x^d}$	average of dth dimension of data points in X	18
n_{kd}	size of cluster k in dimension d	18
u_{ik}	membership of the ith data point belonging to the kth cluster	19
Θ	parameter of Gaussian model	44
L	log-likelihood	44
$\mathcal{N}(.\|.)$	Gaussian distribution	44
x_i	ith data point	8
α_j	mixture weight of model j	44
μ_j	mean of model j	44
Σ_j	covariance of model j	44
Θ^{t-1}	parameters estimated in the previous iteration	45
τ_{ij}	posterior probabilities	45
C_{max}	the number of candidates for searching in SMEM	47

Contents

1 INTRODUCTION **1**

2 CONCEPTS **7**

3 CLUSTER VALIDITY **11**
3.1 Review of validity indexes 11
3.2 Internal Validity Index 15
3.2.1 Knee point detection 19
3.2.2 WB-index 22
3.3 External Validity Index 27
3.4 Centroid Ratio 33

4 CLUSTERING ALGORITHMS **37**
4.1 Review of Algorithms 37
4.2 K-means and swap-based clustering 40
4.3 Pairwise random swap clustering 42
4.4 Expectation Maximisation algorithm 44
4.4.1 EM algorithm 44
4.4.2 Split-and-Merge EM 46
4.4.3 Greedy EM and stochastic EM 48
4.5 Random swap EM algorithm 49

5 IMAGE SEGMENTATION **53**
5.1 Clustering algorithms in image segmentation 53
5.2 Cluster validity in image segmentation 56

6 SUMMARY OF CONTRIBUTIONS **59**

7 SUMMARY OF RESULTS **61**

8 CONCLUSIONS **69**

BIBLIOGRAPHY **71**

1 Introduction

Nowadays, the world is full of data – most of which is stored digitally in electronic media, thus providing huge potential for the development of automatic data analysis, classification and retrieval techniques [2].

Cluster analysis is one of the most widely used techniques for exploratory data analysis, with applications ranging from image processing [3, 4], speech processing [5], information retrieval [6, 7] and Web applications [8, 9]. As a basic tool, clustering has been developed and modified for different application fields, providing many clustering algorithms [2, 10–16]. In most cases, the number of clusters is an unknown parameter because clustering is unsupervised and the user has very little knowledge about the data. Thus, the evaluation of different clustering algorithms, and the problem of determining the number of clusters, are important research problems in cluster analysis.

Clustering is defined as the problem of partitioning data points into groups (clusters), such that the points in the same group are similar, while points in different groups are dissimilar [10]. This basic rule guides the design of clustering algorithms and evaluation of clusterings. Most of the currently existing parametric clustering methods partition data into a predefined number of clusters, with a cluster representative corresponding to each cluster, so that a well-defined cost function involving the data and its representatives is minimised [17]. As such, there are three aspects involved in clustering: data, the cost function and the evaluation function.

The cost function in clustering algorithms is used to decide whether the clustering result is suitable for certain kinds of data structures. The *Mean squared error* (MSE)-based cost function, for example, in *K-means*, assumes the clusters are spherical, while model-based clustering, utilising an *EM algorithm* for example, assumes that the data points originate from a *Gaussian mixture model* (GMM).

The covariance parameter in Gaussian models defines the size and direction of the model, which is more general than the model employed in MSE-based methods. A summary of clustering algorithms for different data types, shapes of clusters and other properties is given in [18].

There are many different ways to express and formulate the clustering problem, as each clustering algorithm may provide a different grouping for a data set depending on the cost function used. The categorisation of clustering methods is neither straightforward nor canonical [19], but one option is to classify the methods as hierarchical methods, partitional methods, density-based methods, graph-based methods, grid-based methods and methods for high-dimensional space data. Based on the relationship of each data point to the clusters, the algorithms can also be categorised into *hard (crisp)* and *soft (fuzzy)* clustering algorithms. In hard clustering, each object belongs to one cluster crisply, while each object belongs to each cluster in soft clustering but only to a certain degree.

Hierarchical methods include agglomerative and divisive algorithms. Hierarchical clustering based on linkage metrics results in clusters of proper (convex) shapes, but to avoid problems with non-uniform sized or shaped clusters, clustering using representatives (*CURE*) [20] employs a novel hierarchical clustering algorithm that adopts a middle ground between the centroid-based and all point extremes. A hierarchical clustering algorithm known as *CHAMELEON* [21] measures similarities between two clusters based on a dynamic model. In the clustering process, two clusters are merged only if the inter-connectivity and closeness between the two clusters are highly relative to the internal interconnectivity of the clusters and the closeness of items therein. The algorithm is applicable to all types of data as long as a similarity matrix can be constructed for the data points. A divisive algorithm, known as principal direction divisive partitioning (*PDDP*) [22], bisects data in Euclidean space by employing a hyperplane that passes through the data centroid orthogonally to eigenvector with the largest sin-

gular value. While PDDP concentrates on how to split a cluster, the problem of which cluster to split is also an important consideration.

In partition-based methods, K-means clustering is popular because it is easy to implement and efficient with $O(MN)$ time complexity, where M is the number of clusters and N the size of the data set. Its major downfall is that it is sensitive to the initialisation and may converge to local minima. Another problem is the settings of the number of clusters in K-means clustering. To solve these issues, the K-means algorithm has been well studied by many researchers [23–37]. The *swap-based clustering algorithm* [25,38,39] is a local search heuristic used to find optimal centroids, which can be used to improve the solution of the K-means clustering. The *random swap* algorithm (RS), originally called *randomised local search* (RLS) [25], is based on randomisation. Here, a randomly selected centroid is swapped to another randomly selected location in the data space.

The expectation maximisation (EM) [40, 41] algorithm is commonly used for the parameter estimation of GMMs. If all covariances are diagonal and equal in each model, K-means is tightly associated with the EM algorithm [42] because EM shares the initialisation problem in common with K-means clustering. Improvements to the EM algorithm have nevertheless been proposed in [15, 41, 43–60]. When only part of the data fits in the memory at one time, on-line EM algorithms can be used in [61,62].

In density-based clustering, clusters are defined as areas of higher density than in the remainder of the data set. The most popular density-based clustering method is *DBSCAN* [63,64]. *OPTICS* [65] can be seen as a generalisation of DBSCAN across multiple ranges, effectively replacing the ϵ parameter in DBSCAN with a maximum search radius. These two algorithms are less sensitive to outliers and can discover clusters of irregular shapes; however, density-based clusterings are not very successful for data sets with large differences in densities.

A number of other clustering algorithms have been developed, such as spectral clustering [10, 66], grid-based clustering [67–70],

ensemble clustering [71–77] and subspace clustering [12].

Cluster validity provides a way of validating the quality of clustering algorithms and the means of discovering the natural structure of data sets. Furthermore, its measures are used to compare the results of different clustering algorithms, as well as two clustering results with different numbers of clusters. Cluster validity can therefore be used for determining the correct number of clusters in a data set. In fact, the clustering procedure and cluster validity have a chicken-and-egg relationship whereby knowing how to define a good clustering criterion requires an understanding of the data, but clustering is one of the principal tools used to help understand the data [72] in the first place.

Many different cluster validity indexes have been proposed and studied [18, 36, 78–84]. In general, they are classified into *internal indexes* and *external indexes*, the former of which are usually based on information intrinsic to the data, while the latter are based on prior knowledge about the data. The problem of determining the number of clusters is solved by finding a knee point among the validity index values of different numbers of clusters in a range, $M = [M_{min}, M_{max}]$. The knee point is the number of clusters with sharp change of the index values. Validity indexes with minimum or maximum value are preferred. However, it is possible that the validity index has several local minimum or maximum points. Thus, knee point detection methods for determining the number of clusters are needed, especially for those indexes with non-obvious knee point.

The focus of this thesis is the study of clustering algorithms and cluster validity indexes. In order to solve the problem of determining the number of clusters, validation measures are studied. Methods for knee point detection on the Bayesian information criterion are proposed [P1, P2] because the original method is too subjective. WB-index, a sum-of-squares based cluster validity index, is introduced and compared systematically to other sum-of-squares based indexes in [P3]. The index takes the minimum value as the optimal number of clusters. While studying external indexes, an extension

of the external index to the internal index by utilising a resampling method is introduced in order to determine the number of clusters [P4]. There is little research on cluster validity at the cluster level, so we introduce in [P5] a novel validity measure called the *centroid ratio*, which has $O(M^2)$ time complexity. Consequently, *pairwise random swap* clustering employing the centroid ratio is introduced. Furthermore, and motivated by the improvements produced by the swap strategy in K-means, we give an improved version of the EM algorithm by employing a random swap strategy in [P6, P7]. The new algorithm (RSEM) provides better and faster optimisation of the cluster models. Finally, a study of RSEM in image segmentation is reported in [P7].

The rest of the thesis is organised as follows. Basic concepts of clustering and cluster validity are summarised in Chapter 2. Cluster validity is discussed in Chapter 3, and the clustering algorithms in Chapter 4. The application of clustering in image segmentation is presented in Chapter 5, while in in Chapter 6, a summary of the contributions of the publications is included. A summary of the main results is given in Chapter 7. Finally, conclusions are drawn and future directions are suggested in Chapter 8. The original research papers are attached at the end of the thesis.

2 Concepts

Given a data set $X \subset \mathbb{R}^D$, with N points $x_i \in X$ in a D-dimensional space, the problem is to group the data set into M clusters at minimum cost, yet with maximum likelihood. Examples of synthetic data sets (S1–S4) [85] with different degree of cluster overlapping can be seen in Fig. 2.1. The two-dimensional data sets consist of 5000 data points and 15 Gaussian clusters.

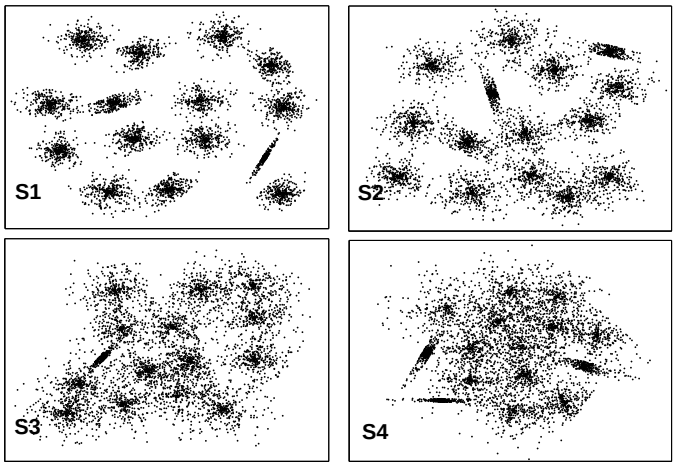


Figure 2.1: A visualisation of data sets S1–S4 [85, 86].

The clusters can be considered as Gaussian models, and the data set as a mixture of Gaussian models (see Fig. 2.2). Among many possibilities, such as *Binomial distribution*, *Poisson distribution* and *Gaussian distribution* for the distribution of the mixture components, the Gaussian is the most popular and practical for mixture models [40]. A Gaussian model (or component) is defined by parameters $\Theta = (\mu, \Sigma)$, where μ represents the mean value and Σ the covariance matrix.

$$\mathcal{N}(X|\Theta) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{-1/2}} \exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right) \qquad (2.1)$$

A GMM is a mixture of Gaussian models by weights α, each of which is a probability in [0,1] summing up to 1.

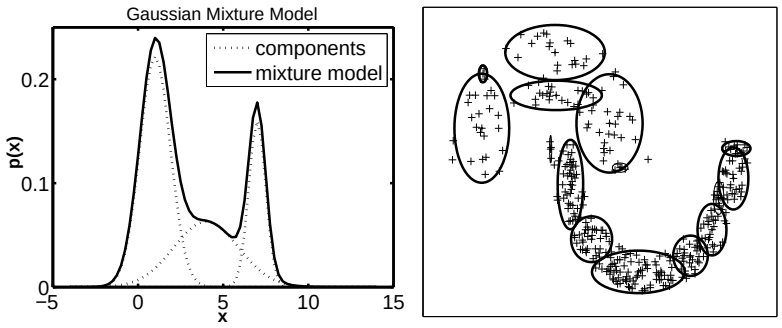


Figure 2.2: A GMM is plotted to approximate one-dimensional data (left). 2-D clusters are represented by Gaussian models (right).

When the covariance matrix Σ is a diagonal matrix and equal in each model, the mean value μ can be represented as the centroid of a cluster in *prototype-based clustering*. We define a set of centroids $C = \{c_1, c_2, ..., c_M\}$ whereby partitions, i.e. cluster labels/memberships for each data point, are defined as $P = [p_{ij}]_{N \times M}$.

$$\sum_{j=1}^{M} p_{ij} = 1; \forall i \in [1, N] \tag{2.2}$$

In hard (crisp) clustering, p_{ij} is either 0 or 1, and $p_{ij} = 1$ for one value of j only. In soft (fuzzy) clustering $p_{ij} \in [0, 1]$ and p_i can have nonzero values in several values of j.

Taking the mean as the centroid of a cluster and MSE as the cost function, for example in the K-means algorithm, the centroid and partition are defined as:

$$c_j \leftarrow (\sum_{p_i=j} x_i) / (\sum_{p_i=j} 1), \forall j \in [1, M] \tag{2.3}$$

$$p_i \leftarrow \underset{1 \le j \le M}{\operatorname{argmin}} \left\| x_i - c_j \right\|^2, \forall i \in [1, N] \tag{2.4}$$

where $\|\cdot\|$ is Euclidean norm.

For given partitions, the optimal set of prototypes consists of the centroids (arithmetic mean) of the clusters and vice versa for a given set of prototypes, where the optimal partition can be obtained by assigning each point to the cluster with the nearest prototype. Thus, partitions and centroids are dual structures, in that if one of them is known, the other one can be determined uniquely. This duality is utilised in the K-means algorithm [2], which finds the nearest local minima for a given initialisation by repeatedly applying these two steps in turn. The two steps are called the *partition step* and *centroid step*, respectively.

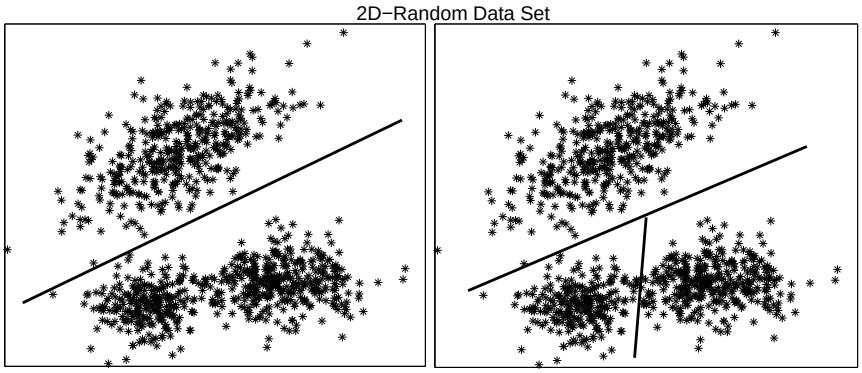


Figure 2.3: The problem of determining the number of clusters. Given the data and partitions on different numbers of clusters, which partitioning is better? How should one evaluate the different partitions?

Cluster Validity includes problems such as measuring the goodness of a clustering algorithm by using different parameter settings, determining the number of clusters (Fig. 2.3) and comparing clustering algorithms (Fig. 2.4).

When the evaluation is based on data set and clustering structures, it is known as an *internal evaluation*, so internal validity indexes are functions of X, C and P. Measures of differences between two clusterings have been found, where one of them is the ground-truth or reference result, and they are considered in this context as *external evaluation*. External indexes are functions of the partitions formed by the clustering algorithm (P_1) and the ground truth

or reference partitions (P_2). An example of a comparison between two clusterings is shown in Fig. 2.4, the difference for which can be obtained either by the difference of two partitions from external indexes or the difference of values from internal indexes.

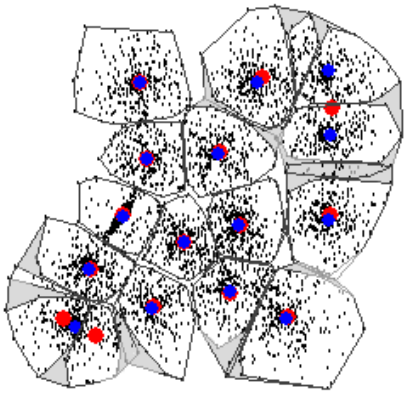


Figure 2.4: Comparison of a random swap clustering solution for the data set S3 against the K-means result. The difference between two solutions is represented by the grey area.

3 Cluster Validity

3.1 REVIEW OF VALIDITY INDEXES

Clustering algorithms with different cost functions give different solutions, and there is no single best choice of the algorithm and the cost function for all possible data sets. The task is therefore to select the best possible clustering method for a data set. For most clustering algorithms, the number of clusters is set as a parameter. However, the number of clusters is initially not available for most data sets, so determining this number is essential. After the correct clustering algorithm and the number of clusters have been selected, evaluating the clustering results on different parameter settings needs to be addressed. These problems are all related to the cluster validity analysis.

Milligan and Cooper [78] presented a comparison of 30 internal validity indexes for hierarchical clustering algorithms, whereas Dimitriadou et al. [79] conducted their comparison for 15 validity indexes in the case of binary data. Meanwhile, a systematic study of 16 external validation measures for K-means clustering is given in [23, 36]. A survey of cluster validation indexes was conducted in [87] for the analysis of post-genomic and other application-specific data.

Since external indexes are based mainly on prior information of the data, e.g. the optimal number of clusters, the indexes are used for choosing the best clustering method for a specific data set. Conversely, internal indexes can be used to choose the best clustering algorithm as well as the optimal number of clusters, without the need for additional information. In practice, prior information regarding data sets is often not available, and internal validation is therefore more useful in general.

Data *resampling* [88] is an alternative approach for statistical validation of clustering results. The resampling is expected to simulate

perturbations of the original data set so as to assess the stability of the clustering results in respect to the sampling variability. The underlying assumption is that the more stable the results are in respect to the simulated perturbations, the more these results are to be trusted [74]. Several resampling techniques such as *bootstrapping*, *jackknife* and *perturbation* [89] are commonly used for generating resamples. The choice of the technique depends on the data and the clustering method used. Once the clustering results have been generated for the resamples, the stability of the partitioning can be obtained. The number of clusters is also estimated, typically based on the maximization or minimization of the stability score from the resampling.

A *gap statistics* method is employed in [90] for estimating the number of clusters by comparing changes in within-cluster dispersion with its expectation under uniform distribution of data as a null hypothesis. Peck et al. [91] developed a bootstrap-based procedure to obtain approximate confidence bounds on the number of clusters in the best clustering, while Ben-Hur et al. [89] presented a method that exploits measurements of the stability of clustering solutions obtained by perturbing the data set. The *Prediction strength* method [92] views clustering as a classification problem, and uses the cross-validation technique in the method. Dudoit and Fridlyand [93] introduced a prediction-based sampling method, *CLEST*, in which a data is first split into two non-overlapping sets – learning and test sets. The learning set is then clustered and a classifier is built using the obtained labels for the points in the training set. The test set is also clustered and the obtained labels from the test set are compared using an external index.

Extensions of resampling approaches to fuzzy clustering are also studied in [94, 95]. A method in [94] determines the number of clusters based on the evaluation of fuzzy partition stability under bootstrap resmapling. An investigation is performed in [95] on whether resampling approaches for hard clustering can be transferred to fuzzy clustering. It turns out that they are applicable to fuzzy clustering as well with certain restrictions.

Determining the number of clusters relies on the cluster validity indexes. In order to determine the optimal number of clusters M^*, other parameters are fixed and parameter M is optimised by the validity indexes. A procedure for determining the optimal number of clusters is shown in Fig. 3.1. Given the data set X, a specific clustering algorithm and a fixed range of number of clusters $[M_{min}, M_{max}]$, the basic procedure involves:

1. Repeat a clustering algorithm successively for the number of clusters M from a predefined range $[M_{min}, M_{max}]$.

2. Obtain the clustering results (partitions P and centroids C) and calculate the validity index value for each.

3. Select the M^* for which the partitioning provides the best result according to the validity index. (see Fig. 3.3).

4. Compare the M^* with external information if available.

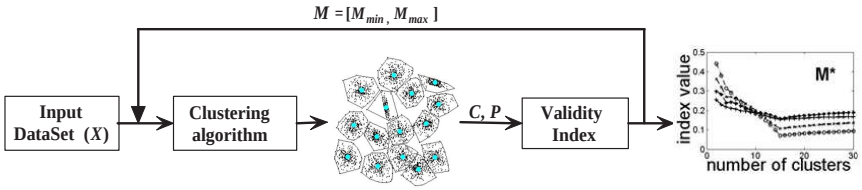


Figure 3.1: Determining the number of clusters in cluster validity analysis.

A knee point is defined as the number of clusters with significant change on the validity index values. Given the range of $M \in [M_{min}, M_{max}]$, the knee point can be the minimum, maximum or points with obvious changes (see Fig. 3.2) of the validity indexes. The points with obvious changes exist in some validity indexes which are monotonously increasing or decreasing with respect to the increasing of the number of clusters. As shown in Fig. 3.3, minimum values of WB-index are the knee points indicating the number of clusters. However, several local minima or maxima of

the validity index may exist. Besides, the elbow points are difficult to locate. Thus, it is meaningful to study knee point detection methods.

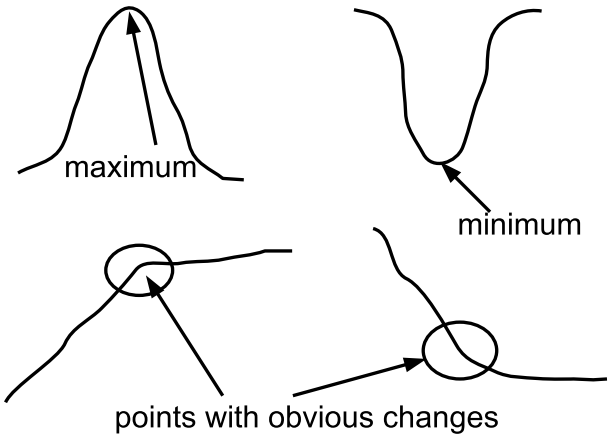


Figure 3.2: Knee points: maximum, minimum and points of curves with obvious changes.

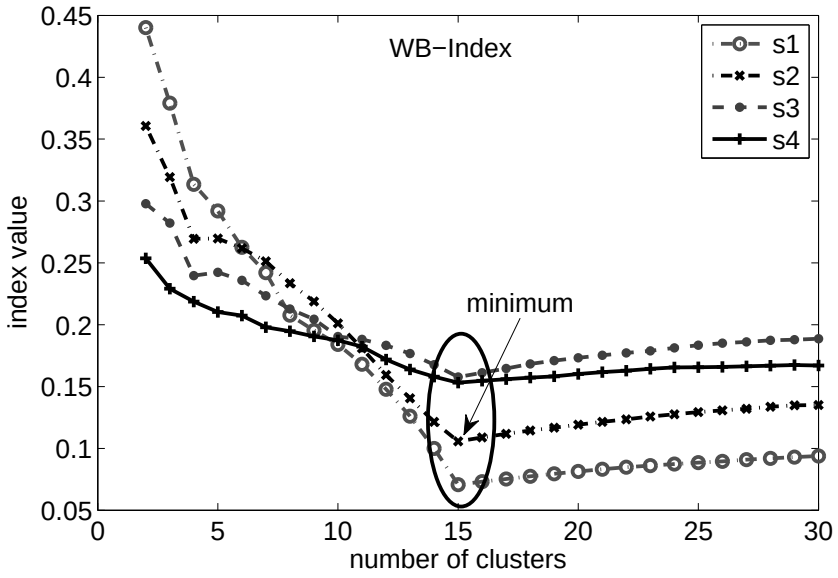


Figure 3.3: A graph of the number of clusters vs. the WB-index [P3] on the data sets S1–S4.

It is meaningless to set $M_{min} = 1$ because a test of uniformity (deficiency of randomness) is enough in this case; the clustering algorithm has no effect when all data are in a single cluster, so it is usual to set $M_{min} = 2$. A rule of thumb is to let $M_{max} \sim (N/2)^{1/2}$ [96] when there is no prior information about the data.

In most validation studies, 2D data sets have been used because it is then possible to verify visually the validity of the results, i.e. how well the clustering algorithm discovers the clusters in the data set. For multidimensional data with more than three dimensions, visualisation is a non-trivial and highly difficult task [97]. Then artificially generated data sets and high dimensional data sets with prior information [85, 98] are commonly used in validation experiments.

A visual cluster validation tool *CVAP* [83] based on Graphical user interface (GUI) provides four external validity indexes, 14 internal validity indexes and five clustering algorithms (K-means, *partitioning around medoids* [99] (PAM), hierarchical clustering, *self-organising map* [100] (SOM) and *affinity propagation* [101]). CVAP is designed for the validity evaluation of clustering solutions, estimating the number of clusters and performance comparisons between clustering algorithms. An **R** package *clValid* [102] contains functions for validating the results of a clustering analysis.

3.2 INTERNAL VALIDITY INDEX

A good clustering algorithm generates clusters with high intra-cluster homogeneity, good inter-cluster separation and high connectedness between neighboring data points [87]. One category of internal indexes is based on these properties, and examples of this type are given by *Dunn* [103], *Davies and Bouldin* [104], *Xie-Beni* [105], *Calinski and Harabasz* [106] and *S_Dbw* [107]. Another category is based on whether the internal indexes are applied to hard (crisp) or soft (fuzzy) clustering. A review of fuzzy cluster validity indexes is available in [108].

The sum-of-squares-based indexes are based on *sum-of-squares*

within cluster (SSW) or *sum-of-squares between clusters* (SSB) values. Sum-of-squares-based indexes such as *Ball and Hall* [109], *Hartigan* [110], *Calinski and Harabasz* [106] and *Xu* [111] are compared in [78, 79].

Examples of other popular indexes are given in [103–105, 107, 112]. Dunn-type indexes [103] are based on the inter-cluster distance and diameter of cluster hyperspheres. The Dunn index is sensitive to outliers, whereas the Davies and Bouldin index is defined by the average of cluster evaluation measures for all the clusters. Xie-Beni [105] adopted the minimum distance between any pair of clusters and the global average of distances between each data object and clusters as inter- and intra-cluster distances, respectively. S_Dbw [107] replaced the total separation with the density of data points in the middle of two clusters and omitted the weighting factor. An improved version of silhouette coefficient (SC) in [113] reduces the computation time on distance calculations by decreasing the number of addition operations.

A model selection method called the *Bayesian information criterion* (BIC) [114, 115] has been applied in model-based clustering, but it can be adapted to partition-based clustering [35], too. Another popular method for determining the number of components and simultaneously learning parameters of GMMs, is to use *Dirichlet Process Mixture Model* (DPMM) [116] framework.

A summary of internal validity indexes is listed in Table 3.1.

Table 3.1: Formulas for internal indexes

Name	Formula
SSW	$SSW_M = \sum_{i=1}^{N} \left\|x_i - c_{p_i}\right\|^2$
SSB	$SSB_M = \sum_{i=1}^{M} n_i \left\|c_i - \overline{X}\right\|^2$
Calinski-Harabasz [106]	$CH = \frac{SSB_M/(M-1)}{SSW_M/(N-M)}$
Ball&Hall [109]	$BH = SSW_M/M$
Xu-index [111]	$Xu = D\log_2(\sqrt{SSW_M/(DN^2)}) + \log M$

Krzanowski-Lai [117]	$diff_M = (M-1)^{2/D} SSW_{M-1} - M^{2/D} SSW_M$ $KL = \lvert diff_M \rvert \,/\, \lvert diff_{M+1} \rvert$
Hartigan [110]	$H = (\frac{SSW_M}{SSW_{M+1}} - 1)(N - M - 1)$ or: $H = \log_2 (SSB_M / SSW_M)$
Dunn's index [103]	$d(c_i, c_j) = \min_{x \in c_i, x' \in c_j} \left\lVert x - x' \right\rVert^2$ $diam(c_k) = \max_{x, x' \in c_k} \left\lVert x - x' \right\rVert^2$ $Dunn = \frac{\min_{i=1}^{M} \min_{j=i+1}^{M} d(c_i, c_j)}{\max_{k=1}^{M} diam(c_k)}$
Davies&Bouldin [104]	$R_{ij} = \frac{S_i + S_j}{d_{ij}}, i \neq j$ where: $d_{ij} = \left\lVert c_i - c_j \right\rVert^2$ $S_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \left\lVert x_j - c_i \right\rVert^2$ and, $R_i = \max_{j=1,...,M} R_{ij}, i = 1, ..., M$ $DBI = \frac{1}{M} \sum_{i=1}^{M} R_i$
R-square [118]	$SSW = \sum_{\substack{k=1,...,M \\ d=1,...,D}} \sum_{i=1}^{n_{kd}} (x_i - \overline{x^d})^2$ $SST = \sum_{d=1,...,D} \sum_{i=1}^{n_d} (x_i - \overline{x^d})^2$ $RS = \frac{SST - SSW}{SST}$

RMSSTD [118]	$RMSSTD = \frac{\sum_{k=1,\ldots,M;\, d=1,\ldots,D} \sum_{i=1}^{n_{kd}} (x_i - \overline{x^d})^2}{\sum_{k=1,\ldots,M;\, d=1,\ldots,D} (n_{kd}-1)}$
SC [113]	$a(x_i) = \frac{1}{n_m - 1} \sum_{j=1, j\neq i}^{n_m} \left\|x_i - x_j\right\|^2_{x_i, x_j \in c_m}$ $b(x_i) = \min\{\sum_{t\neq m} \|c_t - c_m\|^2\}_{x_i \notin C_t}$ $s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))}$ $SC = \frac{1}{N} \sum_{i=1}^{N} s(x_i)$
S_Dbw [107]	$stdev = \sqrt{\sum_{i=1}^{M} \|\sigma(c_i)\|}/M$ $den(c) = \sum_{l=1}^{n_{ij}} f(x_l, c), x_l \in c_i \cup c_j \subseteq X$ $f(x,c) = \begin{cases} 0 & \text{if } d(x,c) > stdev \\ 1 & \text{otherwise} \end{cases}$ $Scat(M) = \frac{1}{M} \sum_{i=1}^{M} \|\sigma(c_i)\| \,/\, \|\sigma(X)\|$ $Dens\_bw(M) = \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} \frac{den(c_{ij})}{max(den(c_i), den(c_j))}}{M(M-1)}$ $S\_Dbw = Scat(M) + Dens\_bw(M)$
BIC [112]	$BIC = L * N - \frac{1}{2} M(D+1) \sum_{i=1}^{M} \log(n_i)$
Xie-Beni [105]	$XB = \frac{\sum_{i=1}^{N} \sum_{k=1}^{M} u_{ik}^2 \|x_i - C_k\|^2}{N \min_{t\neq s}\{\|C_t - C_s\|^2\}}$

Partition Coefficient [119]	$PC = \sum_{i=1}^{N} \sum_{k=1}^{M} u_{ik}^2 / N$
Partition Entropy [119]	$PE = -(\sum_{i=1}^{N} \sum_{k=1}^{M} u_{ik} \log(u_{ik}))/N$

3.2.1 Knee point detection

BIC has been widely used for determining the number of components (clusters) in model-based clustering, but it can be reformulated into partition-based clustering as follows:

$$BIC = \sum_{i=1}^{M} (n_i \log \frac{n_i}{N} - \frac{n_i \times D}{2} \log(2\pi) - \frac{n_i}{2} \log \Sigma_i - \frac{n_i - M}{2}) - \frac{1}{2} M \log N \tag{3.1}$$

and

$$\Sigma_i = \frac{1}{N - M} \sum_{j=1}^{n_i} \left\| x_j - c_i \right\|^2 \tag{3.2}$$

where c_i represents the ith cluster, n_i the size of it and x_j the jth point in cluster c_i.

An example of BIC values for partition-based clustering on data S1–S4 is demonstrated in Fig. 3.4. As a partition-based clustering algorithm, the RS clustering [25] is employed in the experiment. The first decisive local maxima of the BIC values is used for determining the number of clusters in [112]; however, the selection of a local maximum is subjective and it is difficult to choose among them when there are several local maxima. The choice of M_{max} affects the number of local minima and maxima, which in turn affects the number of clusters determined.

The second successive difference between index values (Fig. 3.4) can be used for knee point detection, although this approach only reflects local information. The second successive difference (SD) of

BIC values is defined as:

$$SD = BIC(M-1) + BIC(M+1) - 2 * BIC(M) \qquad (3.3)$$

where $M \in [M_{min}+1, M_{max}-1]$.

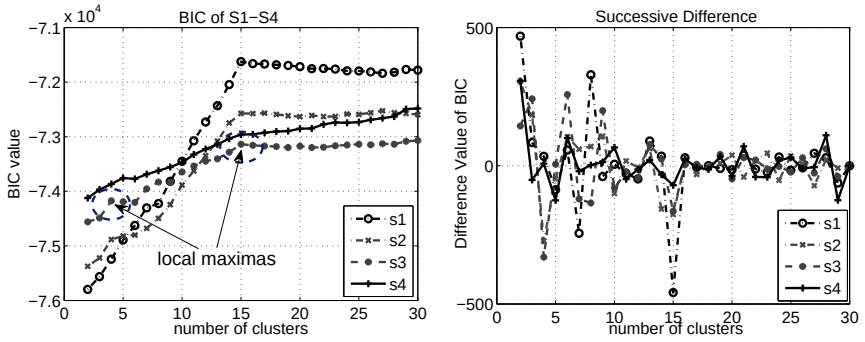


Figure 3.4: Number of clusters vs. BIC on S1–S4 and its second successive difference.

Other methods, such as the L-method [120], have been proposed to find the knee point of the validity index curve by examining the boundary between the pair of straight lines that most closely fit the curve in hierarchical/segmentation clustering. More general methods should be used based on the global trend of the curve.

To improve the BIC index and produce more reliable results on the determined number of clusters, two methods in line with the knee point detection are proposed in [P1, P2].

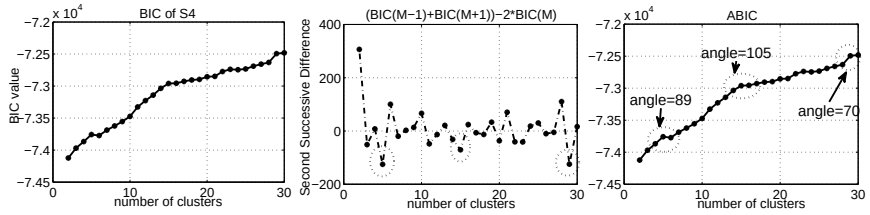


Figure 3.5: Illustration of the angle-based BIC [P1] method used on data set S4. The original BIC (left), the second successive difference of the BIC (middle) and the angles of the local changes (right) are indicated.

In [P1], we use the angle property of a curve. As seen in Fig. 3.5, we calculate the second successive difference of the original BIC

values and detect l locally significant changes by finding the first l minimum values in the successive difference. Here, $l \leq M/2 - 1$ because at least two points can generate one peak in a curve. We sort the detected local minimum values in a decreasing order and then start from the points with bigger peak and calculate their angle. The value of M with a maximum angle is determined as the knee point, which indicates the global trend of the curve.

Another graphical knee point detection method, called *DiffBIC*, is proposed in [P2]. We first normalise the value of BIC to the range $[M_{min},M_{max}]$ giving C_1, C_m is then calculated as the average of C_1 values over the M clusters and C_2 is finally a normalised value of C_m in the range $[M_{min},M_{max}]$.

$$\begin{aligned} C_1 &= (M_{max} - M_{min})(BIC - BIC_{min})/(BIC_{max} - BIC_{min}) \\ C_m &= C_1/M \\ C_2 &= (M_{max} - M_{min})(C_m - C_{m_{min}})/(C_{m_{max}} - C_{m_{min}}) \end{aligned} \tag{3.4}$$

We consider two cases, where the original BIC curve has either a globally increasing trend (case1) or a decreasing trend (case2). We define:

$$DiffBIC = \begin{cases} (C_1 + C_2)/2, & \text{for case1} \\ |C_1 - C_2|/2, & \text{for case2} \end{cases} \tag{3.5}$$

The range $[M_{min},M_{max}]$ is user-defined. M_{max} is assumed to be large enough so that we can refine the range for searching the optimal M value by resetting M_{max}. The operation is called the *max refinement* on DiffBIC (see Fig. 3.6). There will be intersections across the C_1 and DiffBIC values because of the normalisations taking place whenever the trend of the original BIC is increasing or decreasing. The positions of the intersection are affected by the setting of M_{min} and M_{max}, and we assume that M_{max} is large enough to contain M^*. With the assumption that $M_{max} \geq M^*$, the first intersection $M = max'$, where $max' \neq M_{min}$ and $max' > M^*$ exist. The value of max' can be thought of as a refinement to M_{max}, with which the range of M can be reduced to $[M_{min}, max']$.

There are two reasons for the max refinement operation. First, the original range setting is arbitrary and the refined range is a

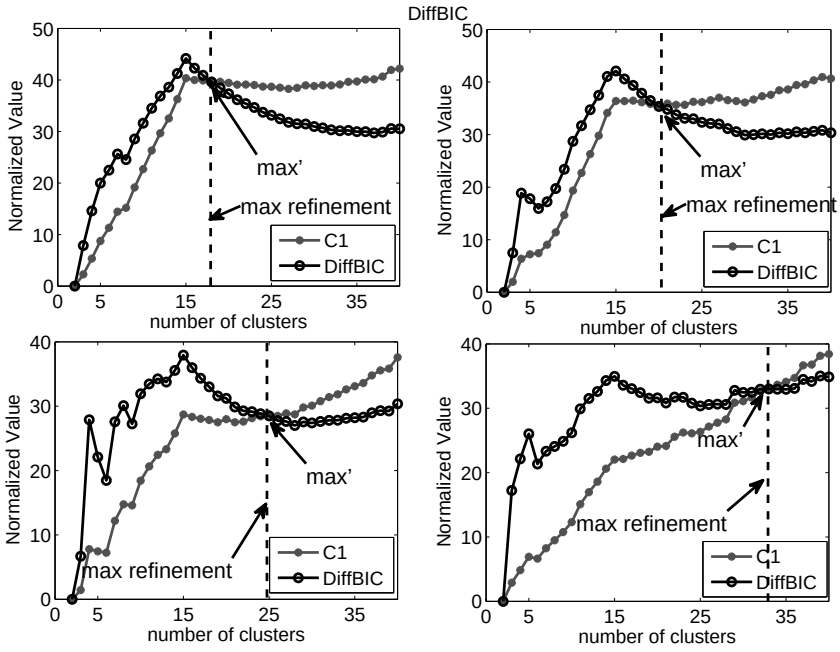


Figure 3.6: An illustration of DiffBIC [P2] for data sets S1–S4 with the RS clustering algorithm. The normalised BIC is represented as C_1 in this context.

smaller range that still contains the optimal value of M. Second, the BIC exhibits a monotonic trend with the number of clusters, so the points after the intersection have less information. Since there exist several local maxima of BIC index, the max refinement step shrinks the search range, which helps the index to produce a more accurate decision.

The methods in [P1, P2] give a direction on the knee point detection of the BIC, which can also be a reference for other validity indexes.

3.2.2 WB-index

The sum-of-squares within (SSW) clusters is a commonly used measure of compactness, while the sum-of-squares between (SSB) clusters is a measure of separation. Sum-of-squares-based indexes (see

Table 3.1) are mainly functions of M, N, D, SSW and SSB, and they usually have a so-called elbow phenomenon. As seen in Fig. 3.7, knee points are minimum and maximum values in Xu-index and Calinski-Harabasz index. However, Ball & Hall and Hartigan indexes have unclear knee points. Thus, knee point detection is needed for them. The second successive difference is commonly used for the knee point detection.

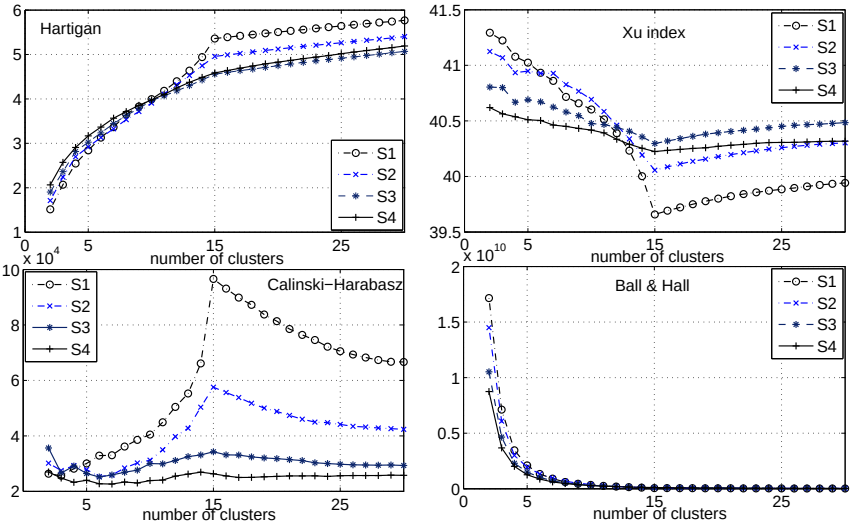


Figure 3.7: A graph of the number of clusters vs. sum-of-squares-based indexes on S1–S4. Knee point detection is needed for Hartigan and Ball & Hall indexes.

We propose a WB-index in [P3]:

$$WB(M) = M \times SSW/SSB \tag{3.6}$$

Let us assume that cluster i has n_i points and we take an average data point (or representative data point) x_i in cluster c_i (see Fig. 3.8). The within-cluster variance for cluster i (W_i) can then be reformulated as:

$$W_i = n_i \left\| x_i - c_i \right\|^2, i \in [1, M] \tag{3.7}$$

$$B_i = n_i \left\| \overline{X} - c_i \right\|^2, i \in [1, M] \tag{3.8}$$

$$\frac{SSW}{SSB}(M) = \frac{\sum_{i=1}^{M} W_i}{\sum_{i=1}^{M} B_i} = \frac{W_1 + W_2 + ... + W_M}{B_1 + B_2 + ... + B_M} > 0 \tag{3.9}$$

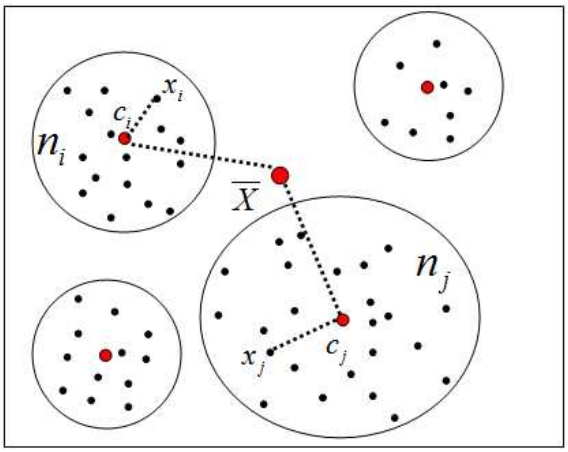


Figure 3.8: The calculation of W_i and B_i.

With increment of one cluster, the difference of SSW/SSB can be written as:

$$\begin{aligned}
\Delta\frac{SSW}{SSB}(M) &= \frac{SSW}{SSB}(M-1) - \frac{SSW}{SSB}(M) \\
&= \frac{\sum_{i}^{M-1} W_i}{\sum_{i}^{M-1} B_i} - \frac{\sum_{i}^{M-1} W_i + W_M}{\sum_{i}^{M-1} B_i + B_M} \\
&= \frac{(\sum_{i}^{M-1} W_i)(\sum_{i}^{M-1} B_i + B_M) - (\sum_{i}^{M-1} B_i)(\sum_{i}^{M-1} W_i + W_M)}{(\sum_{i}^{M-1} B_i)(B_M + \sum_{i}^{M-1} B_i)} \\
&= \frac{B_M \sum_{i}^{M-1} W_i - W_M \sum_{i}^{M-1} B_i}{(\sum_{i}^{M-1} B_i)(B_M + \sum_{i}^{M-1} B_i)}
\end{aligned} \tag{3.10}$$

Since $\sum_{i}^{M-1} W_i$ is monotonously decreasing and $\sum_{i}^{M-1} B_i$ is monotonously increasing with respect to increasing M, $\Delta SSW/SSB(M)$ is then monotonously decreasing also. This indicates that the decrement of SSW/SSB from cluster size $M-1$ to M is larger than that from M to $M+1$. i.e., the decrement is decreasing with increasing M (see Fig. 3.9). When the decrement degree of $\Delta SSW/SSB$ is larger than linear increment of M at the beginning, WB is decreasing until $M^* \geq M_{min}$. A special case is that WB is increasing for all M when $M^* = M_{min}$. Thus, there exists M^* such that $WB(M) \geq WB(M^*)$ for $M \leq M^*$ and $WB(M) < WB(M^*)$ for $M > M^*$. The number of clusters is determined by the minimum value of WB-index. Result of WB-index on data S1–S4 is shown in Fig. 3.3. Although SSW decreases monotonically with increasing M, WB-index has a U-shape with a clear minima at $M = 15$.

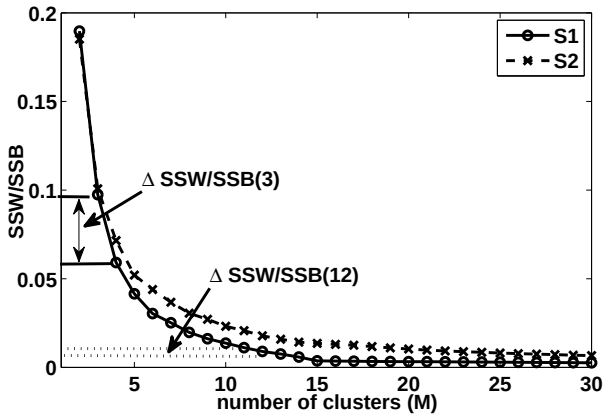


Figure 3.9: Values of SSW/SSB are decreasing when M is linearly increasing. Values of $\Delta(SSW/SSB(M))$ as a function of M are decreasing also.

We develop two approaches for assessing the statistical significance of the proposed method. One approach analyses the variability of each index value by using the quartile range. Quantiles can be used to characterize data with unknown theoretical distribution. With the same input parameter settings, we fix the number of clusters and run the clustering algorithm $B = 100$ times to get

a distribution of WB-index values on the same number of clusters. Then the fifth and 95th percentiles of WB-index values are used to get a 90% probability range of the index. The fifth percentile is obtained by sorting the inde values and taking the fifth value in the order. The 90% probability intervals with RS and K-means clustering are shown respectively in Fig. 3.10, in which the dash line is the boundary of the range.

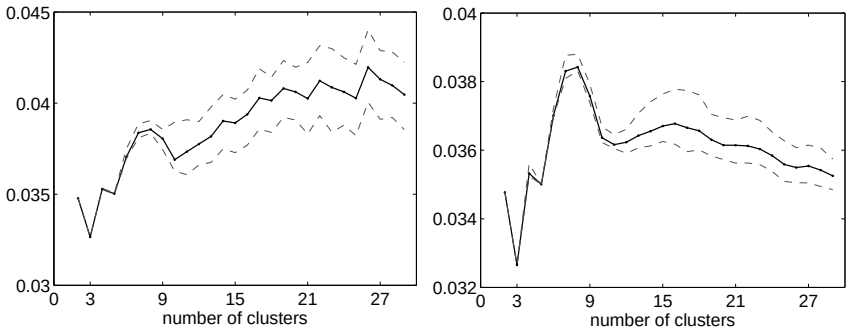


Figure 3.10: A 90% probability interval of the WB-index with RS and K-means clustering on data set Iris.

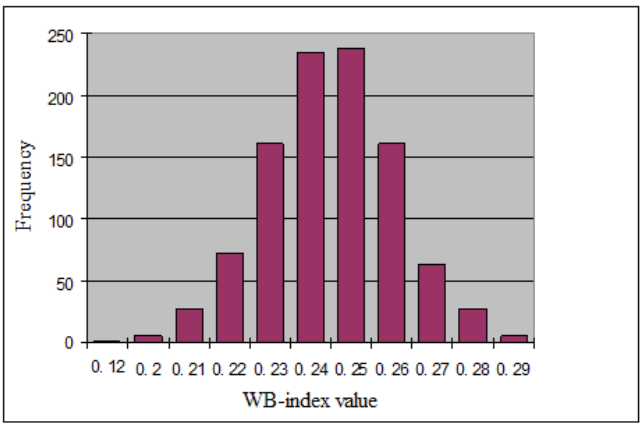


Figure 3.11: Distribution of the WB-index values on Iris data set ($M = 3$) for 1000 permutations of the partitions with RS clustering. The WB-index value with original partitions is very extreme referring to this distribution (WB = 0.03).

Our second approach estimates the certainty of the WB-index through a resampling method. For analysing the certainty of the index, we take a partition P and permutate it $B = 1000$ times, getting a set of partitions $\{P^*\}$. The set of index values $\{WB^*\}$ is calculated on $\{P^*\}$ and the certainty is estimated by counting the probability frequently that $WB^* \leq WB$.

$$Prob = \frac{No.(WB^* \leq WB)}{B} \tag{3.11}$$

The smaller the probability *Prob* is, the more certainty the method obtains. It is not practical to calculate all possible permutations because of the time involved. An example of the certainty analysis on the WB-index is shown in Fig. 3.11.

3.3 EXTERNAL VALIDITY INDEX

External validity indexes are preferable when ground-truth labels are available [121]. The ground-truth consists of class labels assigned to each data point. The ideal clustering is selected based on how well the cluster labels produced by the algorithm match to the ground-truth labels. External measures are used to compare the similarity of the two clustering results. A study of 16 external indexes for K-means clustering was conducted in [36]. The measures are categorised into *pair-counting*, *set-matching* and *information theoretic* in [122]. The common basis of the indexes is that their computations are all based on a contingency table [36] (see Table 3.2). The time complexity of the indexes based on contingency table is $O(M^2 + N)$ for hard partitions.

To construct a contingency table, consider a data set with N points, and suppose we have two partitions $P = \{P_1, P_2, ..., P_M\}$ of M clusters and $G = \{G_1, G_2, ..., G_{M'}\}$ for M' clusters. Denote n_{ij} the number of common points in cluster P_i and G_j. Then the contingency table of P and G is a matrix of the n_{ij}-numbers.

Representatives of pair-counting measures, *Rand Index*, *adjusted Rand Index*, *Jaccard coefficient* and the *Fowlkes and Mallows index* [123] are based on counting the pairs of points on which two clusterings

Table 3.2: Contingency table between two clustering partitions P and G.

	G_1	G_2	...	$G_{M'}$	$\sum$
P_1	n_{11}	n_{12}	...	$n_{1M'}$	$n_{1\cdot}$
P_2	n_{21}	n_{22}	...	$n_{2M'}$	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$
P_M	n_{M1}	n_{M2}	...	$n_{MM'}$	$n_{M\cdot}$
$\sum$	$n_{\cdot 1}$	$n_{\cdot 2}$	...	$n_{\cdot M'}$	n

agree or disagree. Rand Index (Fig. 3.12) is a well-known index of this class. However, its adjusted form is more commonly used [24, 124] because the Rand Index lies within the narrower range of $[0.5, 1]$ in practice. The adjusted Rand Index (ARI) [122] puts the expected value at zero, which gives a more dynamic range $[0, 1]$.

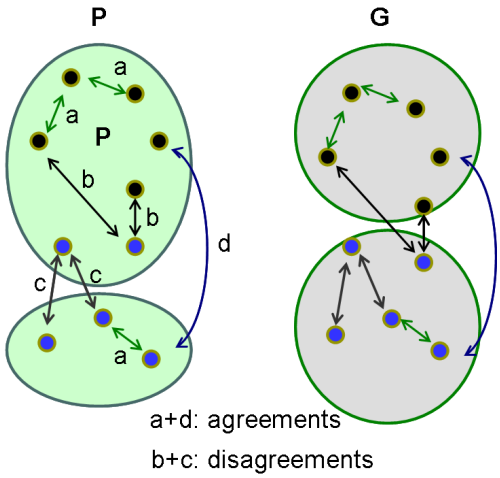


Figure 3.12: A visual explanation of the Rand Index. (a): a pair of points that belong to the same cluster in P and G; (b): belong to the same cluster in P but not in G; (c): belong to the same cluster in G but not in P; (d): are in different clusters in P and G.

Let $p_{ij} = n_{ij}/n, p_i = n_{i\cdot}/n, p_j = n_{\cdot j}/n$. A list of commonly used external indexes is shown in Table 3.3.

Table 3.3: Formulas for external indexes

Name	Formula
Entropy [125]	$E = -\sum_i p_i(\sum_j p_{ij}/p_i \log(p_{ij}/p_i))$
Purity [125]	$P = \sum_i p_i(\max_j p_{ij}/p_i)$
F-measure [126]	$F = \sum_j p_j \max_i [2\frac{p_{ij}}{p_i}\frac{p_{ij}}{p_j}/(\frac{p_{ij}}{p_i}+\frac{p_{ij}}{p_j})]$
Variation of Information [127]	$VI = -\sum_i p_i \log p_i - \sum_j p_j \log p_j - 2\sum_i\sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j}$
Mutual Information [128]	$MI = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j}$
Rand index [129]	$RI = \frac{[\binom{n}{2} - \sum_i \binom{n_{i\cdot}}{2} - \sum_j \binom{n_{\cdot j}}{2} + 2\sum_{ij}\binom{n_{ij}}{2}]}{\binom{n}{2}}$
Adjusted Rand index [122]	$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}] - [\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}]/\binom{n}{2}}$
Jaccard [130]	$J = \frac{\sum_{ij}\binom{n_{ij}}{2}}{[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2} - \sum_{ij}\binom{n_{ij}}{2}]}$
Fowlkes and Mallows [131]	$FM = \sum_{ij}\binom{n_{ij}}{2}/\sqrt{\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}}$
Hubert Γ statistic [132]	$\Gamma = \frac{\binom{n}{2}\sum_{ij}\binom{n_{ij}}{2} - \sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}}{\sqrt{\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}[\binom{n}{2} - \sum_i \binom{n_{i\cdot}}{2}][\binom{n}{2} - \sum_j \binom{n_{\cdot j}}{2}]}}$
Minkowski score [133]	$MS = \frac{\sqrt{\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2} - 2\sum_{ij}\binom{n_{ij}}{2}}}{\sqrt{\sum_j \binom{n_{\cdot j}}{2}}}$
Goodman-Kruskal [134]	$GK = \sum_i p_i(1 - \max_j \frac{p_{ij}}{p_i})$

Two topics can be discussed on external indexes – the first is their extension from hard partitions to soft partitions and the second is the use of external indexes in absence of any ground-truth information.

A fuzzy extension of the Rand index has been introduced in [135]. Other measures such as the adjusted Rand index, Jaccard

coefficient and Fowlkes and Mallows index have also been derived from the same formulation; however, they have a high time complexity of $O(M^2N^2)$. A pioneer solution for fuzzy clustering [136] reduces the complexity significantly to $O(M^2N)$.

In clustering, prior knowledge of the data is usually not available, but resampling methods can be used to overcome this difficulty. The Rand index was extended to calculate pairwise stability [24], which is calculated as the variability of the clustering results by resampling the original data or by multiple initialisations. In [123], a bootstrapping-based measure is proposed. The clustering algorithm is interpreted as a statistical estimator and external indexes are then used for comparing the partitions. Bootstrap resampling has been utilised in evaluating fuzzy partition stability in [94], and its fuzzy extension was introduced in [135], but these methods lead to high time complexity in general.

An extension of external indexes for both hard and soft partitions with no ground-truth (see Algorithm 1) is introduced in [P4]. First, we perform a state-of-the-art sub-sampling algorithm [137] with $O(N)$ time complexity on the original data set X to reduce the number of data points X_s.

Then the procedure in Fig. 3.1 (see page 13) is performed to get the number of clusters. The algorithm consists of two parts: first is to use a reference partition in calculating external validity index, and the second is to apply a resampling method for statistical validation of clustering results.

The reference partition G is generated by an assumption that the data points are indexed subsequently such that for example the reference partition is that the first 100 data points belong to one cluster and the second 100 points into another cluster when a data set of 200 points contains two clusters. Let $c = \lfloor N/M \rfloor$, in which case reference partition $G_{N\times M}$ is generated by:

$$[G]_{ij} = \begin{cases} 1, if \quad i > (j-1) \times c \quad \& \quad i < j \times c + 1 \\ 0, otherwise \end{cases} \tag{3.12}$$

Similar as the resampling method in [90], the proposed method

compares the external index value with its expectation under uniform distribution.

$$I = E_B\{I_u\} - I_x \tag{3.13}$$

where E_B denotes the expectation of validity index values under a sample of size B from the uniform distribution, i.e., $E_B\{I_u\} = \sum_{b=1}^{B} I_u/B$. The samples are generated uniformly for each dimension of data independently over the range of the original data set at that dimension. The external index value I_x is calculated between the partition from the sub-sampled data X_s (P_x) and the reference partition (G), and I_u represents a set of the external index values between the partition from the uniform samples (P_u) and the reference partition (G).

Input: $X = \{x_1, x_2, ..., x_n\}$, M_{max}
Output: M_{opt}

```
Xs = subsampling(X) ;
for m = 2 : M_max do
    Set reference labels G = [g_ij]_{N×M} ;
    P_x = CLUSTER(Xs) ;
    I_x = ExternalIndex(P_x, G) ;
    for b = 1 : B do
        Generate reference data X_b uniformly ;
        P_u = CLUSTER(X_b) ;
        I_u = ExternalIndex(P_u, G) ;
    end
    I(m) = E_B{I_u} − I_x ;
end
M_opt = min(I(m)) ;
return M_opt
```

Algorithm 1: Pseudocode of the proposed method

The external index involved in the method is the adjusted Rand Index (ARI), however, any other external indexes can also be employed instead. When an efficient fuzzy extension of the external index [136] is employed, the proposed method is applicable for fuzzy

partition. Soft clustering algorithms such as the EM algorithm and Fuzzy C-means (FCM) [138] are studied in the method. For hard partition, K-means and hard-cut of EM and FCM are used.

An example of the result from the proposed method is demonstrated in Fig. 3.13. The sub-sampled data of S2 and the partition from FCM on the data is displayed. The sub-sampling method reduces 38%-78% of the running time in the experiment and the fuzzy extension of the external index affects little on the running time (see the running time in Fig. 3.13). For determining the number of clusters, the proposed method works well on real data sets and small Gaussian-distributed data sets. In general, it has better performance on hard partitions than on soft ones.

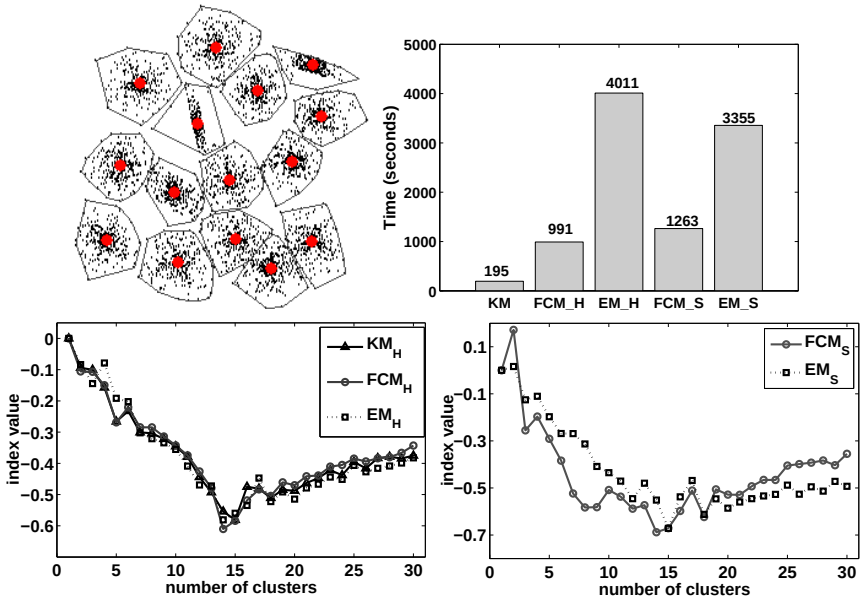


Figure 3.13: Clustering on the sub-sampled data set of S2 from FCM, a comparison on the running time of the proposed method on different soft and hard clusterings and the index value of the proposed method (hard and soft clustering respectively) on the increasing number of clusters. FCM_H and FCM_H represent the hard partition of FCM, similar for EM_H and EM_H. FCM_S and FCM_S represent the soft partition of FCM, similar for EM_S and EM_S.

3.4 CENTROID RATIO

There is little research on cluster-level evaluation measures, which are based on centroids only. Centroids play an important role in clustering because they reveal the allocation of clusters. In evaluation of clustering, the time complexity of internal and external indexes is usually $O(MN)$ or $O(N^2)$ and utilising centroids only in evaluation reduces the time complexity to $O(M^2)$. A cluster-level evaluation criterion called the centroid ratio is introduced in [P5].

Let $C_1 = \{c_{11}, c_{12}, ..., c_{1M}\}$ and $C_2 = \{c_{21}, c_{22}, ..., c_{2M}\}$ be the centroids of two clusterings C_1 and C_2, respectively.

Definition The *nearest pairing* of two sets of centroids (C_1 and C_2) can be stated in graph-theoretic terms as the minimum matching of a given bipartite graph where nodes correspond to the centroids, edges connect centroids from different clusterings, and edge cost stands for the centroid distance.

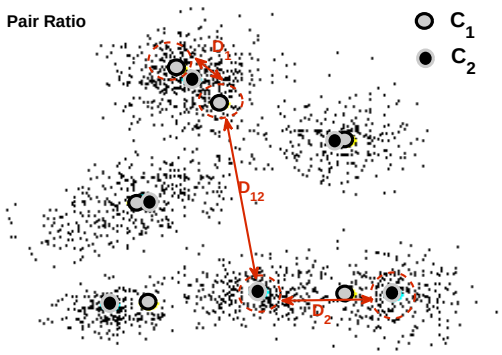


Figure 3.14: Calculation of the pair ratio *for one pair of centroids.*

Definition The *Pair ratio* for centroid i, $PR(i)$, is the degree of matching in terms of distance between centroid i from C_1 and C_2 after the nearest pairing.

The minimum matching in the nearest pairing is solved in a greedy way whereby, for each i, j, where $1 \leq i \leq M$, $1 \leq j \leq M$, we

consider they are paired if c_{2j} is the closest centroid to c_{1i} out of $\{c_{21}, c_{22}, ..., c_{2M}\}$. We thus iterate M times the operations:

$$\begin{aligned} \{i,j\} &= \underset{c_{1i}\in C_1, c_{2j}\in C_2}{\operatorname{argmin}} \left\|c_{1i} - c_{2j}\right\|^2 \\ C_1 &\leftarrow C_1 \backslash \{c_{1i}\} \\ C_2 &\leftarrow C_2 \backslash \{c_{2j}\} \end{aligned} \tag{3.14}$$

For paired centroids $c_{1i} \in C_1$ and $c_{2j} \in C_2$, we define the distances:

$$\begin{aligned} D_1(i) &= \min_{c_{1s}\in C_1} \|c_{1i} - c_{1s}\|^2 \\ D_2(i) &= \min_{c_{2s}\in C_2} \left\|c_{2j} - c_{2s}\right\|^2 \\ D_{12}(i) &= \left\|c_{1i} - c_{2j}\right\|^2 \end{aligned} \tag{3.15}$$

The value of D_{12} is the distance of the matched centroids in two clustering results C_1 and C_2. D_1 is the nearest distance of two centroids in the same set of centroids C_1 and similarly, D_2 is the nearest distance in C_2. The centroids in two clusterings are strictly matched when $D_{12} = 0$. We consider centroid i is stable or correctly located when $D_{12} \leq D_1$ and $D_{12} \leq D_2$. Thus, the *Pair Ratio* for a centroid i of clustering C_1 with respect to C_2 (see Fig. 3.14) is defined by:

$$PR(i) = \frac{D_{12}(i)}{D_1(i)} \times \frac{D_{12}(i)}{D_2(i)} \tag{3.16}$$

A centroid i is considered as stable or correctly located when $PR(i) \leq 1$. For unstable and incorrectly located centroids, $PR(i) > 1$.

Definition Similarity S between the two clusterings C_1 and C_2 is:

$$\begin{aligned} S(C_1, C_2) &= 1 - \sum_{i=1}^{M} \gamma_i / M \\ \gamma_i &= \begin{cases} 1 & \text{if } PR(i) > 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.17}$$

S is in the range of $[0, 1]$, where 1 indicates a complete match of two clusterings, and 0 indicates a complete mismatch.

Definition The *centroid ratio* is involved with the calculations of pair ratio (PR) and similarity (S), where PR finds incorrectly located centroids and the S value indicates the similarity of the two clusterings.

4 Clustering Algorithms

In this chapter, we give a closer look at the partition-based (or centroid-based) clustering represented by K-means, random swap based clustering, and the EM algorithm and its variants in model-based clustering.

4.1 REVIEW OF ALGORITHMS

As a representative of the centroid-based clustering algorithms, K-means suffers from an initialisation problem: the result of the clustering depends on the initial setting of the centroids. A common way of addressing this problem is to run K-means multiple times with a different set of randomly chosen initial centroids [28] and to choose the best solution as a result. We call this variant *repeated K-means* (RKM). For different data sets, the correct number of repetitions for RKM is an empirical choice. *K-means++* [32] chooses initial centroids (seeds) for K-means. This improves both the speed of the computation and the quality of the clustering. In addition, it is $\Theta(\log M)$-competitive with optimal clustering [32], i.e. $E[\phi] \leq 8(\log M + 2)\phi_{OPT}$, where ϕ indicates the cost function and M represents the number of clusters. Other methods based on the selection of initial cluster centroids are proposed in [33, 34].

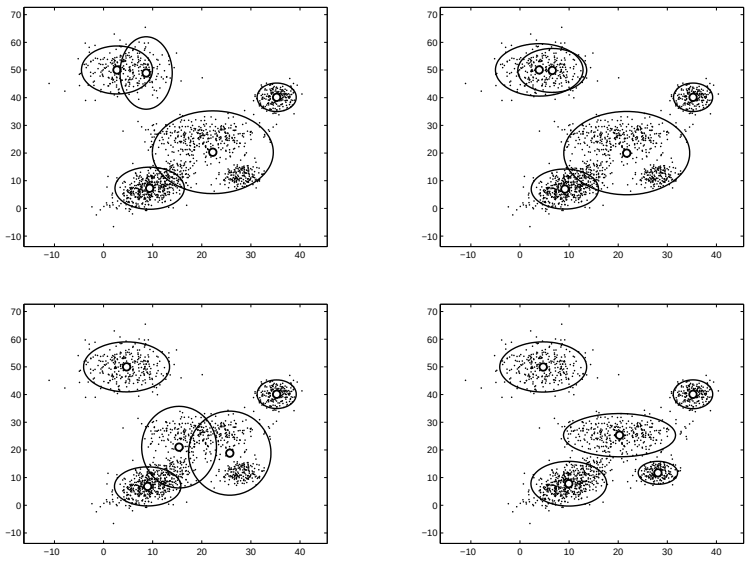
Several methods based on stochastic global optimisation have been developed, such as *simulated annealing* [29] and *genetic algorithms* [30], although these methods have not gained wide acceptance because of their high time complexity. An accelerated algorithm on K-means is introduced in [31, 139]. The algorithm avoids unnecessary distance calculations by applying triangle inequality in two different ways and by keeping track of the lower and upper bounds for distances between points and centres in [139]. Kd-tree is used in [31] for storing the data points, while a *global K-means* algorithm (GKM) [27] is an incremental approach that dynamically

adds one cluster centroid at a time through a deterministic global search procedure. The search procedure consists of N (number of data points) executions of the K-means algorithm from suitable initial positions. Experimental results show that the GKM algorithm considerably outperforms the conventional K-means algorithm.

The stability of clustering has been proposed as a measure of the quality for clustering algorithms, and the stability of K-means clustering is analysed in [24, 140, 141]. Variants are available for improving the K-means by combining the problem of determining the number of clusters. For example X-means [35] searches the space for cluster locations and the number of clusters is optimised by the *Bayesian Information Criterion* (BIC) efficiently.

Clustering can be found by a sequence of centroid swaps and by fine-tuning their exact location by K-means [25, 38]. In each swap-based clustering iteration, a swap strategy is employed to search for a pair of centroids, of which one is to be removed and the other is inserted to lead to an improved solution. If this gives an improved solution, the swap is made and the procedure is iterated after a fine-tuning step by K-means. Swap-based clustering is simple to implement and produces good quality results independent of the initialisation.

In model-based clustering, the expectation maximisation (EM) algorithm [40, 41] is well studied. It iteratively refines the *maximum likelihood* (ML) parameter estimation by first calculating the expectation of the posterior of the latent variables (as opposed to observable variables) while keeping the parameters fixed, at which time the algorithm finds the maximum of the parameters. This iterative process is guaranteed to converge [40, 41]. However, the EM algorithm shares the initialisation problem with the K-means algorithm, as both are hill climbing algorithms, as shown in Fig. 4.1. An initial set of parameters is needed for the initialisation, but unfortunately not all initial values of the parameters lead to the same unique solution when the algorithm has converged [15], and especially for Gaussian mixture models, the log-likelihood landscape is multimodal [142].



(a) initialisations (b) EM results

Figure 4.1: The effect of initialisation of model parameters on the EM algorithm: two different initial solutions (left); two EM results (right).

Several initialisation methods have been presented in [43,44]. A common way to address this problem is to run EM multiple times with a different set of randomly chosen initial parameters [15] and pick the best performing solution as the result. We call this variant *repeated EM* (REM) and it gives higher stability with respect to the log-likelihood and less dependence on the initialisation and data set [43]. However, the deficiency of REM is that it leads to a great waste of computation because it restarts at every initialisation and unimproved solutions make no contribution to the final result. An accelerated EM algorithm [58] is accomplished by deriving a region bounding the possible locations of the local optimum, followed by upper bound estimation on the maximum likelihood. As a result of the estimation, the EM algorithm can be terminated in advance to avoid useless solutions.

A more sophisticated strategy for escaping a poor initial solution is to alternate between converging the solution by EM and in-

troducing a perturbation to the solution. While EM converges to a solution pointed to by a steepest gradient, perturbation can, at least in principle, circumvent that restriction. By accepting only a solution that improves the log-likelihood, perturbation-based methods guarantee that the best solution found so far is not discarded [57].

One possible perturbation operation is to split one component and merge two other components. Ueda and Nakano proposed the *split and merge EM* (SMEM) algorithm [45]. Improved the split-and-merge operation, a variant of SMEM is introduced in [46]. Other algorithmic strategies employed to escape local maximum are: *competitive learning* [60], incremental clustering implemented in *greedy EM* (GEM) [47] and stochastic variants such as *stochastic EM* (SEM) [55] and *Monte Carlo EM* (MCEM) [49].

Input: X, M
Output: C, P, MSE
$c_j = x_i | i = random(1, N), 0 \leq j \leq M$;
while $\Delta MSE \geq \epsilon$ **do**
  $p_i \leftarrow \underset{1 \leq j \leq M}{\operatorname{argmin}} d(x_i, c_j)^2, \forall i \in [1, N]$;
  $c_j \leftarrow (\sum_{p_i=j} x_i) / (\sum_{p_i=j} 1)$;
  $MSE = \sum_{i=1}^{N} d(x_i, C_{p_i})^2 / N$;
end
return C, P, MSE ;

Algorithm 2: *K-means* algorithm

4.2 K-MEANS AND SWAP-BASED CLUSTERING

The K-means (Algorithm 2) is the most famous clustering algorithm. It aims to partition N points into M clusters with minimal cost. The mean squared error (MSE) is commonly used as the cost function:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} d(x_i, c_j)^2 \tag{4.1}$$

where the distance function d is commonly the Euclidean distance.

The initilisation of the centroids in K-means is typically done by randomly selecting M data points. The algorithm stops until convergence when the difference of MSE between two iterations becomes less than a given threshold ($\epsilon = 1.53 \times 10^{-5}$ in our study). Another option for the algorithm is to stop after a fixed number of iterations defined by user.

Input: X, M
Output: C, P, MSE
$C \leftarrow initialiseCentroids(X)$;
$P \leftarrow OptimalPartition(X, C)$;
for T *times* **do**
  $C^{new} \leftarrow RandomSwap(C)$;
  $P^{new} \leftarrow LocalRepartition(P, C^{new})$;
  $KmeansIteration(P^{new}, C^{new})$;
  if $MSE(P^{new}, C^{new}) < MSE(P, C)$ **then**
    $(P, C) \leftarrow P^{new}, C^{new}$;
  end
end
$MSE = \frac{1}{N} \sum_{i=1}^{N} \|x_i - C\|^2$;
return C, P, MSE ;

Algorithm 3: Pseudocode of *Random Swap* algorithm

The initialisation problem of K-means causes that the algorithm may get stuck at local optima. In swap-based clustering, centroids are perturbed through a certain strategy in order to get rid of local minima, and the swap is accepted if it improves the clustering quality. This trial-and-error approach is simple to implement and very effective in practice. The *random swap* algorithm (RS) is based on randomisation whereby a randomly selected centroid is swapped to another randomly selected location in the region of the data. After that, local repartitioning is performed and the clustering is fine-tuned by two K-means iterations. The pseudo code of RS is in Algorithm 3. However, since the swapping is completely random

in RS, the running time is not stabilised. As such, the *deterministic swap* (DS), using different swap criteria, is also studied in [39].

Deterministic swap operations aim at finding good swaps by a systematic analysis rather than in a trial-and-error manner. In general, a favorable clustering can be found in a few swaps only, if the algorithm would know the centroid that should be swapped and the location where it should be relocated.

Several heuristic criteria have been considered for selection of the centroids to be swapped, but simple criteria such as selecting the clusters with the smallest size or variance do not work very well in practice. Other approaches, like removing one cluster [86] or merging two existing clusters, as in agglomerative clustering [143], have also been introduced. With random and deterministic swap strategies, an analysis combining the deterministic heuristic with a random swap was conducted in [39].

4.3 PAIRWISE RANDOM SWAP CLUSTERING

In random swap, the swapping is completely random, so it needs a large number of iterations to provide a clustering of good quality. A method using deterministic swaps aims at finding good swaps through systematic analysis rather than making pure trial and error. The pairwise random swap (PRS) (see Algorithm 4) clustering employs the centroid ratio in section 3.4 to establish candidates for swapping, and no parameter for the number of iterations is needed [P5].

Given a data set X and the number of clusters M as the input, two centroid sets (C_1, C_2) with M clusters are obtained initially by K-means. Then, we calculate the pair ratio value to attain the set of incorrectly located centroids S_{id} and the similarity value $S(C_1, C_2)$ according to Eq. 3.17. We then perform the *swap* function (Algorithm 5) to get an improved solution, in which we swap the detected centroid c_{1j} and c_{2j} in C_1 and C_2 ($j \in S_{id}$) randomly and fine-tune the clustering by K-means. The algorithm stops when the similarity between two centroid sets S is 1, which indicates that the two clus-

Input: X, M
Output: C, MSE
Two initializations: I_1, I_2;
(C_1, MSE_1) = *k-means*(X, I_1, M);
(C_2, MSE_2) = *k-means* (X, I_2, M);
Calculate $S_{id} = i|PR(i) > 1$ and $S(C_1, C_2)$;
while $S \neq 1$ **do**
    $(C_1^{'}, C_2^{'}, MSE_1^{'}, MSE_2^{'})$ = *Swap*(X, M, C_1, C_2, MSE_1, MSE_2, S_{id});
    $MSE_1 = MSE_1^{'}$; $MSE_2 = MSE_2^{'}$;
    $C_1 = C_1^{'}$; $C_2 = C_2^{'}$;
    Calculate $S_{id} = \{i|PR(i) > 1\}$ and $S(C_1, C_2)$;
end
return min (MSE_1, MSE_2) and corresponding C_1 or C_2;

Algorithm 4: *Pairwise Random Swap* clustering algorithm

Input: X, m, C_1, C_2, MSE_1, MSE_2, S_{id}
Output: $C_{r1}^{'}, C_{r2}^{'}$ and $MSE_{r1}^{'}, MSE_{r2}^{'}$
$MSE_{r1}^{'}$ = MSE_1 + 1;
while $MSE_{r1}^{'} > MSE_1$ **do**
    $C_{r1} \leftarrow$ random swap S_{id} on C_1;
    $(C_{r1}^{'}, MSE_{r1}^{'})$ = *k-means*(X, C_{r1}, m);
end
$MSE_{r2}^{'}$ = MSE_2 + 1;
while $MSE_{r2}^{'} > MSE_2$ **do**
    $C_{r2} \leftarrow$ random swap S_{id} on C_2;
    $(C_{r2}^{'}, MSE_{r2}^{'})$ = *k-means*(X, C_{r2}, m);
end
return $C_{r1}^{'}, C_{r2}^{'}$ and $MSE_{r1}^{'}, MSE_{r2}^{'}$;

Algorithm 5: Function of *Swap*

terings are matched. The final solution for the PRS algorithm is the centroid set corresponding to a lower MSE value of the two clusterings. On occasion, the initial centroid sets C_1 and C_2 are completely matching but the partition is local optimal, i.e. $S(C_1, C_2) = 1$ and $S_{id} \in \emptyset$ at the beginning, in which case the PRS algorithm performs a random swap on the centroids.

The proposed algorithm is a type of deterministic swap clustering, since the selection of centroids to be swapped is chosen by the centroid ratio and the allocated position of the centroids is random. The time complexity of the removal step is $O(M^2)$ and $O(1)$ for the addition step. Although the swap heuristic is capable of moving out of a local minimum, it may take a long time to move near to a local minimum. Thus, it is profitable to use K-means for fine-tuning after the swap heuristic [26]. A note for the PRS algorithm is that K-means can be substituted by other prototype-based clustering algorithms.

4.4 EXPECTATION MAXIMISATION ALGORITHM

4.4.1 EM algorithm

The EM algorithm can be used to estimate the *maximum likelihood* (ML) parameters of many different types of parametric densities. Here, we restrict the discussion to the problem of finding the ML estimates of the Gaussian mixtures with a known number M of components. The goal is then to maximise the following log-likelihood:

$$L(\Theta) = \log p(X|\Theta) = \sum_{i=1}^{N} \log \sum_{j=1}^{M} \alpha_j \mathcal{N}(x_i|\Theta_j), \tag{4.2}$$

where $\mathcal{N}(.|.)$ is Gaussian distribution, $X = (x_1, \ldots, x_N)$ is the observed d-dimensional data set of size N), Θ is the configuration of all components and $\Theta_j = (\mu_j, \Sigma_j)$ are the mean vector and covariance matrix of the jth Gaussian, respectively. Finally, α_j is the mixture weight of the jth component. The parameters α_j must satisfy

the following constraints:

$$\sum_{j=1}^{M} \alpha_j = 1, \quad \text{and}, \quad \alpha_j \geq 0, \quad j = 1, ..., M. \tag{4.3}$$

Unfortunately, a closed-form solution of the (4.2) is not possible [40], since it contains the log of the sum. Maximization is then performed on the expectation of the complete-data log-likelihood, given posterior density of the latent variables [40]. This function is usually called the Q-function, and can be written for iteration t in a concrete form of Gaussian mixtures as:

$$Q(\Theta, \Theta^{t-1}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \tau_{ij} \left\{ \log \alpha_j + \log \mathcal{N}(x_i | \Theta_j^{t-1}) \right\}. \tag{4.4}$$

where Θ^{t-1} are parameters estimated in the previous iteration. Maximization of Eq. (4.4), in terms of Θ can be performed easily, by keeping the posterior probabilities τ_{ij} fixed. Then, given estimated parameters, the posterior probability τ_{ij} of x_i from component j, can be calculated as follows:

$$\tau_{ij} = \frac{\mathcal{N}(x_i | \Theta_j^{t-1}) \alpha_j}{\sum_{l=1}^{M} \mathcal{N}(x_i | \Theta_l^{t-1}) \alpha_l} \tag{4.5}$$

The EM algorithm is in Algorithm 6. To find an initial set of parameters in EM algorithm, one possibility is to randomly select mean vectors and set equal weights and whole data covariance matrix for all components [14]. A more common practice is to first run k-means on the dataset to get a hard partitioning. The initial mean vectors are directly the cluster centroids, partition covariance is the component covariance matrix and proportion of vectors in each partition is the component weight. Several short runs of k-means starting with random initial solutions each followed by a long run of EM is recommended in [43].

The implementation of Expectation-step and Maximisation-step for each component j, $j = 1, ..., M$ at each iteration is summarised

Input: Data Set $X = \{x_1, x_2, \ldots, x_N\}$
Output: Parameters $\Theta = \{\alpha, \mu, \Sigma\}$ and log-likelihood $L(\Theta)$
1 $[\Theta, L(\Theta)] \leftarrow$ initialisation(X);
2 **while** $|L(\Theta) - L(\Theta^{t-1})| > \epsilon$ **do**
3 | Expectation-step: calculate Q-function $Q(\Theta|\Theta^{t-1})$;
4 | Maximisation-step: find Θ that maximizes $Q(\Theta|\Theta^{t-1})$;
5 **end**
6 return $\Theta, L(\Theta)$

Algorithm 6: EM algorithm

as follows:

$$\begin{aligned}
\alpha_j^t &= \frac{1}{N}\sum_{i=1}^{N}\tau_{ij}^{t-1} \\
\mu_j^t &= \frac{1}{\alpha_j^t N}\sum_{i=1}^{N}\tau_{ij}^{t-1}x_i \\
\Sigma_j^t &= \frac{1}{\alpha_j^t N}\sum_{i=1}^{N}\tau_{ij}^{t-1}(x_i-\mu_j^t)(x_i-\mu_j^t)^T
\end{aligned} \tag{4.6}$$

The algorithm proceeds by using the newly derived parameters as a guess for the next iteration. A detail derivation of the equations refers to [144].

4.4.2 Split-and-Merge EM

One strategy to overcome sensitivity to the initialisation of the EM algorithm is to identify the parts of the solution that do not fit well to the data, and then revise the solution by making local changes. One way is to split a component into two parts and to merge two other components into one. Carrying out both of these actions at the same time keeps the number of components unchanged. The *split and merge EM* (SMEM) [45] makes a systematic search through all possibilities, after which the algorithm selects the best candidates and performs the necessary operations. After the split and

merge operations have been completed, SMEM smooths the affected components with a few partial EM iterations that change the parameters of the affected components only (see Algorithm 7). The conventional EM is then performed until convergence.

Input: Data Set $X = \{x_1, x_2, \ldots, x_N\}$
Output: Parameters $\Theta = \{\alpha, \mu, \Sigma\}$ and log-likelihood $L(\Theta)$

$[\Theta_0, L(\Theta_0)] \leftarrow \text{EM}(X)$;
while *candidates left to process* **do**
  Sort candidates $(i, j, k)_{C_{\max}}$ by JMerge and JSplit (equation 4.7);
  for $c = 1 : C_{\max}$ **do**
    $[\Theta', L(\Theta')] \leftarrow \text{partialEM}((i, j, k)_c)$;
    $[\Theta^*, L(\Theta^*)] \leftarrow \text{EM}(X, \Theta')$;
    if $(L(\Theta^*) > L(\Theta))$ **then**
      $\Theta = \Theta^*$; $L(\Theta) = L(\Theta^*)$;
    end
  end
end
return $\Theta, L(\Theta)$

Algorithm 7: SMEM algorithm

SMEM algorithm searches among the candidates composed of combinations of all components i, j and k until the likelihood value improves [45]. The candidates are sorted by the merge and split criteria. Merge criterion is based on the correlation of posterior probabilities of components i and j. The split criterion is based on the Kullback-Leibler divergence between component k and the local data density.

$$\text{JMerge}(i, j) = \frac{\tau_i(\Theta)^T \tau_j(\Theta)}{||\tau_i(\Theta)||||\tau_j(\Theta)||}$$
$$\text{JSplit}(k) = \int f_k(X, \theta_k) \log \frac{f_k(X, \theta_k)}{p_k(X, \theta_k)} dx \tag{4.7}$$

where, $\tau_i(\Theta) = (\tau_{1i}(\Theta), ..., \tau_{Ni}(\Theta))$ is an N-dimensional vector con-

sisting of the posterior probabilities for the ith component. T denotes the transpose operation and $1 < k \neq i \neq j < M$. The $f_k(X, \theta_k)$ is the local data density around the component k and the $p_k(X, \theta_k)$ is the empirical distribution of X. The merged components are combined linearly and the split component is split by adding a small offset ϵ in vector or matrix on the original parameters. Then a partial EM step is performed on the merge and split candidate.

The original acceptance rule, line 7 in Algorithm 7, used the Q-function instead of $L(\Theta)$ [45]. However, it was found in [48] that by doing so the global maximum might be accidentally rejected. In [P6], we therefore check the log-likelihood in order to accept the new solution.

4.4.3 Greedy EM and stochastic EM

The *Greedy EM* (GEM) [47] algorithm increases the number of components by one at each iteration. Selection of the component for insertion is a crucial step in the algorithm. The data is partitioned into M disjoint subsets A_i for a M-component mixture. For each subset A_i, k candidate components are generated. Two data points x_l and x_r are randomly picked from subset A_i. The subset is then partitioned into two disjoint subsets A_{ir} and A_{il} in such a way that the elements in A_{il} are closer to x_l than x_r and vice versa for A_{ir}. The mean and covariance of A_{il} and A_{ir} are used as parameters for two candidate components. It is repeated until k candidate components are obtained for subset A_i. Partial EM is performed on the $M \times k$ candidates. After that, the new component is selected among $M \times k$ candidates such that it maximizes the log-likelihood when mixed into the existing mixture. The time complexity of the greedy EM algorithm is $O(M^2N)$ or $O(kMN)$ if $M < k$, where $k = 10$ is the number of candidates in [47].

Stochastic variants of the EM algorithm have also been introduced [51]. These variants typically perform a simulation of the conditional distribution of the missing data τ_{ij}, to approximate the Q function. Partitions $P = (P_1, ..., P_N)$ of X is designed by assigning

each point x_i randomly to one of the mixture components according to the multinomial distribution with τ_{ij} in the simulation step of MCEM [49] and SEM [55]. A comparison between three stochastic variants of the EM algorithm can be found in [52], while a number of asymptotic convergence properties of the stochastic EM algorithms are presented in [59]. The time complexity of the stochastic versions is $O(kMN)$, where k is the number of simulations in one simulation step.

4.5 RANDOM SWAP EM ALGORITHM

The idea of the *random swap EM* (RSEM) [P6, P7] algorithm is to alternate between simple perturbation by random swap and convergence towards the nearest optimum by employing the EM algorithm. The random swap consists of removal and addition operations of components.

The pseudo code for RSEM is presented in Algorithm 8. The initialisation is performed as for the EM algorithm, described in Section 4.4.1. After the solution has been initialised, we perform t random swap iterations (called RS-iterations). During each iteration, a component is removed, a new one is added and the resulting solution is improved using the EM algorithm. The best solution, in terms of log-likelihood, is maintained as the starting point for the next RS iteration.

Let $L(\Theta^t)$ denote the value of the log-likelihood function obtained at iteration t of the EM algorithm. Furthermore, suppose that component r is the randomly selected component for removal and the rest of components will be kept unchanged. Then the posterior probability after swapping will be calculated as:

$$\tau_{ij}^s = \frac{\alpha_j^t \mathcal{N}(x_i|\Theta_j^t)}{\sum_{l=1,l\neq r}^{M} \alpha_l^t \mathcal{N}(x_i|\Theta_l^t)} \tag{4.8}$$

and the equations for parameter changes on the rth component are:

$$\begin{aligned}
\mu_r^s &= x_p \\
\alpha_r^s &= \alpha_r^t \quad or \quad \alpha_r^s = \sum_{l=1, l\neq r}^{M} \left(\sum_{i=1}^{N} \tau_{ij}^s \right) \alpha_l^t \\
\Sigma_r^s &= \Sigma_r^t \quad or \quad \Sigma_r^s = \sum_{k=1, k\neq r}^{M} \tau_{ij}^s \Sigma_k
\end{aligned} \tag{4.9}$$

In order to retain a valid Gaussian mixture model after the swap operation, the weights $\alpha_i, 1 \leq i \leq M$ are normalised to sum up to unity.

Input: Data Set $X = \{x_1, x_2, \ldots, x_N\}$
Output: Parameters $\Theta = \{\alpha, \mu, \Sigma\}$ and log-likelihood $L(\Theta)$

```
[Θ_0, L(Θ_0)] ← initialisation(X);
Θ = Θ_0, L(Θ) = L(Θ_0) ;
for Iteration = 1 : t do
    r = U(1, M), remove rth component: α_r = 0, μ_r = 0;
    p = U(1, N), add a new component at pth position (see
    equation 4.9);
    normalise weights α to sum to 1;
    new parameters Θ^s = {α^s, μ^s, Σ^s};
    [Θ^st, L(Θ^st)] ← EM(X, Θ^s);
    if L(Θ^st) > L(Θ) then
        Θ = Θ^st;
        L(Θ) = L(Θ^st);
    end
end
return Θ, L(Θ)
```

Algorithm 8: RSEM algorithm

After each swap, the new parameters Θ^s are set as initial solutions for EM, which fine-tunes the result. After EM has converged, with a new likelihood value $L(\Theta^{st})$, we compute

$$\Delta L = L(\Theta^{st}) - L(\Theta^t) \tag{4.10}$$

If ΔL is positive, the new parameter estimate replaces the previous best solution. Otherwise the new parameter estimate is discarded. This process is repeated until all possible swap pairs are tried out and none of them improves the solution. However, as a practical matter we restrict the total number of swaps to a user selectable number of RS iterations t. A result for RSEM on S2 is shown in Fig. 4.2.

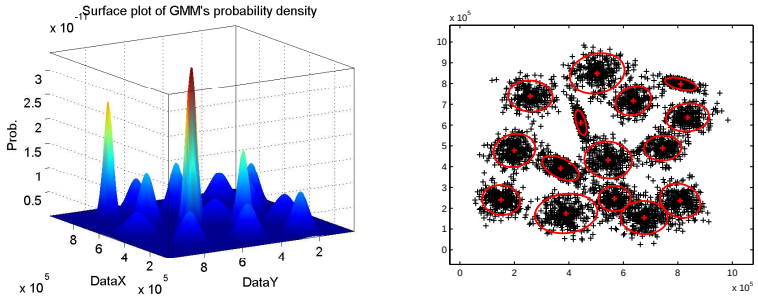


Figure 4.2: Surface plot of GMM's probability density (S2) and a clustering result from RSEM on S2.

5 Image Segmentation

Image segmentation is a key step in several image analysis methods [145]. In image analysis, the pixels contained in each region provide a good statistical sampling of data values for more reliable labeling in feature space. In image compression, the regions form a basis for a compact representation of image data. Content-based image indexing for image retrieval is another potential application of image segmentation [56].

Many methods [146, 147] have been proposed and studied in the last decades to solve the image segmentation problem. They include *active contours* [148] (e.g., *snakes* and *level sets*), *region growing* and *split-and-merge* [149], clustering-based (e.g., *mean shift*, K-means, Fuzzy C-means, EM and *normalized cuts*) and energy-based methods [150] (e.g., *variational formulation* and *Markov random field*).

In this chapter, we give a brief introduction to how clustering algorithms can be applied in image segmentation. The cluster validity measures are also studied in clustering-based image segmentation.

5.1 CLUSTERING ALGORITHMS IN IMAGE SEGMENTATION

Clustering is concerned with the partitioning of a data set into several groups such that the similarity within a group is larger than that among groups. It has a similar goal to image segmentation, where each region is homogeneous and adjacent regions are heterogeneous. Image segmentation can be converted into a clustering problem, the key issue of which is feature selection. An image has features such as texture, colour and shape, which can be selected as input data for clustering. Clustering algorithms such as K-means, Fuzzy C-means (FCM), EM and spectral clustering have been widely applied in image segmentation [151–157].

The most straightforward application of clustering algorithms is

colour quantisation, which is considered as the simplest colour image segmentation approach using cluster analysis. When the input data set is the colour space of an image, clustering points in three-dimensional space are treated as standard colour quantisation. After the clusters have been located, typically the points in each cluster are averaged to obtain the representative colour to which colours of all pixels in that cluster are mapped. However, the result is closer to real "segments" if the spatial connectivity of pixels is combined with colour quantisation.

There exist segmentation methods considering clustering methods and spatial connectivity in the neighborhood of each pixel simultaneously. Spatial information is incorporated into the membership function for clustering in Fuzzy C-means [158]. A *fast generalized fuzzy c-means* (FGFCM) clustering algorithm is proposed in citeFGFCM by incorporating local spatial and gray information together. Taking into account the inherent spatial relationships of pixels, spatial constraints for K-means is introduced in [159] to succeed in finding an accurate segmentation. A segmentation method is proposed in [160] based on a fusion of several segmentation maps from K-means clustering on an input image expressed in different color spaces. A Bayesian model is proposed in [161] for image segmentation based upon Gaussian mixture model (GMM) with spatial smoothness constraints.

Considering that the spatial connectivity is applied after feature clustering, a method called *JSEG* [156] has been designed for colour-texture image segmentation. Clustering methods are applied to obtain class maps from the original colour image, which can be viewed as a set of spatial data points located on a 2D plane (Figure. 5.1). The value of each point is the image pixel position, a 2-D vector (x, y). Spatial smoothness between pixels is enforced on class maps obtained from clustering methods.

To evaluate the difference of pixels, J value (see Eq. 5.3) is calculated based on between-class and within-class distance information over a local window (e.g., 9×9 pixels) of a class map. Suppose that X represents the pixels in the local window and X is classified into

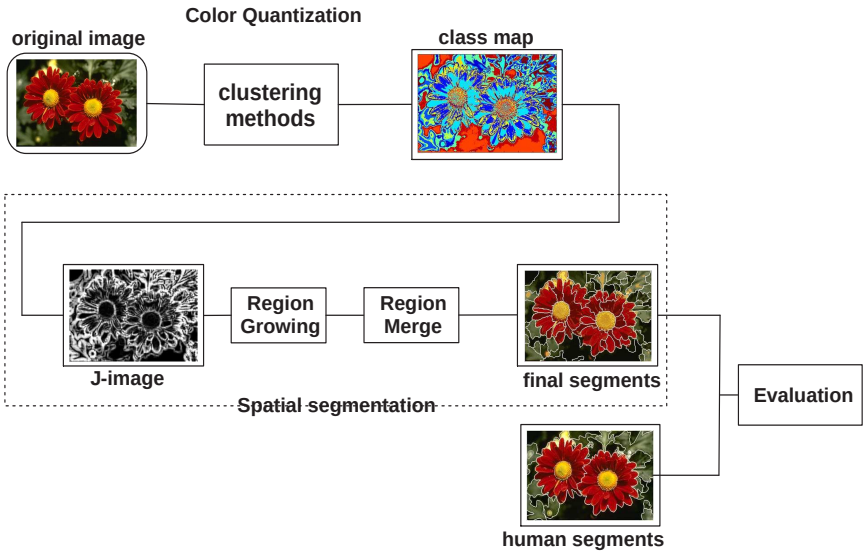


Figure 5.1: Schematic of the clustering methods applied in image segmentation algorithm JSEG.

M classes c_i, $i = 1, ..., M$. Let $\overline{X}$ stand for the mean of the pixels in the local window.

$$S_T = \sum_{x \in X} \left\| x - \overline{X} \right\|^2 \tag{5.1}$$

$$S_w = \sum_{i=1}^{M} \sum_{x \in c_i} \left\| x - c_i \right\|^2 \tag{5.2}$$

$$J = (S_T - S_w)/S_w \tag{5.3}$$

The J-image is a grey-scale image whose pixel values represent the J values calculated over local windows centered on these pixels. The image reflects the texture information. With small local windows, the J-image is useful in localizing the edges, while it is useful for detecting texture boundaries with large windows. A spatial segmentation algorithm by region growing is then performed on the J-images.

5.2 CLUSTER VALIDITY IN IMAGE SEGMENTATION

The evaluation of the quality of image segmentation [162–165] is important when comparing different segmentation methods. In most cases, segmentation acquired by dividing an image into salient regions is objective. For the evaluation of segmentation results, human interpretation is usually employed as a reference. A generic framework for evaluation of segmentation is introduced in [162]. A measure of similarity, the *normalised probabilistic Rand index*, is used for quantitative comparison between image segmentation algorithms using a hand-labelled set of ground-truth segmentations in [163]. In this index, the original Rand index is extended by combining multiple ground-truth segmentations of an image. A survey of unsupervised evaluation methods is given in [164].

Determining the number of clusters has been discussed as finding the number of segments [166–168]. One of the criteria for a good segmentation is that regions should be uniform and homogeneous with respect to certain characteristics. Internal indexes such as the sum-of-squares-based index can be applied for evaluation of unsupervised image segmentation, while model selection criteria such as *Akaike's information criterion* (AIC) [167] and *Minimum Description Length* (MDL) have also been applied for evaluation of image segmentation when model-based clustering is used.

We used the evaluation method proposed in [P4] and the BIC on EM result to determine the number of clusters, see Fig. 5.2 for results with K-means, EM and FCM. The best number of clusters in this case is three for the method employed in [P4], while it is seven or eight for the BIC on the EM algorithm. The choice between these numbers of segments is therefore subjective and expert information should therefore be involved.

External indexes can be used for evaluation of image segmentation. Usually, human segmentation is required to compare the results of different segmentation algorithms. We tested a number of EM variants in image segmentation in Fig. 5.3. It is difficult to tell the difference between the segmentations from the EM variants

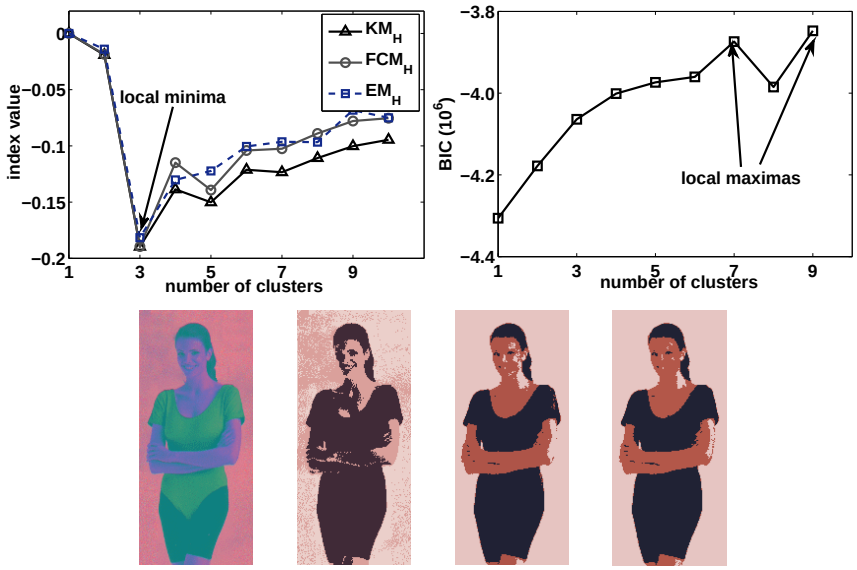


Figure 5.2: Evaluation results for the proposed method in [P4] (left) and BIC (right) for an image; The image in the YUV colour space and image segmentations to three clusters by KM, EM and FCM.

Table 5.1: Evaluation by external indexes on the segmentations by EM variants with a human-segmented result. Value 1 *indicates they are completely matched. The segmentation by RSEM is the closest to the human segmentation and REM generates the most different segmentation.*

	RI	ARI	Jac	FM
REM	0.76	0.35	0.29	0.52
SMEM	0.78	0.37	0.31	0.54
RSEM	**0.82**	**0.44**	**0.36**	**0.59**
SEM	0.79	0.38	0.32	0.54
GEM	**0.82**	**0.44**	**0.36**	0.58

and the human segmentation directly from the images. With the external indexes such as the Rand Index (RI), adjusted Rand Index (ARI), Jaccard coefficient (Jac) and Fowlkes and Mallows index (FM), it is more straightforward to see the difference (see Table 5.1). The higher the index values are, the more agreement the two seg-

mentations share. Thus, the segmentations from RSEM and GEM are closer to the human segmentation.

(a) human segmented image

(b) RSEM

(c) REM

(d) SMEM

(e) SEM

(f) GEM

Figure 5.3: Image segmentation results of different EM variants on image flower. For the pistil area, SEM and SMEM have over segmentation. REM has less segments on the non-flower area than the others. RSEM and GEM get closer segments as human segmented result.

6 Summary of Contributions

In this chapter, we summarise the contribution of the original publications [P1–P7]. The work can be separated into two research topics: cluster validity [P1–P4] and the clustering algorithm [P5–P7].

[P1]: An angle-based knee point detection method for Bayesian Information Criterion (BIC) is proposed. The BIC was originally used in model-based clustering with a Guassian mixture model. The first decisive local maximum value is considered to give the number of clusters, although the first decisive maxima is subjective. We reformulate the BIC for determining the number of clusters in partition-based clustering. The angle-based method includes information on the angles of BIC curve at the local maxima, which gives more reliable results than just considering the first maxima on the number of clusters. It provides better results than the original BIC for six out of 17 data sets while keeping the same result for eight out of 17 in terms of the percentage of the correctly determined number of clusters (see Table 7.1, 7.2 and 7.3).

[P2]: This work introduces another knee point detection method for BIC, which takes advantage of the information on the original BIC and the number of clusters. It provides better results than the original BIC for seven out of 17 data sets and maintains the same result for 10 data sets (see Table 7.1, 7.2 and 7.3). Knee point detection methods on the BIC can provide a useful guide for using the BIC in determining the number of clusters. The methods serve also as a reference to other validity indexes.

[P3]: A new sum-of-squares based validity index is proposed in this publication. The proposed WB-index takes the minimum value of the index as the determined number of clusters. The proposed index avoids the problem of knee point detection, which exists in some of the sum-of-squares based indexes. It detects the correct number of clusters for eight out of 17 data sets and provides the most correctly determined number of clusters among sum-of-

squares indexes in Table 7.1, 7.2 and 7.3. Besides, we also provide a variability and certainty analysis on the index, where quantile range and resampling method are used. In our test, the WB-index was 100% stable under RS, as shown in Fig. 7.2.

[P4]: In this article, we propose a method for extending external validity indexes for determining the number of clusters by employing a resampling method, when ground-truth information is not available. The method is applicable to both soft and hard clusterings and was verified through 10 correctly determined numbers of clusters in 17 real and artificial data sets in Table 7.1, 7.2 and 7.3. The proposed method is not affected significantly by the choice of the clustering algorithm and the structure of data sets.

[P5]: There is scant research on cluster-level validity indexes, so we propose a new index called the centroid ratio, which can be used to detect unstably or incorrectly located centroids. The time complexity of the index is $O(M^2)$, which is less than that of other known validity indexes. A pairwise random swap clustering approach employing the centroid ratio is also proposed. The algorithm is compared to random swap, deterministic random swap, repeated K-means and K-means++ and the new approach is the most efficient method in these comparisons. Moreover, it is not necessary to set any parameters in pairwise random swap clustering.

[P6]: The expectation maximisation (EM) algorithm is studied and an improved variant called the random swap EM (RSEM) is introduced. RSEM is based on the random swap strategy (addition and removal) of components to overcome the initialisation problem of EM. Random swaps of components are repeatedly performed, which can break the stuck configuration of parameters. The proposed method was in our tests 9–63% faster compared to the repeated EM, and 20–83% faster than the split and merge EM.

[P7]: In this publication, the RSEM in [P6] is used for the parameter estimation of GMMs. When tested with synthetic data, the parameters estimated by RSEM were closer to those from the EM and SMEM. The results were also verified in colour image segmentation.

7 Summary of Results

The data sets[1] in the experiment includes shaped, Gaussian-like and real data. Internal indexes, pairwise random swap clustering and RSEM, and other EM variants are implemented in *C* and *C++*, the knee point detection methods on the BIC, the method in [P4] and image segmentation *JSEG* are implemented using MATLAB. The program codes can be found in supplementary materials[2].

The validity indexes are tested with K-means and RS with repetitions. The results are studied for the performance of the indexes with different clustering algorithms using both artificial and real data, and $M_{min} = 2$ and $M_{max} = \sqrt{N}$. The values of 15 indexes are computed for all clusters for values of M in $[M_{min}, M_{max}]$. The determined number of clusters corresponds to the minimum (Krzanowski-Lai, Xu, Wb, DBI, Xie-Beni, ABIC and External) or maximum value (Calinski-Harabsz, Dunn, SC, SCI and DiffBIC) of the indexes. For some of the indexes (Ball&Hall, Hartigan and BIC), the minimum or maximum value of the second successive difference is used as a knee point detection method.

We plot the performance of validity indexes on DBI, Xie-Beni and the WB-index with K-means and RS. As shown in Fig. 7.1 and Fig. 7.2, the validity indexes with K-means rarely achieve the correct number of clusters. However, there are clear minima for indexes with RS. Furthermore, the indexes have higher variance with K-means than RS, so it is necessary to choose a stable algorithm in cluster validity. Indexes using the min or max function such as DBI and Xie-Beni have high variance among 100 repetitions. On the other hand, sum-of-squares indexes such as the WB-index are more stable.

The numbers in the tables (Table. 7.1, 7.2 and 7.3) show the percentage of the correctly determined number of clusters when using

[1] `http://cs.joensuu.fi/sipu/datasets/`
[2] `http://cs.joensuu.fi/sipu/soft/`

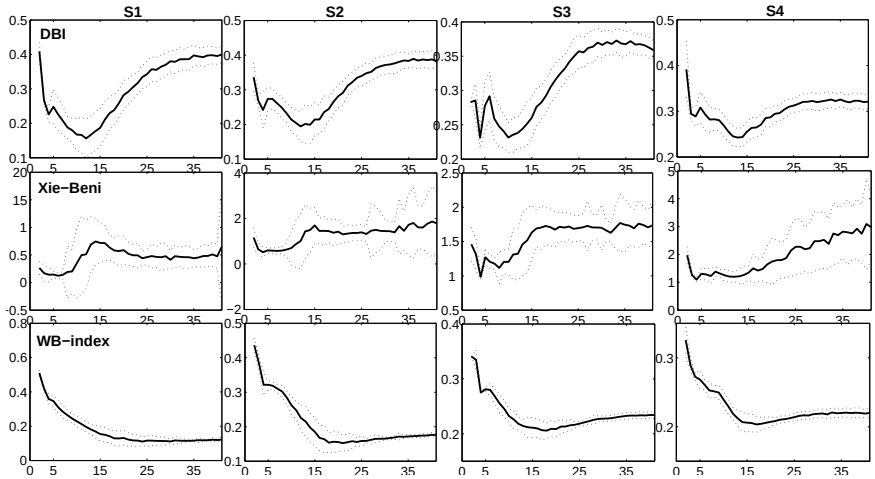


Figure 7.1: Validity indexes DBI, Xie-Beni and the WB-index when K-means algorithm has been repeated 100 times on data sets S1–S4. The solid line is the mean value and the dashed lines show the minimum and maximum values.

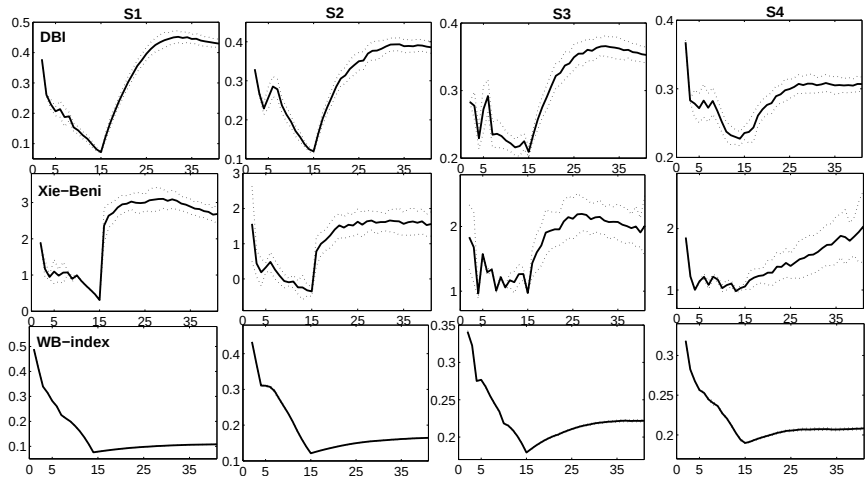


Figure 7.2: Validity indexes including DBI, Xie-Beni and the WB-index when RS algorithm has been repeated 100 times on data sets S1–S4. The solid line is the mean value and the dashed lines show the minimum and maximum values.

different validity indexes with RS. The percentages are obtained from 100 repetitions, except for birch1 (one repetition) and External (one for all data) [P4].

Table 7.1: Percentage of the correctly determined number of clusters for different cluster validity indexes with RS on shaped data.

Index \ Data	*Touching*	*pathbased*	*Compound*	*Aggregation*
M^*	**2**	**3**	**6**	**7**
M_{max}	8	17	19	28
Ball& Hall	0	0	3	0
Calinski-Harabsz	50	0	0	0
Hartigan	0	100	0	0
Krzanowski-Lai	50	0	0	0
Xu-index	0	0	0	0
WB-index [P3]	0	0	0	0
Dunn	6	0	0	1
DBI	99	100	0	0
SC	100	100	0	0
SCI	0	100	0	0
Xie-Beni	36	66	0	0
BIC	0	100	0	0
ABIC [P1]	0	20	11	0
DiffBIC [P2]	0	100	0	0
External [P4]	1	1	0	0

For unbalanced and shaped data (e.g., Aggregation), as well as for data with densities (e.g., Compound), almost all of the indexes fail (see in Table 7.1. It is interesting to view the performance of the indexes for these data sets through the lens of density-based clustering or spectral clustering instead of RS, which can be stud-

ied in future. DBI and SC work very well for both Touching and pathbased functions. As seen in Table. 7.1, the number of clusters is correctly determined in all cases by SC and DBI for data pathbased.

Table 7.2: Percentage of the correctly determined number of clusters for different cluster validity indexes with RS on Gaussian-like data.

Index \ Data	*R15*	*D31*	*S1*	*S2*	*S3*	*S4*	*birch1*
M^*	**15**	**31**	**15**	**15**	**15**	**15**	**100**
M_{max}	24	56	70	70	70	70	316
Ball& Hall	66	0	100	0	0	0	0
Calinski-Harabsz	52	44	100	100	0	1	0
Hartigan	41	2	100	18	0	0	0
Krzanowski-Lai	67	5	14	0	0	0	0
Xu-index	80	60	100	100	100	94	0
WB-index [P3]	80	60	100	100	100	96	0
Dunn	0	21	77	74	1	5	0
DBI	53	45	98	71	24	6	0
SC	85	61	100	100	100	90	0
SCI	15	61	100	100	93	0	0
Xie-Beni	62	59	100	87	37	1	0
BIC	0	6	100	1	0	0	0
ABIC [P1]	13	51	96	92	73	5	0
DiffBIC [P2]	15	59	100	100	100	9	0
External [P4]	1	1	0	1	1	1	0

The performance of indexes on Gaussian-like data is much better than that on shaped data (see Table 7.2). For large data sets such as birch1, most of the validity indexes produce cluster numbers close to 100; however, none of them gives exactly 100 as the number of clusters. Among the sum-of-squares indexes, Xu and the WB-index have similar performances. For highly overlapped

data such as S4, the Xu and WB indexes and SC have good performance in comparison to other indexes. Since ABIC and DiffBIC are improved versions of the original BIC, they outperform the original BIC in general.

For real data, the indexes give more diverse results; their performance depends strongly on the data set. For instance, the WB-index is the only index working for Iris, but it does not work for wine, control, image and yeast see Table 7.3. DiffBIC outperformed the original BIC and ABIC in these tests.

Table 7.3: Percentage of the correctly determined number of clusters for different cluster validity indexes with RS on real data.

Index \ Data	*Iris*	*Wine*	*Control*	*Image*	*Wdbc*	*Yeast*
M^*	**3**	**3**	**6**	**7**	**2**	**10**
M_{max}	12	13	24	48	23	38
Ball& Hall	0	0	10	0	0	0
Calinski-Harabsz	0	0	0	0	100	0
Hartigan	0	100	0	0	0	0
Krzanowski-Lai	0	42	0	0	100	0
Xu-index	0	0	0	0	0	0
WB-index [P3]	100	0	0	0	100	0
Dunn	0	0	0	0	0	0
DBI	0	0	0	0	100	0
SC	0	0	0	0	100	0
SCI	0	0	0	0	98	0
Xie-Beni	0	0	100	0	100	0
BIC	0	0	0	0	100	0
ABIC [P1]	0	0	0	0	0	0
DiffBIC [P2]	0	100	100	0	100	0
External [P4]	1	1	0	0	1	0

The GMMs obtained from EM variants on data S2 and R15 are

shown in Fig. 7.3 and Fig. 7.4. The median and best GMMs of 20 repetitions in terms of log-likelihood value are compared among the EM variants.

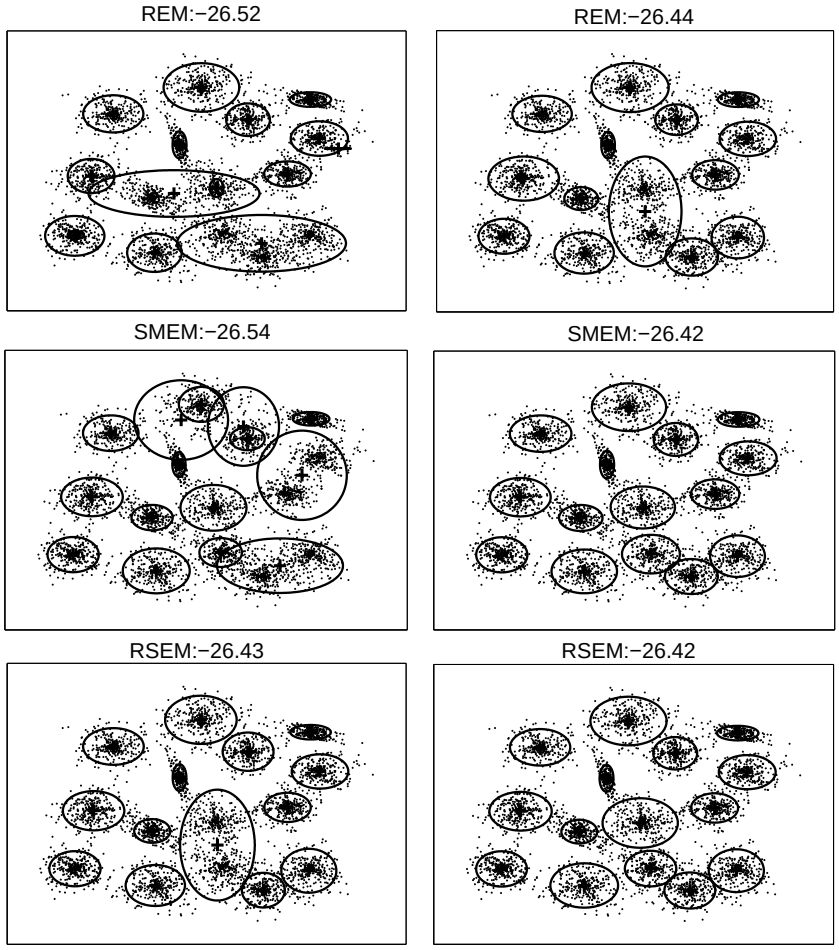


Figure 7.3: GMMs on data S2 estimated by EM variants of REM, SMEM and RSEM. The GMMs with median log-likelihood value from 20 repetitions are in the first column and the GMMs with the best log-likelihood value are in the second column.

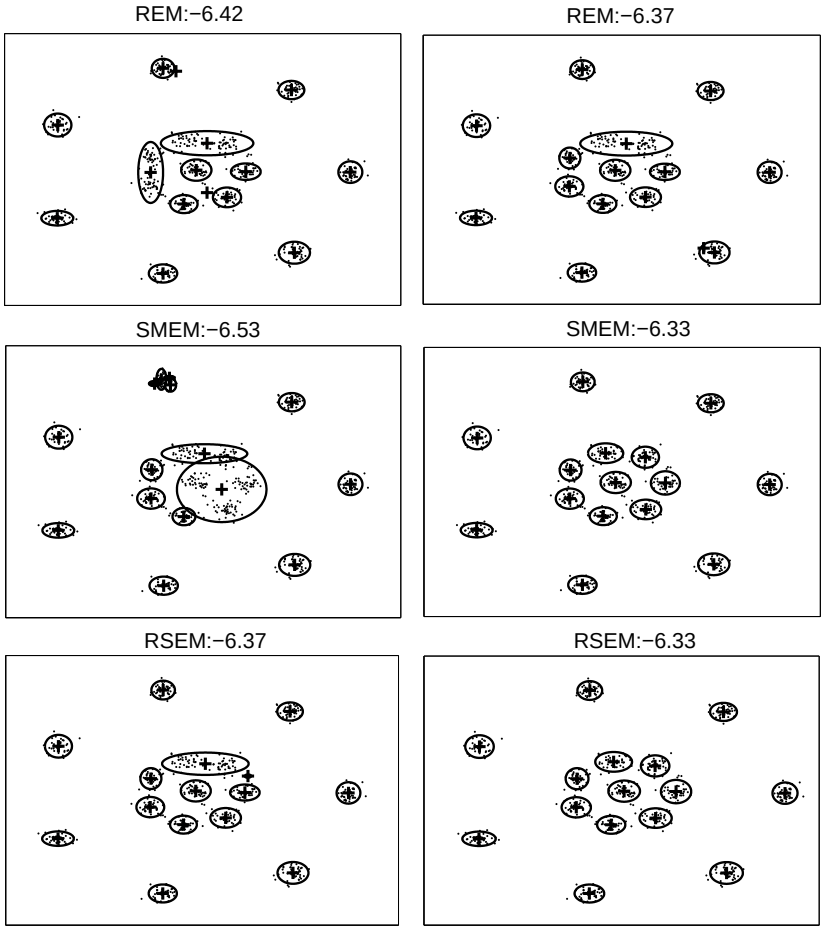


Figure 7.4: GMMs on data R15 estimated by EM variants of REM, SMEM and RSEM. The GMMs with median log-likelihood value from 20 repetitions are in the first column and the GMMs with the best log-likelihood value are in the second column.

8 Conclusions

In this thesis, we have studied cluster validity measures for evaluating the quality of clustering and determining the number of clusters. Clustering algorithms, especially swap-based clustering and the EM algorithm, have been studied in depth. The initialisation problem of EM for GMM estimation is focused.

The cluster validity is an important issue in cluster analysis, as evaluating different clustering algorithms helps the user to gain a better understanding on the properties and efficiency on different algorithms. Meanwhile, determining the number of clusters cannot be avoided in cluster methods. Furthermore, according to the study on existing internal and external validity indexes, there is no perfect, generic index suitable for every type of data, so finding the best index among the existing ones, and proposing a general validity index, is our goal.

In our practical tests, sum-of-squares-based indexes worked well for Gaussian-type data, although most of them were not capable of providing a global minimum or maximum point for the correct number of clusters. As such, two knee point detection methods were designed. The methods proposed for the BIC can be generalized to other validity indexes as well, and as a sum-of-squares based index, the WB-index used the minimum value as the optimal number of clusters. Consequently, it showed the best performance among sum-of-squares based indexes in our study. The extension of external indexes for determining the number of clusters by using the resampling method is less sensitive to the shape of data and it works with different dataset types. As a drawback, the method exhibits high time complexity, and so a more efficient design is needed. We found the results of the indexes are more stable for RS than for K-means, so this indicates that the performance of validity indexes is affected by the choice of clustering algorithms. Also, the sum-of-squares indexes (e.g., WB-index) are more stable than

indexes employing min-max functions (e.g., DBI). Finally, a validity index based on centroids only is rarely considered. The centroid ratio for two sets of centroids was proposed to evaluate the clustering globally, but its application remains a topic of future work. Based on the results of the study, a more general validity index, which relies less on the kind of input data and clustering algorithms, is highly preferred.

Cluster validity indexes were applied in image segmentation to determine the number of segments and to evaluate segmentation results. Other applications, for example validity indexes in short text clustering, can be considered in the future work, too.

Cluster validity depends strongly on clustering algorithms. For this reason we studied clustering algorithms in this thesis. Our main focus was on partition-based and model-based clustering algorithms. Pairwise random swap clustering is a swap-based algorithm, which employs the centroid ratio for selecting swapping candidates. The algorithm required 26% to 96% less processing time than the random swap, deterministic random swap, repeated K-means and K-means++ algorithms.

The EM algorithm is an iterative method for parameter estimation of Gaussian mixture models (GMM). A random swap EM algorithm (RSEM) was proposed in order to dispose of the tendency of the standard EM algorithm to get stuck in local maxima. Comparing the RSEM to the repeated EM, which is the conventional way to solve the same the problem, RSEM reaches higher or comparable levels of log-likelihood and is 9-63% faster, which was proved by a bound derived from our formulas. RSEM is also easier to implement and more efficient (20-83% faster) than SMEM. Determining the number of components for GMMs in EM variants is a problem and it should be studied further.

Bibliography

[1] I. Kärkkäinen, *Methods for fast and reliable clustering (PhD. thesis)* (University of Joensuu, 2006).

[2] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters* **31(8),** 651–666 (2010).

[3] P. Scheunders, "A comparison of clustering algorithms applied to color image quantization," *Pattern Recognition Letters* **18,** 1379–1384 (1997).

[4] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Trans. on Image Processing* **19,** 2761–2773 (2010).

[5] T. Kinnunen, I. Sidoroff, M. Tuononen, and P. Fränti, "Comparison of clustering methods: a case study of text-independent speaker modeling," *Pattern Recognition Letters* **32,** 1604–1617 (2011).

[6] S. Fodeh, W. Punch, and P. Tan, "Combining statistics and semantics via ensemble model for document clustering," *Proc. of the 2009 ACM symposium on Applied Computing* 1446-1450 (2009).

[7] H. Zheng, B. Kang, and H. Kim, "Exploiting noun phrases and semantic relationships for text document clustering," *Information Sciences* **179,** 2249–2262 (2009).

[8] A. Jaffe, M. Naaman, T. Tassa, and M. Davis, "Generating summaries and visualization for large collections of geo-referenced photographs," *Proc. of the 8th ACM international workshop on Multimedia information retrieval* 89-98 (2006).

[9] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, "Exploiting semantic annotations for clustering geographic areas and

users in location-based social networks," *Fifth International AAAI Conference on Weblogs and Social Media* 32-35 (2011).

[10] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing* **17,** 395–416 (2007).

[11] R. Xu and D. WunschII, "Survey of clustering algorithms," *IEEE Trans. on Neural Networks* **16,** 645–678 (2005).

[12] R. Vidal, "A tutorial on subspace clustering," *IEEE Signal Processing Magazine* (2010).

[13] L. Rokach, "A survey of clustering algorithms," *Data Mining and Knowledge Discovery Handbook* 269-298 (2010).

[14] M. A. Figueiredo and A. K. Jain, "Unsupervised learning of Finite Mixture Models," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **24,** 381–396 (2002).

[15] G. J. McLachlan and D. Peel, *Finite Mixture Models* (John Wiley &Sons, 2000).

[16] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys (CSUR)* **31,** 264–323 (1999).

[17] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *Journal of Machine Learning Research* **6,** 1705–1749 (2005).

[18] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems* **17,** 107–145 (2001).

[19] P. Berkhin, "Survey of clustering data mining techniques," (2002).

[20] G. Sudipto, R. Rajeev, and S. Kyuseok, "CURE: An efficient clustering algorithm for large databases," *Information Systems* **26(1),** 35–58 (2001).

[21] G. Karypis, E. Han, and V. Kumar, "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling," *IEEE Computer* **32(8),** 68–75 (1999).

[22] D. Boley, "Principal direction divisive partitioning," *Data Mining and Knowledge Discovery* **2,** 325–344 (1998).

[23] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," *15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'09)* 877-886 (2009).

[24] L. Kuncheva and D. Vetrov, "Evaluation of stability of k-means cluster ensembles with respect to random initialization," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **28,** 1798–1808 (2006).

[25] P. Fränti and J. Kivijärvi, "Randomised local search algorithm for the clustering problem," *Pattern Analysis and Applications* **3,** 358–369 (2000).

[26] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "A local search approximation algorithm for k-means clustering," *Computational Geometry* **28,** 89–112 (2004).

[27] A. Likas, N. Vlassis, and J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition* **36,** 451–461 (2003).

[28] A. Jain, M. Murty, and P. Flynn, "Data clustering: a review," *ACM Computing Surveys* **31,** 264–323 (1999).

[29] G. Babu and M. Murty, "Simulated annealing for selecting optimal initial seeds in the K-means algorithm," *Journal of Pure and Applied Mathematics* **25,** 85–94 (1994).

[30] K. Krishna and M. Murty, "Genetic K-means algorithm," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics* **29,** 433–439 (1999).

[31] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silberman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **24,** 881–892 (2002).

[32] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," *ACM-SIAM Symp. on Discrete Algorithms (SODA'07)* 1027-1035 (2007).

[33] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for k-means clustering," *Pattern Recognition Letters* **25,** 1293–1302 (2004).

[34] P. Bradley and U. Fayyad, "Cluster center initialization algorithm for k-means clustering," *Proc. of the 15th Int. Conf. on Machine Learning (ICML'98)* 91-99 (1998).

[35] D. Pelleg and A. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," *Proc. of the 17th Int. Conf. on Machine Learning* 727-734 (2000).

[36] J. Wu, J. Chen, H. Xiong, and M. Xie, "External validation measures for K-means clustering: A data distribution perspective," *Expert Systems with Applications* **36,** 6050–6061 (2009).

[37] S. Ray and R. Turi, "Determination of number of clusters in k-means clustering and application in color image segmentation," *Proc. of the 4th Int. Conf. on Advances in Pattern Recognition and Digital Techniques* 137-143 (1995).

[38] P. Fränti, O. Virmajoki, and V. Hautamäki, "Probabilistic clustering by random swap algorithm," *19th Int. Conf. on Pattern Recognition (ICPR'08)* **55,** 287–314 (2008).

[39] P. Fränti and O. Virmajoki, "On the efficiency of swap-based clustering," *Int. Conf. on Adaptive and Natural Computing Algorithms (ICANNGA'09)* 303-312 (2009).

[40] C. Bishop, *Pattern Recognition and Machine Learning* (Springer Verlag, 2006).

[41] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of Royal Statistical Society B* **39,** 1–38 (1977).

[42] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Section 16.1. Gaussian Mixture Models and k-Means Clustering, Numerical Recipes: The Art of Scientific Computing (3rd ed.)* (New York: Cambridge University Press. ISBN 978-0-521-88068-8, 2007).

[43] G. Biernacki, C. Celeux, and G. Govaert, "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models," *Computational Statistics and Data Analysis* **41,** 561–575 (2003).

[44] D. Karlis and E. Xekalaki, "Choosing initial values for the EM algorithm for finite mixtures," *Computational Statistics and Data Analysis* **41,** 577–590 (2003).

[45] N. Udea, R. Nakano, Z. Gharhamani, and G. Hinton, "SMEM algorithm for mixture models," *Neural Computation* **12,** 2109–2128 (2000).

[46] Z. Zhang, C. Chen, J. Sun, and K. Chan, "EM algorithms for Gaussian mixtures with split-and-merge operation," *Pattern Recognition* **36,** 1973–1983 (2003).

[47] J. Verbeek, N. Vlassis, and B. Krose, "Efficient greedy learning of Gaussian mixture models," *Neural Computation* **15,** 469–485 (2003).

[48] A. Minagawa, N. Tagawa, and T. Tanaka, "SMEM algorithm is not fully compatible with Maximum-Likelihood Framework," *Neural Computation* **14,** 1261–1266 (2002).

[49] G. Wei and M. Tanner, "A Monte Carlo Implementation of the EM algorithm and the poor man's data augmentation

algorithms," *Journal of the American Statistical Association* **85,** 699–704 (1990).

[50] S. Richardson and P. J. Green, "On Bayesian analysis of mixtures with an unknown number of components," *Journal of the Royal Statistical Society* **59,** 731–792 (2002).

[51] G. Celeux, D. Chauveau, and J. Diebolt, "On stochastic versions of the EM algorithm," *Techniqual report no.2514* (1995).

[52] G. Celeux, D. Chauveau, and J. Diebolt, "Stochastic versions of the EM algorithm: An experimental study in the mixture case," *Journal of statistical computation and simulation* **55,** 287–314 (1996).

[53] N. Ueda and R. Nakano, "Deterministic annealing EM algorithm," *Neural Networks* **11,** 271–282 (1998).

[54] C. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics* **11,** 95–103 (1983).

[55] G. Celeux and J. Diebolt, "The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem," *Computational Statistics Quaterly* **2,** 73–82 (1985).

[56] S. B. Chad, C. Carson, H. Greenspan, and J. Malik, "Color- and Texture-Based image segmentation using EM and its application to content-based image retrieval," *Proc. of the Sixth Int. Conf. on Computer Vision* 675-682 (1998).

[57] S. Huda, J. Yearwood, and R. Togneri, "A stochastic version of Expectation Maximization algorithm for better estimation of Hidden Markov Model," *Pattern Recognition Letters* **30,** 1301–1309 (2009).

[58] Z. Zhang and B. T. D. A. K. Tung, "Estimating local optimums in EM algorithm over Gaussian Mixture Model," *Proc. of the 25th Int. Conf. on Machine Learning* 1240-1247 (2008).

[59] S. F. Nielsen, "The stochastic EM algorithm: estimation and asymptotic results," *Bernoulli* **6,** 457–489 (2000).

[60] B. Zhang, C. Zhang, and X. Yi, "Competitive EM algorithm for finite mixture models," *Pattern Recognition* **37,** 131–144 (2004).

[61] I. Kärkkäinen and P. Fränti, "Gradual model generator for single-pass clustering," *Pattern Recognition* **40,** 784–795 (2007).

[62] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," *Learning in graphical models* 355-368 (1999).

[63] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proc. of the Second Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)* 226-231 (1996).

[64] J. Sander, M. Ester, H. Kriegel, and X. Xu, "Density-Based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Mining and Knowledge Discovery* 169-194 (1998).

[65] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Int. Conf. on Management of data* 49-60 (1999).

[66] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **26,** 214–225 (2004).

[67] W. Wang, D. Yang, and R. Muntz, "STING: a statistical information grid approach to spatial data mining," *Proc. of the 1997 Int. Conf. on Very Large Data Bases (VLDB'97)* 186-195 (1997).

[68] N. Park and W. Lee, "Statistical grid-based clustering over data streams," *ACM SIGMOD Record* **33,** 32–37 (2004).

[69] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: a multi-resolution clustering approach for very large spatial databases," *Proc. of the 1998 Int. Conf. on Very Large Data Bases (VLDB'98)* 428-439 (1998).

[70] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *Proc. of the SIGMOD Conference* 94-105 (1998).

[71] A. Strehl and J. Ghosh, "Cluster Ensembles - A knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research* **3,** 583–617 (2002).

[72] R. Caruana, M. Elhawary, N. Nguyen, and C. Smith, "Meta clustering," *Sixth Int. Conf. on Data Mining (ICDM'06)* 107-118 (2006).

[73] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. on Knowledge Discovery from Data* **1** (2007).

[74] N. Nguyen and R. Caruana, "Consensus clusterings," *Seventh IEEE Int. Conf. on Data Mining (ICDM'07)* 607-612 (2007).

[75] R. Avogadri and G. Valentini, "Ensemble clustering with a fuzzy approach," *Studies in Computational Intelligence* **126,** 49–69 (2008).

[76] A. Topchy, A. K. Jain, and W. Punch, "A mixture model for clustering ensembles," *In Proc. of the SIAM Int. Conf. on Data Mining* 379-390 (2004).

[77] X. Fern and C. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," *Proc. of the 21st Int. Conf. on Machine Learning* (2004).

[78] G. Milligan and M. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika* **50,** 159–179 (1985).

[79] E. Dimitriadou, S. Dolnicar, and A. Weingassel, "An examination of indexes for determining the number of clusters in binary data sets," *Psychometrika* **67(1),** 137–160 (2002).

[80] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **24,** 1650–1654 (2002).

[81] N. Pal and J. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. on Fuzzy Systems* **3,** 370–379 (1995).

[82] Y. Zhang, W. Wang, X. Zhang, and Y. Li, "A cluster validity index for fuzzy clustering," *Information Sciences* **178,** 1205–1218 (2008).

[83] K. Wang, B. Wang, and L. Peng, "CVAP: Validation for cluster analyses," *Data Science Journal* **8,** 88–93 (2009).

[84] E. Rendon, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus external cluster validation indexes," *International Journal of Computers and Communications* **5,** 27–34 (2011).

[85] P. Fränti, "Clustering data sets," `http://cs.joensuu.fi/sipu/datasets/` (2009).

[86] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition* **39,** 761–765 (2006).

[87] J. Handl, J. Knowles, and D. Kell, "Computational cluster validation in post-genomic data analysis," *Bioinformatics* **21,** 3201–3212 (2005).

[88] C. Lunnerborg, "Data analysis by resampling: concepts and applications," *Duxbury Press, ISBN 0-534-22110-6, Pacific Grove, CA, USA* (2000).

[89] A. Ben-Hur, A. Elisseeff, and I. Guyon, "A stability based method for discovering structure in clustered data," *Proc. of the Pacific Symposium on Biocomputing* 6-17 (2002).

[90] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of Royal Statistical Society. B* **63,** 411–423 (2001).

[91] R. Peck, L. Fisher, and J. Ness, "Approximate confidence intervals for the number of clusters," *Journal of the American Statistical Association* **84,** 184–191 (1989).

[92] R. Tibshirani and G. Walther, "Cluster validation by prediction strength," *Journal of Computational and Graphical Statistics* **14,** 511–528 (2005).

[93] S. Dudoit and J. Fridlyand, "A prediction-based resampling method for estimating the number of clusters in a dataset," *Genome Biology* **3** (2002).

[94] M. Falasconi, A. Gutierrez, M. Pardo, G. Sberveglieri, and S. Marco, "A stability based validity method for fuzzy clustering," *Pattern Recognition* **43,** 1292–1305 (2010).

[95] C. Borgelt and R. Kruse, "Finding the number of fuzzy clusters by resampling," *IEEE Int. Conf. on Fuzzy Systems* 48-54 (2006).

[96] K. Mardia, J. Kent, and J. Bibby, *Multivariate analysis* (Academic Press, 1979).

[97] H. Siirtola, *Interactive visualization of multidimensional data (PhD. thesis)* (Tampereen Yliopisto, 2007).

[98] A. Asuncion and D. Newman, "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml/ (2007).

[99] L. Kaufman and P. Rousseeuw, *Finding groups in data: an introduction to cluster analysis* (New York, John Wiley & Sons, 1990).

[100] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE* **78,** 1464–1480 (1990).

[101] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science* **315,** 972–976 (2007).

[102] G. Brock, V. Pihur, S. Datta, and S. Datta, "clValid: An R package for cluster validation," *Journal of Statistical Software* **25,** 1–28 (2008).

[103] J. Dunn, "Well separated clusters and optimal fuzzy partitions," *Journal of Cybernetica* **4,** 95–104 (1974).

[104] D. Davies and D. Bouldin, "Cluster separation measure," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **1,** 95–104 (1979).

[105] X. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **13,** 841–847 (1991).

[106] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communication in statistics* **3,** 1–27 (1974).

[107] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment: Finding the optimal partitioning of a data set," *Proc. of the 2001 IEEE Int. Conf. on Data Mining (ICDM'01)* 187-194 (2001).

[108] W. Wang and Y. Zhang, "On fuzzy cluster validity indices," *Fuzzy Sets and Systems* **158,** 2095–2117 (2007).

[109] G. Ball and L. Hubert, "ISODATA, A novel method of data analysis and pattern classification," *(Tech. Rep. NTIS No. AD 699616). Menlo Park, CA: Standford Research Institute* (1965).

[110] J. Hartigan, "Clustering algorithms," *New York, NY: Wiley* (1975).

[111] L. Xu, "Bayesian Ying-Yang machine, clustering and number of clusters," *Pattern Recognition Letters* **18,** 1167–1178 (1997).

[112] C. Frayley and A. Raftery, "How many clusters? Which clustering method? answers via model-based cluster analysis," (1998), Technical Report no. 329, Department of Statistics, University of Washington.

[113] M. Zoubi and M. Rawi, "An efficient approach for computing silhouette coefficients," *Journal of Computer Science* **4,** 252–255 (2008).

[114] S. Still and W. Bialek, "How many clusters? An information theoretic perspective," *Neural Computation* **16,** 2483–2506 (2004).

[115] A. Dasgupta and A. Raftery, "Detecting features in spatial point process with clutter via model-based clustering," *Journal of the American Statistical Association* **93,** 294–302 (1998).

[116] C. E. Rasmussen, "The infinite Gaussian Mixture Model," *in Advances in Neural Information Processing Systems* **12,** 554–560 (2000).

[117] W. Krzanowski and Y. Lai, "A criterion for determining the number of groups in a data set using sum-of-squares clustering," *Biometrics* **44,** 23–34 (1985).

[118] M. H. Y. Batistakis and M. Vazirgiannis, "Clustering validity checking methods: Part II," *ACM SIGMOD Record* **31,** 19–27 (2002).

[119] M. H. Y. Batistakis and M. Vazirgiannis, "Cluster validity methods: Part I," *ACM SIGMOD Record* **31,** 40–45 (2002).

[120] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," *Proc. of the 16th IEEE Int. Conf. on Tools with Artificial Intelligence* 576-584 (2004).

[121] B. E. Dom, "An information-theoretic external cluster-validity measure," *Research Report RJ 10219, IBM* (2001).

[122] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clustering comparison: Is a correction for chance necessary?," *ICML'09* 1073-1080 (2009).

[123] M. Law and A. Jain, "Cluster validity by bootstrapping partitions," *Technical Report MSU-CSE-03-5, Dept. of Computer Science and Engineering, MSU, Michigan, USA* (2003).

[124] H. W. S. Zhang and Y. Shen, "Generalized adjusted rand indices for cluster ensembles," *Pattern Recognition* **45,** 2214–2226 (2012).

[125] Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," *Machine Learning* **55,** 311–331 (2004).

[126] C. Rijsbergen, "Information Retrieval (2nd Edition)," *Butterworths, London* (1979).

[127] M. Meila, "Comparing clusterings by the variation of information," *Proc. of the 16th annual conf. of computational learning theory (COLT)* 173-187 (2003).

[128] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypershpere using von mises-fisher distributions," *Journal of Machine Learning Research* **6,** 1345–1382 (2005).

[129] W. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association* **66,** 846–850 (1971).

[130] R. Sokal and P. Sneath, "Principles of numeric taxonom," *W.H. Freeman, San Francisco,* (1963).

[131] E. Fowlkes and C. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American Statistical Association* **78,** 553–569 (1983).

[132] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification* **2,** 193–218 (1985).

[133] A. Ben-Hur and I. Guyon, "Detecting stable clusters using principal component analysis," *Methods in Molecular Biology* (2003).

[134] L. Goodman and W. Kruskal, "Measures of association for cross classification," *Journal of the American Statistical Association* **49,** 732–764 (1954).

[135] R. Campello, "A fuzzy extension of the Rand index and other related indexes for clustering and classification assessment," *Pattern Recognition Letters* **28,** 833–841 (2007).

[136] C. Michele and M. Antonio, "A fuzzy extension of some classical concordance measures and an efficient algorithm for their computation," *Int'l Conf. on Knowledge-Based Intelligent Information and Engineering Systems* 755-763 (2008).

[137] T. Hasan, Y. Lei, A. Chandrasekaran, and J. Hansen, "A novel feature sub-sampling method for efficient universal background model training in speaker verification," *ICASSP'10* 4494-4497 (2010).

[138] J. Bezdek, "FCM: the fuzzy c-means clustering algorithm," *Computers & Geosciences* **10,** 191–203 (1984).

[139] C. Elkan, "Using the triangle inequality to accelerate k-means," *Proc. of the Twentieth Int. Conf. on Machine Learning (ICML'03)* (2003).

[140] S. David, D. Pal, and H. Simon, "Stability of k-Means Clustering," *Learning Theory* 20-34 (2007).

[141] A. Rakhlin and A. Caponnetto, "Stability of k-Means clustering," *Advances in Neural Information Processing Systems* **19** (2007).

[142] G. Mclachlan and T. Krishnan, *The EM algorithm and extensions* (John Wiley &Sons, 1996).

[143] H. Frigui and R. Krishnapuram, "Clustering by competitive agglomeration," *Pattern Recognition* **30,** 1109–1119 (1997).

[144] J. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Mixture Models," *Technical Report TR-97-021, ICSI* (1997).

[145] H. D. Cheng, X. H. Jiang, Y. Sun, and J. L. Wang, "Color image segmentation: Advances and prospects," *Pattern Recognition* **34,** 2259–2281 (2001).

[146] R. Szeliski, *Computer Vision: Algorithms and Applications* (Springer, Berlin, Germany, 1st edition, 2011).

[147] Y. Boykov and G. Funka-lea, "Current methods in medical image segmentation," *annual Review of Biomedical Engineering* **2,** 315–337 (2000).

[148] M. I. A. Blake, *Active Contours* (Springer-Verlag, 1998).

[149] C. Adams and L. Bischof, "Seeded region growing," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **16,** 641–647 (1994).

[150] Y. Boykov and G. Funka-lea, "Graph cuts and efficient N-D image segmentation," *International Journal of Computer Vision* **70,** 109–131 (2006).

[151] C. Carson, S. Belongie, H. Greenspan, and J.Malik, "Blobworld: Image segmentation using Expectation-Maximization and its application to image querying," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **24,** 1026–1038 (2002).

[152] W. Cai, S. Chen, and D. Zhang, "Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation," *Pattern Recognition* **40,** 825–838 (2007).

[153] M. Figueiredo, D. Cheng, and V. Murino, "Clustering under prior knowledge with application to image segmentation," *Advances in Neural Information Processing Systems 19* (2007).

[154] G. Coleman and H. Andrews, "Image segmentation by clustering," *Proc. of the IEEE* **67,** 773–785 (1979).

[155] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **22,** 1026–1038 (2000).

[156] Y. Deng and B. Manjunath, "Unsupervised segmentation of color-texture regions in images and videos," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **23,** 800–810 (2001).

[157] D. Comaniciu and P. Meer, "Mean Shift: a robust approach toward feature space analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **24,** 603–619 (2002).

[158] K. Chuang, H. Tzeng, S. Chen, J. Wu, and T. Chen, "Fuzzy c-means clustering with spatial information for image segmentation," *Computerized medical imaging and graphics* **30,** 9–15 (2006).

[159] M. Mignotte, "A de-texturing and spatially constrained K-means approach for image segmentation," *Pattern Recognition Letters* **32,** 359–367 (2011).

[160] M. Mignotte, "Segmentation by fusion of histogram-based k-means clusters in different color spaces," *IEEE Trans. on Image Processing* **17,** 780–787 (2008).

[161] C. Nikou, A. C. Likas, and N. P. Galatsanos, "A Bayesian framework for image segmentation with spatial varying mixtures," *IEEE Trans. on Image Processing* **19,** 2278–2289 (2010).

[162] J. Cardoso and L. Corte-Real, "Toward a generic evaluation of image segmentation," *IEEE Trans. on Image Processing* **14,** 1773–1782 (2005).

[163] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **29,** 929–944 (2007).

[164] H. Zhang, J. Fritts, and S. Goldman, "Image segmentation evaluation: A survey of unsupervised methods," *Computer Vision and Image Understanding* **110,** 260–280 (2008).

[165] S. Chabrier, B. Emile, H. Laurent, C. Rosenberger, and P. Marche, "Unsupervised evaluation of image segmentation application to multi-spectral images," *Int. Conf. on Pattern Recognition (ICPR'04)* **1,** 576– 579 (2004).

[166] H. L. Capitaine and C. Frelicot, "On selecting an optimal number of clusters for color image segmentation," *Int. Conf. on Pattern Recognition (ICPR'10)* 3388-3391 (2010).

[167] M. El-Melegy, E. Zanaty, W. Abd-Elhafiez, and A. Farag, "On cluster validity indexes in fuzzy and hard clustering algorithms for image segmentation," *ICIP'07* 5-8 (2007).

[168] D. Langan, J. Modestino, and J. Zhang, "Cluster validation for unsupervised stochastic model-based image segmentation," *IEEE Trans. on Image Processing* **7,** 180–195 (1998).

Paper **P1**

©2008 Springer Science + Business Media. Reprinted, with permission.

Q. Zhao, V. Hautamäki and P. Fränti,
Knee Point Detection in BIC for Detecting the Number of Clusters, in *Advanced Concepts for Intelligent Vision Systems*, pp. 664–673, Juan-les-Pins, France, 2008.

Knee Point Detection in BIC for Detecting the Number of Clusters

Qinpei Zhao, Ville Hautamaki, and Pasi Fränti

Department of Computer Science, University of Joensuu
Box 111, Fin-80101 Joensuu
Finland
{zhao,villeh,franti}@cs.joensuu.fi

Abstract. Bayesian Information Criterion (BIC) is a promising method for detecting the number of clusters. It is often used in model-based clustering in which a decisive first local maximum is detected as the number of clusters. In this paper, we re-formulate the BIC in partitioning based clustering algorithm, and propose a new knee point finding method based on it. Experimental results show that the proposed method detects the correct number of clusters more robustly and accurately than the original BIC and performs well in comparison to several other cluster validity indices.

1 Introduction

Cluster analysis is to group a collection of patterns, which is usually represented as a vector of measurements or a point in a multidimensional space, into clusters according to a clustering similarity function or a clustering validity index. The output of clustering over the same dataset could be very different if the input parameters for clustering vary. This is due to the fact that variation of clustering parameters has changed the behaviour and the execution of clustering substantially. An essential input parameter for clustering is the number of clusters that best fits a given dataset. Thus, a common question arising before clustering is how many clusters are present in a given dataset. Moreover, most clustering algorithms face several common issues in execution of clustering: if different partitions are obtained for a given dataset, then amongst the resulting partitions, which one is the most suitable or optimal one.

A number of measures have been well developed for this problem in literature [1-12]. Milligan and Cooper [1] have provided a comparison of thirty validity indices for data sets by using only hierarchical clustering algorithms. Dimitriadou et al [2] present another comparison of fifteen validity indices for binary data sets. Based on a typical definition of clusters where the points within the same cluster are close to each other while the clusters themselves are far from each other, several measures have been proposed. Calinski and Harabasz [3] proposed the F-statistic method, which takes advantage of within-cluster variance and between-cluster variance. Dunn's index [4] considered both the diameter of each cluster and the distance between clusters. As the diameter will be severely affected by noise, the Dunn's index may not perform very well as a cluster validity index. This issue has been addressed in [5].

J. Blanc-Talon et al. (Eds.): ACIVS 2008, LNCS 5259, pp. 664–673, 2008.
© Springer-Verlag Berlin Heidelberg 2008

Davies-Bouldin index [6] is another well known index which is based on the idea that inter cluster separation as well as intra cluster homogeneity and compactness should be high for a good partition.

Because different kinds of clustering algorithms often have different properties, different types of measures based on specific clustering algorithms have been proposed. For example, Xie-Beni index [7] was originally proposed to identify separation for fuzzy c-partitions. It depends on the data set, geometric distance measure, distance between cluster centroids, and more importantly on the fuzzy partition generated by any fuzzy algorithm. When dealing with model-based clustering, Banfield and Raftery used a heuristically derived approximation to twice the log Bayes factor [8] called the "AWE" to determine the number of clusters in hierarchical clustering based on the classification likelihood. When EM is used to find the maximum mixture likelihood, a more reliable approximation to AWE called *Bayesian Information Criterion* (BIC) [9] is applicable. A new K-means based algorithm incorporating model selection was proposed in [10]. This so-called X-means algorithm uses BIC to make local decisions that maximize the posterior probabilities of the model under the assumption that the models are spherical Gaussians. Because of the effectiveness of BIC in model-based clustering, we re-formulate BIC to determine the number of clusters in partitioning based clustering.

Some of the indices can be easily used to determine the number of clusters by finding the minimal or maximal value, but several of them cannot. A criterion with within-group sum-of-squares objective function trace (W) was proposed by Krzanowski [11], in which the plot of index value against number of clusters was monotonically decreasing. They considered using the successive difference of the function to find the optimum value. Yet, in the visual "number of clusters vs. criterion metric" graph, a clear knee point (or jump point) is often used to detect the number of clusters, see Fig.1. In principle, the problem of finding the knee point can be attacked by successive difference method. But the successive difference method only considers some adjacent points and local trend of the graph which may lead to incorrect results. We therefore propose to measure the knee point based on the angles of the local significant changes in the successive difference results, and demonstrate that by this method, the performance of the BIC method can be improved.

The rest of the paper is organized as follows. The problem formulation is given in Section 2.1. The BIC method in partitioning based clustering is renewed in Section 2.2 and the angle-based method is introduced in Section 2.3. The proposed method is compared to several existing methods in Section 3. The results show that the proposed knee point finding method improves the original BIC method, which takes the first local minimum as the number of clusters and outperforms most of the existing method on the data sets tested. Conclusions are drawn in Section 4.

2 Proposed Method

We proposed a knee point finding method for BIC in partitioning based clustering, which is called *angle-based method*. The next section describes the proposed method.

2.1 Preliminary

The problem of determining the number of clusters is defined as follows:

Given a fixed number of clusters $m \geq 2$, and a specific clustering algorithm, find the clustering that best fits for the data set with different parameters. The procedure of identifying the best clustering scheme involves the following parts:

- Select a proper cluster validity index.
- Repeat a clustering algorithm successively for number of clusters, *m* from a predefined minimum to a predefined maximum.
- Plot the "number of clusters vs. criterion metric" graph and select the *m* at which the partition appears to be "best" when the criterion is optimized.

Based on this procedure, one can identify the best clustering scheme. However, the problem of selecting the optimal *m* for the validity index remains. Mean square error (MSE), for example, exhibits a decrease with respect to *m* increasing. Meanwhile, some indexes show the maximum or minimum in the curve. No matter what kind of case we have, there exists the significant local change in the curve called the knee or jump point.

Locating the knee point in the validity index curve is not well-studied. A straightforward approach is to take the difference of successive index values, for example, calculating the difference between previous and current values of the index. The method like L-method [12] is proposed to find the knee point of the curve by the boundary between the pair of straight lines that most closely fit the curve. For some indexes, the maximum or minimum value will be considered as the knee point. However, if there are several local maximum (minimum) values existing, the challenge is to decide which one is the most suitable one to indicate the information of the data sets. According to our study, BIC indicates a good prospect in determining the number of clusters in partitioning based clustering. To improve the accuracy of BIC, a good knee point finding method instead of taking the first local maximum is needed.

2.2 Bayesian Information Criterion (BIC)

The *Bayesian Information Criterion* (BIC) has been successfully applied to the problem of determining the number of components in model-based clustering by Banfield and Raftery. The problems of determining the number of clusters and the clustering method are solved simultaneously.

We derive the formula of BIC based on Kass and Wasserman [13].

$$BIC = L(\theta) - \frac{1}{2} m \log n \tag{1}$$

Where $L(\theta)$ is the log-likelihood function according to each model, *m* is the number of clusters and *n* is the size of the data set. Under the identical spherical Gaussian assumption, the maximum likelihood estimate for the variance of the i^{th} cluster is:

$$\sum_i = \frac{1}{n_i - m} \sum_{j=1}^{n_i} \| x_j - C_i \|^2 \tag{2}$$

Where n_i is the size of each cluster, x_j is the j^{th} point in the cluster and C_i is the i^{th} cluster. For m clusters, the sum of log-likelihood of each cluster is as follows.

$$L(\theta) = \sum_{i=1}^{m} L(\theta_i) \tag{3}$$

Define $pr(x_i)$ as the probability of the i^{th} point in data sets, and $C_{p(i)}$ is the cluster corresponding to the partitioning. The variable d is the dimension of the data sets. Then, log-likelihood of the i^{th} cluster can be derived as follows:

$$\begin{aligned} L(\theta_i) &= \log \prod_{i=1}^{n_i} pr(x_i) = \sum_{i=1}^{n_i} \log pr(x_i) \\ &= \sum_{i=1}^{n_i} \log(\frac{n_i}{n} \frac{1}{(2\pi)^{d/2} \sum^{1/2}} \exp(-\frac{\| x_i - C_{p(i)} \|^2}{2\sum_i})) \\ &= \sum_{i=1}^{n_i} (\log \frac{n_i}{n} - \log((2\pi)^{d/2} \sum_i^{1/2}) - \frac{\| x_i - C_{p(i)} \|^2}{2\sum_i}) \\ &= n_i \log n_i - n_i \log n - \frac{n_i * d}{2} \log(2\pi) - \frac{n_i}{2} \log \sum_i - \frac{n_i - m}{2} \end{aligned} \tag{4}$$

To extend the log-likelihood of each cluster to all of the clusters, we use the fact that the log-likelihood of the points that belong to every cluster is the sum of the log-likelihood of the individual ones. So the total log-likelihood will be:

$$BIC = \sum_{i=1}^{m} (n_i \log n_i - n_i \log n - \frac{n_i * d}{2} \log(2\pi) - \frac{n_i}{2} \log \sum_i - \frac{n_i - m}{2}) - \frac{1}{2} m \log n \tag{5}$$

We use this BIC formula globally for each number of clusters in a predefined range. In general, m should be as small as possible according to [9]. Their strategy for the number of clusters is that a decisive first local maximum indicates strong evidence for the model size. However, according to our experiments, a good knee point detection method would be a better choice for deciding which local maximum has stronger evidence for the correct number of clusters.

2.3 Angle-Based Method

Some existing validity indices indicate the structure of data sets very well and contribute a lot to the problem. However, we can not directly obtain the correct number of clusters from these indices when they decrease or increase monotonously or only have some significant local changes. In this case, the structure of the dataset can be revealed by using a good knee point detection method. One efficient way is to calculate the difference between previous and afterward index values. There will be peaks at the points with significant local changes in the difference curve. It is also possible to consider more points of the curve in successive difference.

$$DiffFun(m) = F(m-1) + F(m+1) - 2*F(m) \tag{6}$$

Where *DiffFun* is the difference function, *F(m)* is the index value and *m* is the current number of clusters. It takes use of the previous, afterward and current values simultaneously. The disadvantage of the successive difference method is that it only considers several points instead of the whole curve allowing the index to find only local changes without a global perspective. If there are several local changes, then it may give a wrong result.

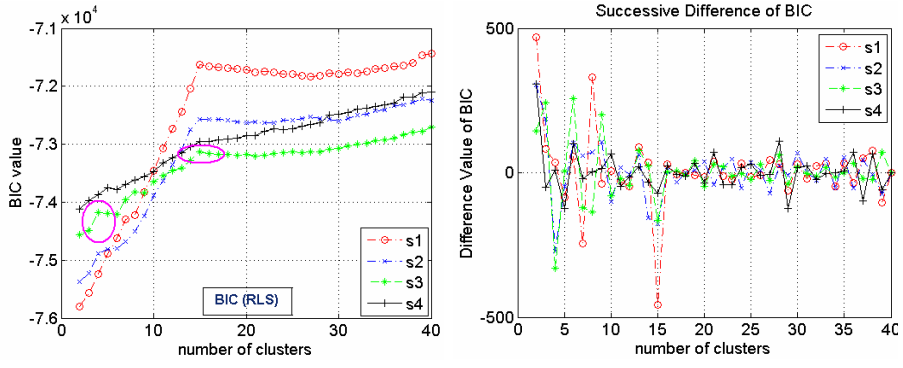


Fig. 1. Number of clusters vs. criterion metric graph of BIC on four datasets (s1-s4) (left), and successive difference (right)

In Fig.1, the calculated BIC values are plotted using four data sets with different degrees of cluster overlapping. There are at least two obvious jumps in each curve. The first decisive local maximum is usually considered to be the number of clusters in the original BIC. The successive difference graph also gives strong support on this rule. The problem is that the second local change (m=15) in the BIC curve also indicates strong evidence on the number of clusters in a global view. To decide which one is the optimal number, we take use of the angle property of a curve and propose an angle-based method to define and locate the optimal local knee (jump) in a graph of BIC.

Given a function *F(m)* of BIC where m is in the range [*min*, *max*]. Calculate the successive difference in terms of formula (6) to get the function difference *DiffFun* and detect n local significant changes by finding the first n minimum values in *DiffFun*. Here $n \leq m/2\text{-}1$ because at least 2 points can generate 1 trough. Sort the local minimum values in a decreasing order. Start from the points with bigger troughs; calculate the angle of those points by (7).

$$Angle = atan(1/|F(m)-F(m-1)|)+atan(1/|F(m+1)-F(m)|) \quad (7)$$

As in Fig.2 shows, the procedure will stop when the first maxima angle appears, which indicates the trend of the curve globally because it makes use of both the successive difference and angle property.

Angle-based Method on Knee Point Detecting Problem

Input: Graph(*m*) (*m*[*min*, *max*])
Output: Number of clusters *m*
Initialize:
Current_Value = Graph(*min*);
Previous_Value = Graph(*min*);
After_Value = Graph(*min*);
Begin:
for *m* = *min* to *max*
Current_Value = Graph(*m*);
After_Value = Graph(*m*+1);
DiffFunc = Previous_Value + After_Value - 2*Current_Value;
Previous_Value = Current_Value;
end
Find first *n* local minimas in DiffFunc
LocalMin[*n*] = (*m*, Current_Value, Previous_Value, After_Value);
for each *n* with decreasing order of LocalMin value,
angle[*n*] = AngleCalc(Current_Value, Previous_Value, After_Value);
Stop when the first maxima among the angles appear.
end
return *m* with the first maxima angle;

Fig. 2. Pseudo-code of the angle-based method on knee point detecting problem

3 Experimental Results

Here we use four two-dimensional artificially generated data sets denoted as s1 to s4 and one four-dimensional real data set Iris (Fig.3). The data sets s1 to s4 are generated with varying complexity in terms of spatial data distributions, which have 5000 vectors scattered around 15 predefined clusters with a varying degrees of overlapping. Iris is obtained from the UCI Machine Learning Repository. It contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The data sets can be found at:

- s1-s4: cs.joensuu.fi/~isido/clustering/
- Iris: www.ics.uci.edu/~mlearn/MLRepository.html

As the measures have to be tested on a certain clustering algorithm, we run K-means and Randomize Local Search (RLS) [14] clustering with m= [2, 30] in the case

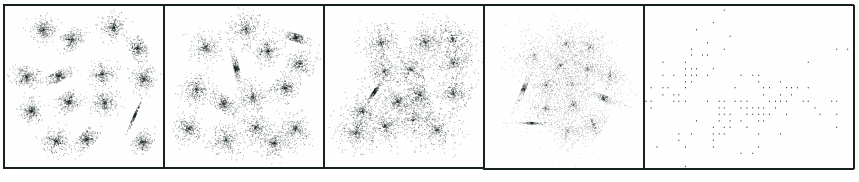


Fig. 3. The two dimensional visual of data sets: s1-s4 and Iris used for testing

of s1-s4, and m= [2, 10] in the case of Iris. To emphasize the effectiveness of the proposed method, we compare it with other measures:

- Dunn's index (DI) + maximum
- Davies-Bouldin's Index (DBI) + minimum
- Xie-Beni (XB) + minimum
- Bayesian Information Criterion (BIC) + first local maximum
- Angle-based BIC (ABIC).

Among them, DI, DBI and XB select the number of clusters either as the minimum or maximum value of the measure. We also report the results of the original BIC using the first local maximum as the number of clusters, and the proposed method.

Table 1. Detected number of clusters (m) of different cluster validity indices using RLS clustering algorithm (with 5000 RLS iterations and 2 K-means iterations)

Index	Data Set				
	s1	s2	s3	s4	Iris
DI	15	7	16	25	2
DBI	15	15	8	13	2
XB	15	15	4	13	2
BIC	15	4	4	5	3
ABIC	15	15	15	15	3

Table 2. Detected number of clusters (m) of different cluster validity indices using K-means clustering algorithm (20 iterations)

Index	Data Set				
	s1	s2	s3	s4	Iris
DI	2	2	2	2	2
DBI	15	15	11	16	2
XB	15	15	4	13	2
BIC	15	4	4	5	3
ABIC	15	15	15	15	3

In Table 1 and Table 2, we list the number of clusters found by different measures, data sets and clustering algorithms. Fig.4 visualizes the results for the other four measures with RLS and K-means clustering algorithms respectively. In Fig.5, we show the result of each step in our method with data sets s4 and Iris.

- DI gives clear maximum for the easiest data set (s1) but fails with more challenging ones. When K-means is applied with 20 iterations, it fails completely even with s1.
- DBI finds the correct minimum for s1 and s2, but the results for s3 and s4 indicate minimum somewhere around 10 and 15, resulting to uncorrected number of clusters (8, 13 and 2).

- XB takes the minimum as the number of clusters, which is clearly visible in the case of the easiest data set (s1). A correct result is also found for s2, but again, the index fails with the more demanding sets (s3, s4 and Iris).
- The original BIC, which considers the first decisive local maximum as the number of cluster gets the correct number only for s1 and Iris.
- The proposed ABIC provides accurate results in all cases.

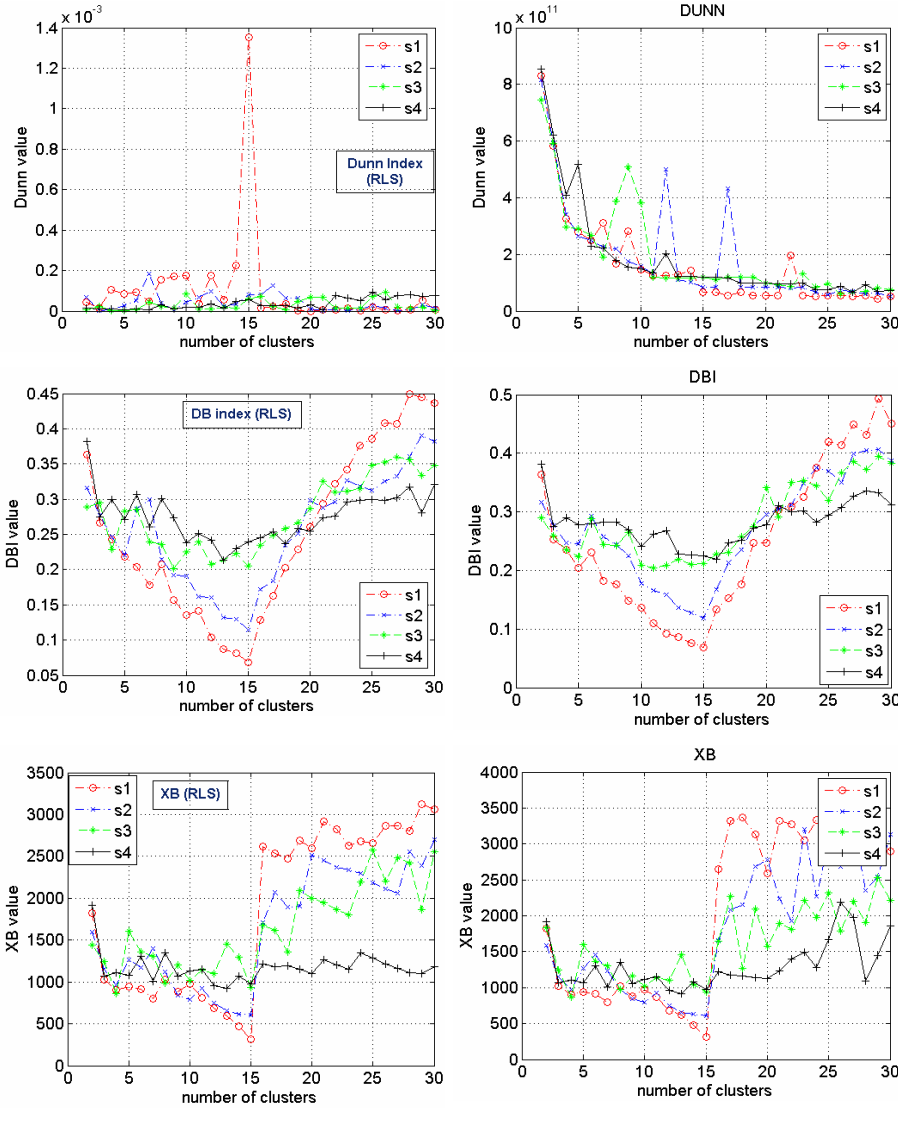


Fig. 4. Comparison of the other four measures on datasets s1 to s4 with two clustering algorithms: RLS clustering (left column) and K-means clustering (right column)

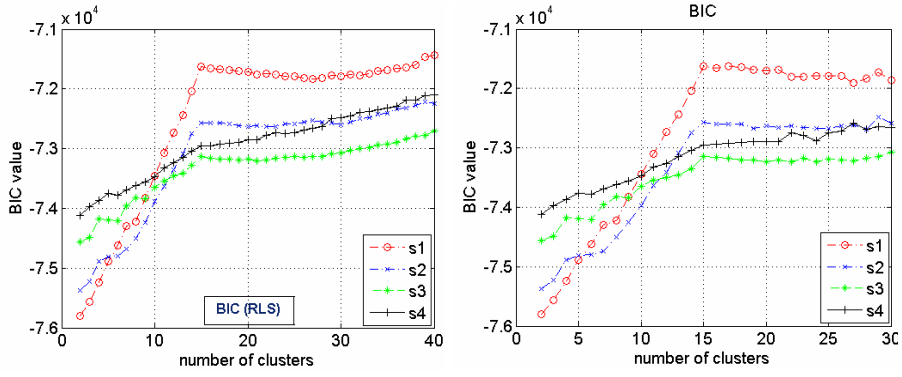


Fig. 4. (*continued*)

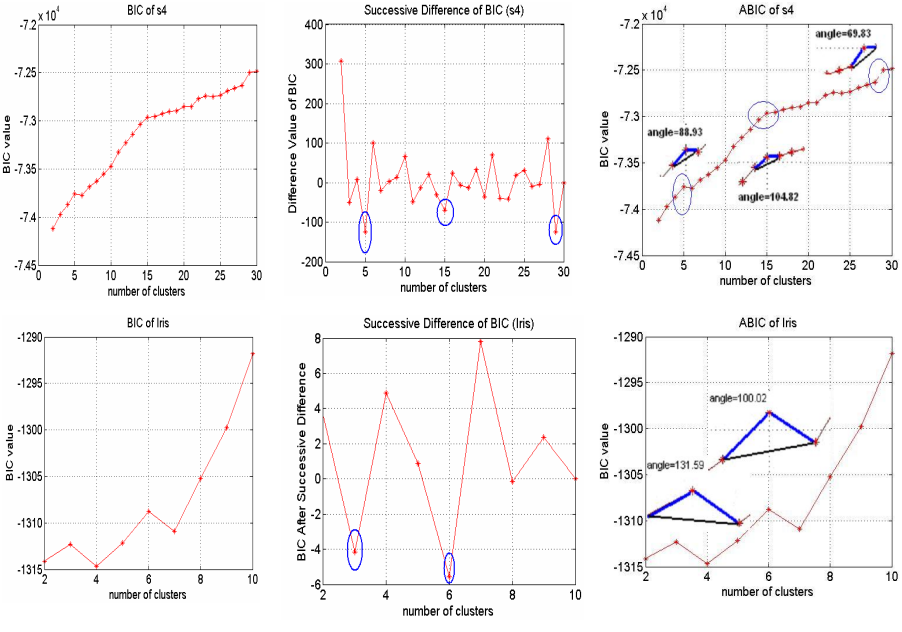


Fig. 5. Steps of the angle-based method on data sets s4 and Iris; BIC curve (left), the successive difference of BIC (middle) and the angles of the local significant changes (right).

4 Conclusions

We re-formulate BIC in partitioning based clustering, which shows a good prospect for determining the number of clusters. The original method to decide the knee point of BIC is to take the first decisive local maximum, which is not accurate enough according to our experiments. To improve the BIC for getting more reliable results, an angle-based method for knee point finding of BIC is proposed in this paper. As the

proposed method makes use of the global trend of the index curve, it is reliable to get the number of clusters. Experimental results also prove its effectiveness compared with other measures.

References

1. Milligan, G.W., Cooper, M.C.: An examination of procedures for determining the number of clusters in a data set. Psychometrika 50, 159–179 (1985)
2. Dimitriadou, E., Dolnicar, S., Weingassel, A.: An examination of indexes for determining the number of clusters in binary data sets. Psychometrika 67(1), 137–160 (2002)
3. Calinski, T., Harabasz, J.: A dendrite method for cluster analysis. Communication in statistics 3, 1–27 (1974)
4. Dunn, J.C.: Well separated clusters and optimal fuzzy partitions. Journal of Cybernetica 4, 95–104 (1974)
5. Bezdek, J.C., Pal, N.R.: Some new indexes of cluster validity. IEEE Transactions on Systems, Man and Cybernetics, Part B 28(3), 301–315 (1998)
6. Davies, D.L., Bouldin, D.W.: Cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1(2), 95–104 (1979)
7. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. on Pattern Analysis and Machine Intelligence 13(8), 841–847 (1991)
8. Kass, R.E., Raftery, A.: Bayes factors. Journal of the American Statistical Association 90(430), 773–795 (1995)
9. Frayley, C., Raftery, A.: How many clusters? Which clustering method? answers via model-based cluster analysis. Technical Report no. 329, Department of Statistics, University of Washington (1998)
10. Pelleg, D., Moore, A.: X-means: Extending K-means with efficient estimation of the number of clusters. In: Proceeding of the 17th International Conference on Machine Learning, pp. 727–734 (2000)
11. Krzanowski, W.J., Lai, Y.T.: A criterion for determining the number of groups in a data set using sum-of-squares clustering. Biometrics 44(1), 23–34 (1988)
12. Salvador, S., Chan, P.: Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In: Proceeding of the 16th IEEE International Conference on Tools with Artificial Intelligence, pp. 576–584 (2004)
13. Kass, R.E., Wasserman, L.: A reference Bayesian test for nested Hypotheses and its relationship to the Schwarz Criterion. Journal of the American Statistical Association 90(431), 928–934 (1995)
14. Fränti, P., Kivijärvi, J.: Randomized local search algorithm for the clustering problem. Pattern Analysis and Applications 3(4), 358–369 (2000)

Paper **P2**

©2008 ICTAI. Reprinted, with permission.
Q. Zhao, M. Xu and P. Fränti,
"Knee Point Detection on Bayesian Information Criterion", in *IEEE Int. Conf. Tools with Artificial Intelligence*, pp. 431–438, Dayton, Ohio, USA, 2008.

2008 20th IEEE International Conference on Tools with Artificial Intelligence

Knee Point Detection on Bayesian Information Criterion

Qinpei Zhao, Mantao Xu, Pasi Fränti
Speech & Image Processing Unit
Department of Computer Science, University of Joensuu
Box 111, Fin-80101 Joensuu
FINLAND
{zhao, franti}@cs.joensuu.fi, mantao.xu@carestreamhealth.com

Abstract

The main challenge of cluster analysis is that the number of clusters or the number of model parameters is seldom known, and it must therefore be determined before clustering. Bayesian Information Criterion (BIC) often serves as a statistical criterion for model selection, which can also be used in solving model-based clustering problems, in particular for determining the number of clusters. Conventionally, a correct number of clusters can be identified as the first decisive local maximum of BIC; however, this is intractable due to the overtraining problem and inefficiency of clustering algorithms. To circumvent this limitation, we proposed a novel method for identifying the number of clusters by detecting the knee point of the resulting BIC curve instead. Experiments demonstrated that the proposed method is able to detect the correct number of clusters more robustly and accurately than the conventional approach.

1. Introduction

One of the main difficulties for cluster analysis is that, the correct number of clusters for different types of datasets is seldom known in practice. However, most of clustering algorithms are designed only to investigate the inherited grouping or partition of data objects according to a known number of clusters. Thus, identifying the number of clusters is an important task for any clustering problem in practice albeit it must be faced with many operational challenges. A tractable way for cluster analysis is to ask the end user to input the number of clusters in advance, which needs the expert domain knowledge over the underlying datasets. On the other hand, many statistical criteria or clustering validity indices have been investigated in the sense of automatically selecting an appropriate number of clusters. Obviously, the clustering validity criteria must be carefully defined not only according to a presumably known data distribution of clusters but also to the specification of the input datasets. More importantly, those clustering validity criteria serve as a tool to measure the goodness of groups in clustering as well as a principle for selecting the "best" number of clusters meanwhile. A number of efforts have been made in the previous literatures, e.g., Milligan and Cooper [1] presented a comparison study over thirty validity indices for hierarchical clustering algorithms whereas Dimitriadou et al [2] conducted their comparison study over fifteen validity indices for the case of binary data.

However, one class of clustering methods, model-based clustering, has received considerable attention recently, in a framework of the estimation of Bayesian likelihood or the estimation of Bayesian parameters, e.g. the well-known EM algorithm. The model-based clustering combines both the advantage of the optimal model parameter estimation in model selection and the advantage of selecting the most appropriate number of mixture components [3]. In particular the mixture model approach allows for an approximation of Bayes factor [4] even if clusters are in distinctively different models. Thanks to Banfield and Raftery's intuitive approximation of twice logarithm of Bayes factor, called "AWE", the number of clusters can be identified directly according to the classification likelihood. The approximation of Bayes factor can be extended to a more general principle, *Bayesian Information Criterion* (BIC) [5-8] for the sake of selecting an appropriate number of model parameters or the number of clusters.

In order to seek an optimal number of clusters particularly for a large-scale clustering problem, one could apply an intuitively heuristic approach instead of using an optimization algorithm. A remarkable example is that of Thorndike [9] who identified the optimal number of clusters such that a flattening of the clustering validity curve or a knee point can be observed. In contrast to finding the maximum or minimum of clustering validity index, the knee point detection algorithm is more practical because most of clustering validity indices are monotonically decreased or increased [10] with the number of clusters. Clearly, seeking a maximum or minimum is intractable. The monotony of clustering validity indices hinges on the fact that the likelihood of

1082-3409/08 $25.00 © 2008 IEEE
DOI 10.1109/ICTAI.2008.154

IEEE computer society

the training data is undesirably improved when the number of clusters is increasing, which mainly results in overtraining problem if the number of parameters is too large. Of course, one could apply the successive difference of the clustering validity index, to seek the optimal number of clusters. However, most of those heuristic decision approaches are highly subjective or heuristic. For instance, the first decisive local maximum of BIC can be viewed as a good number of clusters but the resulting number of clusters is often inaccurate due to inefficiency of clustering optimization procedures. To overcome these difficulties, we propose a simple knee point detection algorithm for BIC in automatic detection of the number of clusters. The knee point detection algorithm is quite intuitive and heuristic since the clustering validity curve monotonically decreases or increases after the knee point. For simplicity of determining the number of clusters, we re-formulate BIC in the framework of partitioning based clustering.

The rest of the paper is organized as follows. The problem formulation is given in Section 2.1. The BIC method in partitioning based clustering is renewed in Section 2.2, and the proposed method is introduced in Section 2.3. The experiments on the proposed method are presented in Section 3. The results on different kinds of datasets demonstrate that the proposed method improves the original BIC knee point detection algorithm. Conclusions are drawn in Section 4.

2. Proposed Method

2.1 Preliminary

The problem of determining the number of clusters is defined here as follows:

Given a fixed number of clusters $m \geq 2$, and a specific clustering algorithm, find the clustering that best fits for the data set with different parameters. The procedure of identifying the best clustering scheme involves the following parts:

- Select a proper cluster validity index.
- Repeat a clustering algorithm successively for number of clusters, m from a predefined minimum to a predefined maximum.
- Plot the "number of clusters vs. criterion metric" graph and select the m at which the partition appears to be "best" in terms of the optimization on the criterion.

Based on this procedure, one can identify the best clustering scheme. The problem remains that how to select the optimal m for the validity index. Mean square error (MSE), for example, exhibits a decreasing monotony with respect to the number of clusters, m, whereas some clustering validity indices may embrace a local maximum or local minimum in the curve. Regardless of the monotony of the underlying clustering validity curve, in most cases, a significant local change could be observed on the curve, which is the so-called *knee* or *jump point*.

Locating the knee point in the validity index curve has not been well-studied. A straightforward approach is to compute difference of successive index values, for example, calculating the difference between previous and current values of the index. Other method, such as L-method [11] has been proposed to find the knee point of the curve by the boundary between the pair of straight lines that most closely fit the curve in Hierarchical / segmentation clustering. For some indices, the local maximum or minimum value will be considered as the knee point. However, if there are several local maximal (minimal) values, the challenge is to decide which one is the most suitable one to indicate the information of the data sets. According to the experimental results in our study, BIC indicates a good estimation in determining the number of clusters in partitioning based clustering. To improve the accuracy of BIC, a good knee point detection method is needed instead of taking the first local maximum.

2.2 Bayesian Information Criterion (BIC)

The *Bayesian Information Criterion* (BIC) has been successfully applied to the problem of determining the number of components in model-based clustering by Banfield and Raftery [12]. The problems of determining the number of clusters and the clustering problem are solved simultaneously.

We derive the formula of BIC based on Kass and Wasserman [13].

$$BIC = L(\theta) - \frac{1}{2} m \log n \qquad (\mathbf{1})$$

where, $L(\theta)$ is the log-likelihood function of data θ according to each model, m is the number of clusters and n is the size of the data set. Under the identical spherical Gaussian assumption, the maximum likelihood estimate for the variance of the i^{th} cluster is:

$$\sum_i = \frac{1}{n_i - m} \sum_{j=1}^{n_i} \| x_j - C_i \|^2 \qquad (\mathbf{2})$$

where C_i represents the i^{th} cluster or is the i^{th} cluster center, n_i is the size of the i^{th} cluster and x_j is the j^{th} point in the cluster. For m clusters, the sum of log-likelihood of each cluster is as follows.

$$L(\theta) = \sum_{i=1}^{m} L(\theta_i) \qquad (\mathbf{3})$$

Suppose that $pr(x_j)$ is the probability of the j^{th} data point in the data sets, and the variable d is the dimension of the data set. Then, log-likelihood of data belonging to the i^{th} cluster can be derived as follows:

$$L(\theta_i) = \log \prod_{j=1}^{n_i} (pr(C_i) \bullet pr(x_j)) = \sum_{j=1}^{n_i} \log(pr(C_i)\, pr(x_j))$$

$$= \sum_{j=1}^{n_i} \log(\frac{n_i}{n} \frac{1}{(2\pi)^{d/2} \sum_i^{1/2}} \exp(-\frac{\| x_j - C_i \|^2}{2\sum_i})) \quad (4)$$

$$= \sum_{j=1}^{n_i} (\log \frac{n_i}{n} - \log((2\pi)^{d/2} \sum_i^{1/2}) - \frac{\| x_j - C_i \|^2}{2\sum_i})$$

$$= n_i \log n_i - n_i \log n - \frac{n_i * d}{2} \log(2\pi) - \frac{n_i}{2} \log \sum_i - \frac{n_i - m}{2}$$

To extend the log-likelihood of each cluster to all of the clusters, the fact is applied that the log-likelihood of the whole data set is the sum of the log-likelihood of the individual cluster. Therefore the total log-likelihood will be:

$$BIC = \sum_{i=1}^{m} (n_i \log \frac{n_i}{n} - \frac{n_i * d}{2} \log(2\pi) - \frac{n_i}{2} \log \sum_i - \frac{n_i - m}{2}) - \frac{1}{2} m \log n \quad (5)$$

We use this BIC formula globally for each number of clusters in a predefined range. In general, m should be as small as possible according to [5]. Their strategy for the number of clusters is that a decisive first local maximum indicates strong evidence for the model size. However, according to our experiments, a good knee point detection method would be a better choice for deciding which local maximum has the strongest evidence for the correct number of clusters.

2.3 Knee Point Detection of BIC

In this section, we analyze the drawback of knee point detection on BIC by using the first decisive local maximum as the number of clusters. Successive difference on BIC is also analyzed. We then propose our knee point detection method on BIC called *DiffBIC* method in partitioning based clustering.

2.3.1 Existing Methods. There is a slight option difference on how to find the optimal value of BIC for cluster validity except the first decisive local maximum. However, our experimental findings indicate several local maximums in the BIC curve (see Fig.1) due to the fact that the clustering performance is highly subjective to the initial clustering guess or partition. Hence, the resulting first decisive local maximum could often be the local maximum approaching or very close to the initial guess. This can be observed in the BIC curve for dataset s3 in Fig.1: the first decisive local maximum is achievable at m=4 albeit the right number of clusters m is 15 where there is a more significant change of BIC (not a conventional knee point). The difference values of BIC for dataset s3 and s4 also reveal that detection of knee point for BIC may be faced with the same challenge as the first local decisive maximum. A more objective method of detecting the knee point of BIC curve is therefore demanded.

Several alternative techniques on knee point detection methods have been proposed in the literature. Successive difference of two adjacent points is one possible way and it can be calculated as: SD(n) = BIC(n-1)+BIC(n+1)-2*BIC(n); where n is the current point. However, it can locate the knee point only locally as it considers only several successive points in the curve as shown in Fig.1. According to the figure, we can find the highest differences for each dataset with successive difference at the points m_{opt}(s1)=15, m_{opt}(s2)=15, m_{opt}(s3)=4 and m_{opt}(s4)=5. The detected points offer the most significant changes of BIC but without taking into account of BIC value itself. Eventually, this method is not always reliable in particular when a local maximum close to the initial guess can be quickly obtained by clustering algorithms.

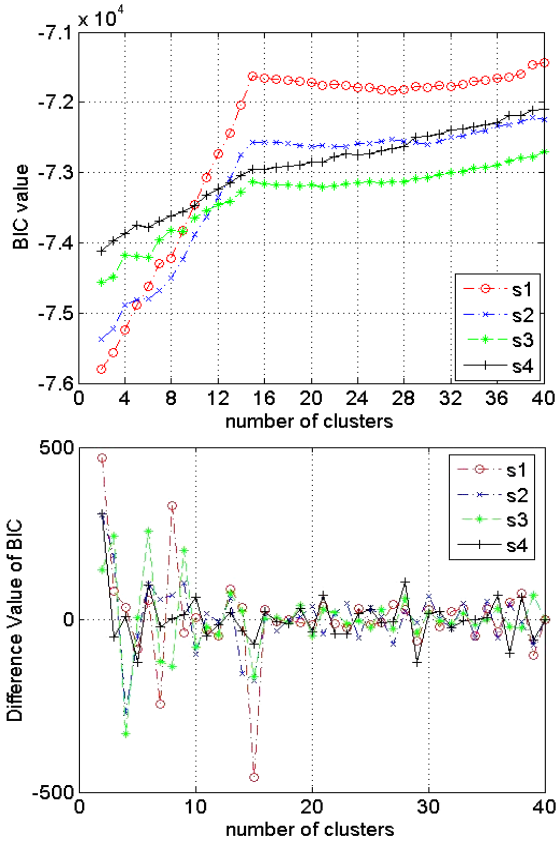


Figure 1. The original BIC curve (up) for datasets s1 to s4 obtained by RLS clustering algorithm [14] and successive difference of BIC (down).

2.3.2 DiffBic Function. We propose to combine both the information on BIC and the number of clusters m. The value of original BIC contains the information about the quality of clustering for each number of clusters. The knee point of BIC has to be the one that reflects this information overall. Two main features should be

satisfied i.e. the detected knee point can indicate the most significant change, and be as large as possible. The proposed method is designed based on these two features.

Given the range of m: $[m_{min}, m_{max}]$ where $m_{max} >> m_{opt}$ to contain the optimal m, obtain BIC value for each m. Normalize the obtained BIC value into the range of $[m_{min}, m_{max}]$ to get C_1. Then C_1 is divided by the number of clusters m getting the value Cm. This is further normalized into the same range of $[m_{min}, m_{max}]$ to obtain C_2. With the normalizations, C_1 and C_2 are under the same range. BIC_{max} and BIC_{min} in (6) represent the maximal and minimal value among the BIC values. Besides, Cm_{max} and Cm_{min} are respectively the maximal and minimal value among the Cm values.

$$C_1 = (m_{max} - m_{min})\,(BIC - BIC_{min})/(\,BIC_{max} - BIC_{min})$$

$$Cm = C_1 / m$$

$$C_2 = (m_{max} - m_{min})\,(Cm - Cm_{min})\,/(Cm_{max} - Cm_{min}) \qquad (6)$$

The value of Cm calculates the ratio between the normalized BIC value and the number of clusters, which reveals the global trend of the BIC curve as is shown in Fig.2. Each Cm value represents the angle α, which makes $\tan(\alpha)=C_1/m=Cm$. Whenever there is a local maximum in the original curve, angle α will indicate a difference.

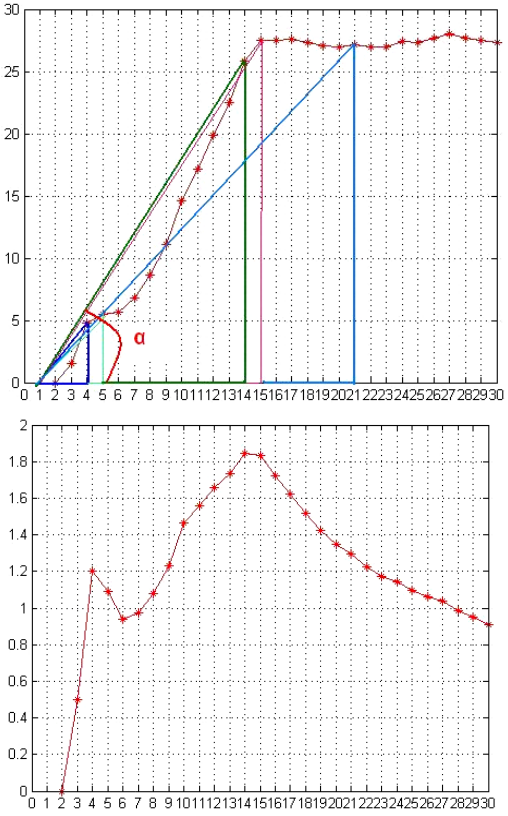


Figure 2. How the value of Cm (Normalized BIC value divided by the number of clusters) reveals the global trend. Normalized BIC curve (up); Result of Cm (down).

We consider two cases that the original BIC curve has globally increasing trend (case1) or decreasing trend (case2). Basically, a large BIC value is preferred to be the optimal m. In case1, the value depends on m_{max}, meanwhile in case2, on the other side, it depends on m_{min}. In case1, C_2 reaches several local maximums. When C_2 find the point that indicates the most significant change, it will not have an increasing trend anymore. The largest value of C_2 is considered as the most significant change. Thus, the sum of C_1 and C_2 will be calculated to reach the maximum information. In the other case, the original BIC has a decreasing trend, which makes C_2 to show a decreasing trend. As both of C_1 and C_2 are decreasing, the absolute subtraction of them is calculated to reach the most significant change. In both cases, two is divided in order to set the $DiffBic$ value into the same range of C_1.

$$DiffBic = \begin{cases} (C_1 + C_2)/2 \ldots\ldots\ldots\ldots case1 \\ |\,C_1 - C_2\,|/2 \ldots\ldots\ldots\ldots case2 \end{cases} \qquad (7)$$

2.3.3 Max Refinement. The range of m: $[m_{min}, m_{max}]$ is user-defined, which is assumed to contain the optimal m. Basically the most reliable way is $m_{max} = n$, n is the size of the dataset. However, m_{max} will be set as a more reasonable value in practice because of the heavy computation when $m_{max} = n$. In this paper, we define the m_{max} large enough, and then a max refinement is carried out.

There will be intersections across the C_1 and $DiffBic$ value in (7) because of the normalizations whenever the trend of the original BIC is increasing or decreasing. The positions of the intersection are affected by the setting of m_{min} and m_{max}. We assume that m_{max} is large enough to contain m_{opt}. With the assumption that $m_{max} \geq m_{opt}$, the first intersection $m=max'$ where $max' \neq m_{min}$ and $max' > m_{opt}$ exists. The value of max' can be thought as the refinement to m_{max} value. With this max refinement, the range of m can be reduced to $[m_{min}, max']$. There are two reasons for max refinement designing. One is that the original range setting is arbitrary; and the refined range is a smaller range that already contains the optimal value. The other is that BIC has an increasing or decreasing trend with the increment of the number of clusters, the points after the intersection has less information. Refine the original range $[m_{min}, m_{max}]$ into smaller one $[m_{min}, max']$ can make the decision accurately.

Finding the maximum value of $DiffBic$ in the new range: $[m_{min}, max']$, the optimal number of clusters is obtained by the proposed method. As Fig.3 shows, we get the second case for datasets s1 to s4. For each dataset, an intersection can be found to refine the max value. The maximum value of the proposed method is thought as the optimal number of clusters. According to this, the results from the proposed method is: m_{opt}(s1)= 15, m_{opt}(s2)= 15, m_{opt}(s3)= 15 and m_{opt}(s4)= 15.

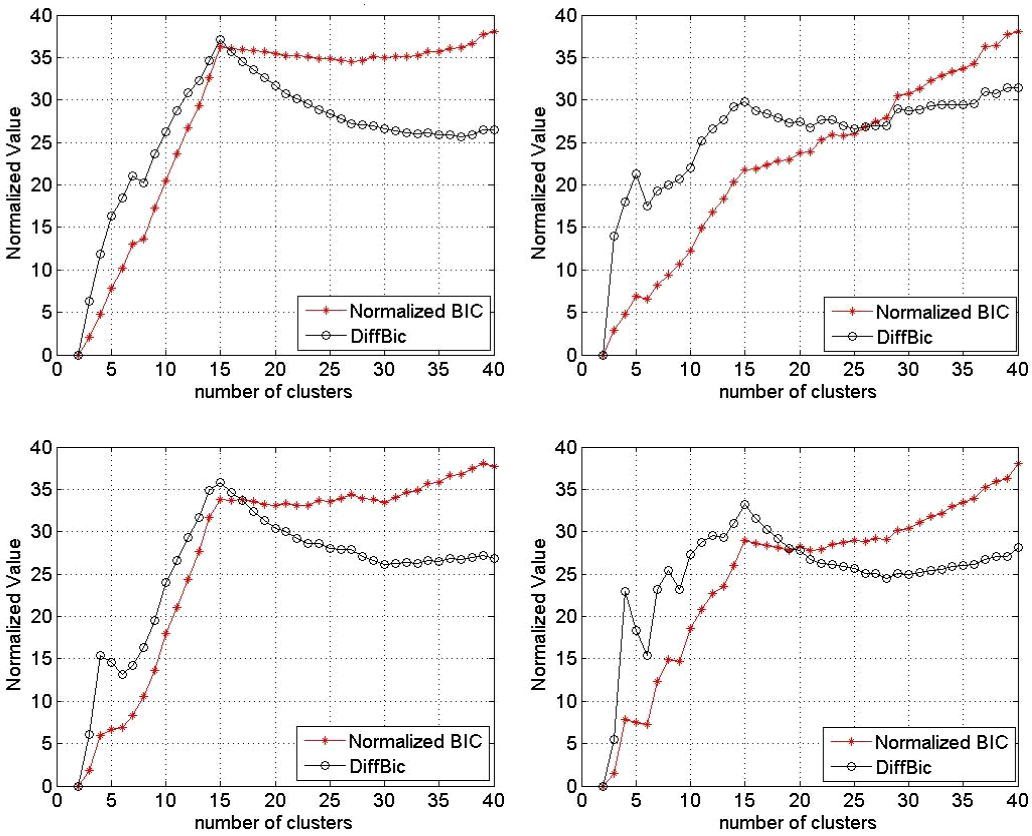


Figure 3. The results come from the proposed method for datasets s1 to s4 (left to right, up to down) with RLS clustering algorithm. Normalized BIC is represented as C1 in the context; DiffBic is the result from the proposed method.

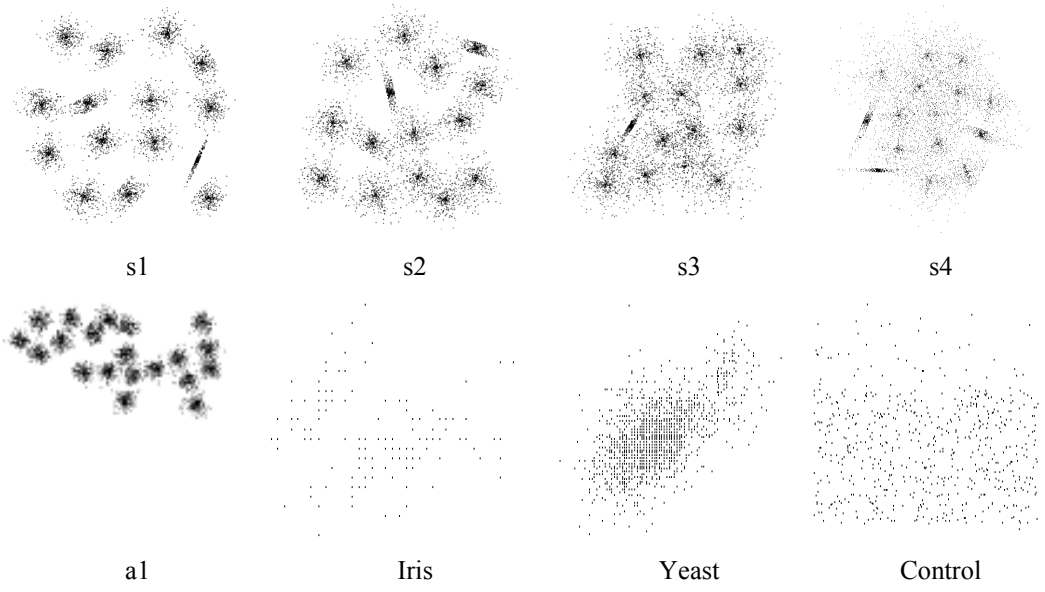


Figure 4. Two-dimensional visualization of the datasets for experiments.

We have further experiments on the proposed method with more datasets here. Both artificially generated datasets and real datasets are tested. The two dimensional view of the datasets is shown in Fig.4.

3. Experimental Results

The datasets s1 to s4 are generated with varying complexity in terms of spatial data distributions, which have 5000 vectors scattered around 15 predefined clusters with varying degrees of overlapping. The dataset a1 is generated in 2-dimensional Gaussian distribution. Datasets Iris, Yeast and Control are obtained from the UCI Machine Learning Repository. Iris contains 3 classes of 50 instances each, where each class refers to a type of iris plant. Yeast is originally used for protein localization sites prediction. The class distribution from a rule-based expert system indicates the optimum number of clusters as 10. However, as the size of 6 clusters among them is too small, our clustering algorithms reach 5 clusters as the optimal clustering. Dataset Control contains 600 examples of control charts synthetically generated by the process of Alcock and Manolopoulos (1999). There are six different classes of control charts.

The data sets can be found here:

- s1-s4, a1: http://cs.joensuu.fi/~isido/clustering/
- Iris, Yeast, Control: www.ics.uci.edu/~mlearn/MLRepository.html

Table 1. Data sets with their properties including the size of the dataset, dimension, the number of clusters and how they have been generated.

Data Set	Size	Dimension	No. of Clusters	Generated
s1-s4	5000	2	15	synthetic
a1	3000	2	20	synthetic
Iris	150	4	3	real
Yeast	1484	8	5	real
Control	600	NA	6	real

As cluster validity criterion is related to clustering algorithm, we test the revised BIC on both K-means and Randomized Local Search (RLS) [14] clustering algorithms. The RLS method is run using 5000 iterations and 2 K-means iterations within the algorithm. Meanwhile, in the K-Means clustering algorithm, 20 iterations are used for synthetic datasets (s1-s4, a1), 200 iterations for Iris and Yeast, and 500 iterations for control dataset. The proposed knee point detection method is then applied to the calculated BIC value. The results from different datasets by the proposed method with K-means and RLS clustering algorithms are summarized in Fig.5 and Fig.6.

The results from different datasets with RLS clustering algorithm are all visible and correct. However, the results from the K-means clustering algorithm are not good for real datasets even if the number of iterations is well-tuned. The datasets Control gets the result m_{opt}(control)=5. This can not prove the failure on our knee point detecting method; the actual reason is the K-Means clustering algorithm itself. Table 2 shows the results from different knee point detection methods on BIC.

4. Conclusions

Determining the number of clusters is one of the most difficult problems in cluster analysis. We re-formulate BIC in partitioning based clustering, which shows good prospect for determining the number of clusters. The original method to decide the knee point of BIC is to take the first decisive local maximum, which is not accurate enough according to our experiments. To improve the BIC for getting more reliable results, a new knee point detecting method of BIC is proposed in this paper. As the proposed method takes advantage of the information of criterion and number of clusters, it is reliable to get the optimal results. Experimental results on different kinds of data sets also prove its effectiveness.

Table 2. The number of clusters obtains from different knee point detection method on BIC. BIC represents the first local maximum. SD is the successive difference on BIC. Cm is the value that gets from the normalized BIC value divided by the number of clusters, taking the maximum as its optimal value. DiffBic represents the proposed knee point detection method.

Method	Data Sets							
	s1	s2	s3	s4	a1	Iris	Control	Yeast
BIC	15	4	4	5	3	3	2	2
SD	15	15	4	5	3	17	2	2
Cm	15	14	4	14	3	NA	2	2
KP	15	15	15	15	20	3	6	5

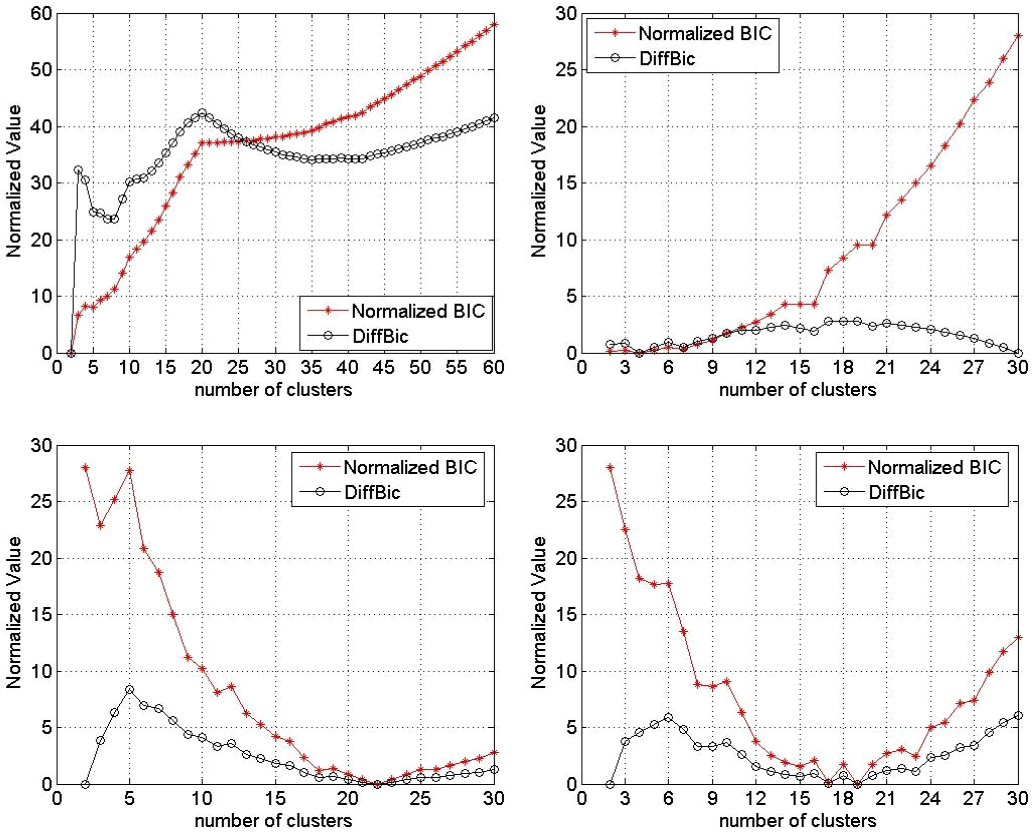
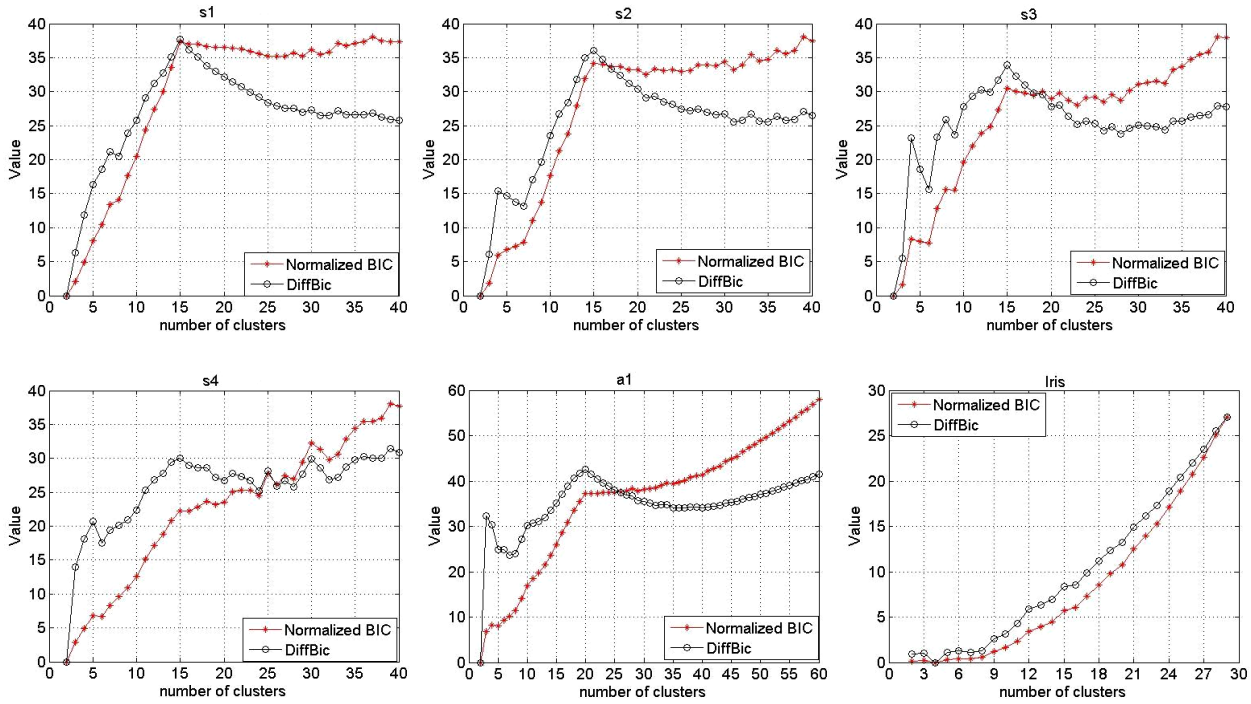


Figure 5. Results on different datasets (a1, Iris, Yeast, Control from left to right, top to down) with RLS clustering algorithm; Normalized BIC is represented as C1 in the context; DiffBic is the result from the proposed method.



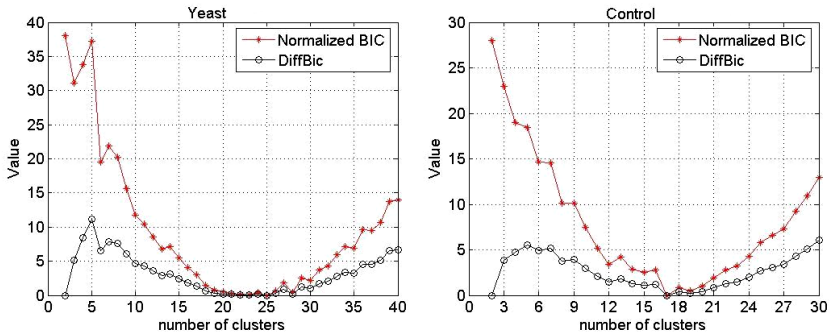


Figure 6. Results on different datasets (a1, Iris, Yeast, Control from left to right, top to down) with K-Means clustering algorithm; Normalized BIC is represented as C1 in the context; DiffBic is the result of the proposed method.

5. Acknowledgements

This research is supported by CIMO fellowship.

Reference:

[1] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, Vol.50, pp. 159-179, 1985.

[2] E. Dimitriadou, S. Dolnicar, and A. Weingassel. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, Vol.67, No.1, pp. 137-160, 2002.

[3] X.L. Hu and L. Xu. Investigation on several model selection criteria for determining the number of cluster. *Neural Information Processing*, Vol. 4, No.1, July 2004.

[4] R.E. Kass and A. Raftery. Bayes factors. *Journal of the American Statistical Association*, Vol.90, No.430, pp. 773-795, 1995.

[5] C. Frayley and A. Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The computer Journal*, Vol.41, No.8, pp. 578-588, 1998.

[6] A. Dasgupta and A. Raftery. Detecting features in spatial point process with clutter via model-based clustering. *Journal of the American Statistical Association*, 93, pp. 294-302, 1998.

[7] D.Pelleg, A.Moore: X-means: Extending K-means with efficient estimation of the number of clusters. *Proceeding of the 17th International Conference on Machine Learning*, pp.727-734, 2000.

[8] S.S. Chen and P.S. Gopalakrishnan. Clustering via the Bayesian information criterion with applications in speech recognition. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol.2, pp. 645-648.

[9] R.L. Thorndike. Who belongs in the family? *Psychometrika*, Vol. 18, 267-276, 1953.

[10] W.J. Krzanowski, Y.T.Lai, A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, Vol.44, No.1 (Mar., 1988), pp.23-34.

[11] S. Salvador and P. Chan. Determining the number of clusters / segments in hierarchical clustering / segmentation algorithms. *Proceeding of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 576-584, 2004.

[12] J.D. Banfield and A.E. Raftery. Model-Based Gaussian and Non-Gaussian Clustering. *Biometrics*, Vol. 49, pp. 803-821, 1993.

[13] R.E. Kass and L. Wasserman. A reference Bayesian test for nested Hypotheses and its relationship to the Schwarz Criterion. *Journal of the American Statistical Association*, Vol. 90, No. 431, pp.928-934, 1995.

[14] P. Fränti and J. Kivijärvi. Randomized local search algorithm for the clustering problem. *Pattern Analysis and Applications*, Vol.3, No.4, pp. 358-369, 2000.

Paper **P3**

©2009 Springer Science + Business Media. Reprinted, with permission.

Q. Zhao, M. Xu and P. Fränti,
"Sum-of-Squares Based Cluster Validity Index and Significance Analysis", in *Int. Conf. on Adaptive and Natural Computing Algorithms*, pp. 313–322, Kuopio, Finland, 2009.

Sum-of-Squares Based Cluster Validity Index and Significance Analysis*

Qinpei Zhao, Mantao Xu, and Pasi Fränti

Department of Computer Science, University of Joensuu
Box 111, Fin-80101 Joensuu
Finland
{zhao,franti}@cs.joensuu.fi, mantao.xu@carestreamhealth.com

Abstract. Different clustering algorithms achieve different results with certain data sets because most clustering algorithms are sensitive to the input parameters and the structure of data sets. The way of evaluating the result of the clustering algorithms, cluster validity, is one of the problems in cluster analysis. In this paper, we build a framework for cluster validity process, while proposing a sum-of-squares based index for purpose of cluster validity. We use the resampling method in the framework to analyze the stability of the clustering algorithm, and the certainty of the cluster validity index. For homogeneous data based on independent variables, the proposed clustering validity index is effective in comparison to some other commonly used indexes.

1 Introduction

Clustering is an unsupervised process which intends to discover the unknown structure of data sets accurately. There are a number of clustering algorithms [1] based on different strategies and they are developed to satisfy with different needs from the data sets. The common sense is that there is no general algorithm applicable to all kinds of data sets. The problem comes up that how to evaluate the effect of clustering algorithms on different data sets. Cluster validity provides the way of validating the quality of clustering algorithms and the means of discovering the natural structure of the data sets. If cluster analysis is to make a significant contribution, much more attention must be paid to the cluster validity issues. Cluster validity measures are the methods, which can not only compare the results of two different sets of clustering algorithms to determine the better one, but determine the "correct" number of clusters in the data set.

Amounts of cluster validity indexes have been proposed. Milligan and Cooper [2] have presented a comparison study over thirty validity indexes for hierarchical clustering algorithms whereas Dimitriadou et al [3] conducted their comparison study over fifteen validity indexes for the case of binary data. Different indexes under different situations achieve different results. We introduce several indexes mentioned in these two literatures for purpose of comparison.

* Thanks to Nokia Foundation for financial support.

M. Kolehmainen et al. (Eds.): ICANNGA 2009, LNCS 5495, pp. 313–322, 2009.
© Springer-Verlag Berlin Heidelberg 2009

We separate the indexes in this paper into two types, one is sum-of-squares based type, and the other is classical type. The methods in the first type measure the dispersion of the data points within a cluster and between the clusters respectively. The indexes are:

- Ball and Hall [4], the maximum value of the successive difference is determined as the optimal number of clusters.
- Calinski and Harabasz [5], the minimum value of the successive difference is determined as the optimal number of clusters.
- Hartigan [6], the minimum value of the successive difference is determined as the optimal number of clusters.
- Xu [7], the maximum value can be determined as the optimal number of clusters, the successive difference is applicable but not necessary.

The classical measures are mostly proposed in different area and perform well to some extend. These measures share the advantage of using the maximum or minimum value as the optimal number of clusters.

- Dunn's index [8], the maximum of the index value is determined as the optimal number of clusters.
- Davies-Bouldin index [9], the minimum of the index value is determined as the optimal number of clusters.
- Xie-Beni's separation index [10], the minimum of the index value is determined as the optimal number of clusters.
- Bayesian Information Criterion [11], which is a model selection criteria. The first local maximum is determined as the optimal number of clusters.
- Silhouette Coefficient [12], the maximum of the index value is determined as the optimal number of clusters.

Applications of resampling method, such as bootstrapping, subsampling, or cross validation to cluster validity are not new in the cluster validity. Peck et al. [13] developed a bootstrap-based procedure to obtain approximate confidence bounds on the number of clusters in the "best" clustering. Ben-Hur et al. [14] presented a method that exploited measurements of the stability of clustering solutions obtained by perturbing the data set. Cluster validation by prediction strength [15] considered clustering as a classification problem, which used the way of cross validation technique. Dudoit and Fridlyand [16] introduced a prediction-based sampling method, CLEST, in which, the data was first split into two non-overlapping sets. Then the learning set was clustered and a classifier was built using the obtained labels; the test set was also clustered and the obtained labels were compared using an external index.

We establish a framework of cluster validity process with resampling methods to validate the clustering algorithm and the validity index. Moreover, a sum-of-squares based index is proposed. The rest of the paper is organized as follows. We introduce the framework of the cluster validity in Section 2. The proposed index is formulated in Section 3. Experiments on the proposed method are presented in Section 4, in which the results on both artificial generated and real data sets are also displayed. Two clustering algorithms are applied in the experiment. A further step on variability and certainty analysis is introduced in Section 5. Conclusions and future work are drawn in Section 6.

2 Related Work

Cluster validity relates to the clustering algorithms. The fundamental clustering problem is to partition a given data set into groups, so that the points in the same group are more similar to each other than the points in different groups. Thus, one way of the cluster validity is to analyze within-between group variance.

Let $X = \{x_1, x_2 \ldots x_n\}$ be a set of data with n samples. Suppose the samples in X have hard labels that mark them as representatives of m non-overlapping clusters, says $C = \{C_1, C_2 \ldots C_m\}$. The clustering algorithm is to find the optimal partition $P = \{P_1, P_2 \ldots P_m\}$. The most important parameter among them is the parameter m, the number of clusters, because most of the clustering algorithms require the parameter m as the input and thus the clustering result is also affected by it.

Given the data set X, a specific clustering algorithm, and a fixed range of number of clusters, the basic procedure of the cluster validity involves the following steps:

- Fix the data sets with external information.
- Repeat the clustering algorithm successively for the number of clusters, m from a predefined minimum m_{min}, to a predefined maximum m_{max}.
- Get the clustering results: partitions and codebooks. Calculate the index value of each number of clusters.
- Plot the "number of clusters vs. index metric" graph and select the m at which the partition appears to be the "best" according to how the index is optimized.
- Compare the detected number of clusters (m^*) with the "external information" to prove the effectiveness of the index.

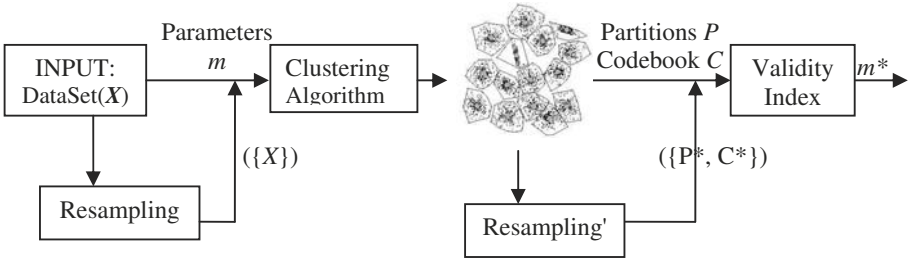


Fig. 1. Scheme diagram of cluster validity process

The clustering algorithm can be any of the existing algorithms. We use the Random Local Search algorithm (RLS) [17] in the validity procedure. The RLS clustering algorithm shares the advantage of both the k-means and the local search. To eliminate the effect on index from the clustering algorithm, K-means clustering, the most typical clustering algorithm is also tested in this paper.

Based on this procedure, we can easily have the scheme diagram of cluster validity in Fig.1. To estimate the stability of the clustering algorithm, we could use resampling method as is shown in the *resampling* part. Furthermore, in order to exclude the effect

of data sets and clustering algorithm, another resampling method is employed, as the *resampling'* part shows. This part will be shown in section 4 in detail.

Basically, comparison is essential to prove the effectiveness. The two types' indexes mentioned above are compared to the proposed index in the experiments section.

3 Proposed Method

In cluster analysis, the within group variance and between group variance can be calculated by *sum-of-squares within cluster* (SSW) and *sum-of-squares between clusters* (SSB) respectively. We analysis the existing index based on SSW and SSB, and then propose a sum-of-squares based method, so-called WB-index.

The value of SSW is defined as:

$$SSW(C,m) = \frac{1}{n}\sum_{i=1}^{m}\sum_{j\in C_i} \| x_j - C_{P(j)} \| \tag{1}$$

which is minimized over all *m*-partitions *C* in the clustering procedure. According to ANOVA, the *total sum-of-squares* (SST) can be decomposed into two parts that are SSW and SSB for any partition *C*.

$$SSB(C,m) = \frac{1}{n}\sum_{i=1}^{m} n_i \| C_i - \bar{x} \| \tag{2}$$

where n_i is the number of elements in each cluster, and $\bar{x}$ is the mean value of the whole data set, *m* is the number of clusters. Hence, we can now define a generalized *within-between cluster type* (SSWB) in Eq.3, which is a function of the SSW or SSB:

$$SSWB = function(SSW(C,m), SSB(C,m)) \tag{3}$$

Table 1. Sum-of-squares based indexes

No.	Index Name	Formula
1	Ball & Hall	SSW/m
2	Calinski&Harabasz	$CH = \frac{SSB/(m-1)}{SSW/(n-m)}$
3	Hartigan	$H-index = -\log(SSW/SSB)$
4	Xu	$Xu = d\log(\sqrt{SSW/(dn^2)}) + \log(m)$

The sum-of-squares based methods above (table.1) are all based on the property of the SSW and SSB. We study these indexes in Fig.1. As in Fig1.(a) shows, the trends of normalized SSW and SSW/SSB are almost same, indicating that the factor of the SSW has a more important effect in the ratio of SSW/SSB. In other *WB-type* indexes except for Xu's index, we find that they either monotonously increase/decrease or need

additional knee point detection method, such as successive difference in order to get the optimal number of clusters. Xu's index has clear minimum knee point; however, our experiments in section 4 will show it doesn't work well on real data sets.

Thus, we propose a simpler sum-of-square method, WB-index as:

$$WB = m \bullet SSW / SSB \tag{4}$$

We emphasize the effect of SSW with multiplying the number of clusters. The advantages of the proposed method are that it determines the number of clusters by minimal value of it without any knee point detection method, and it is easy to be implemented.

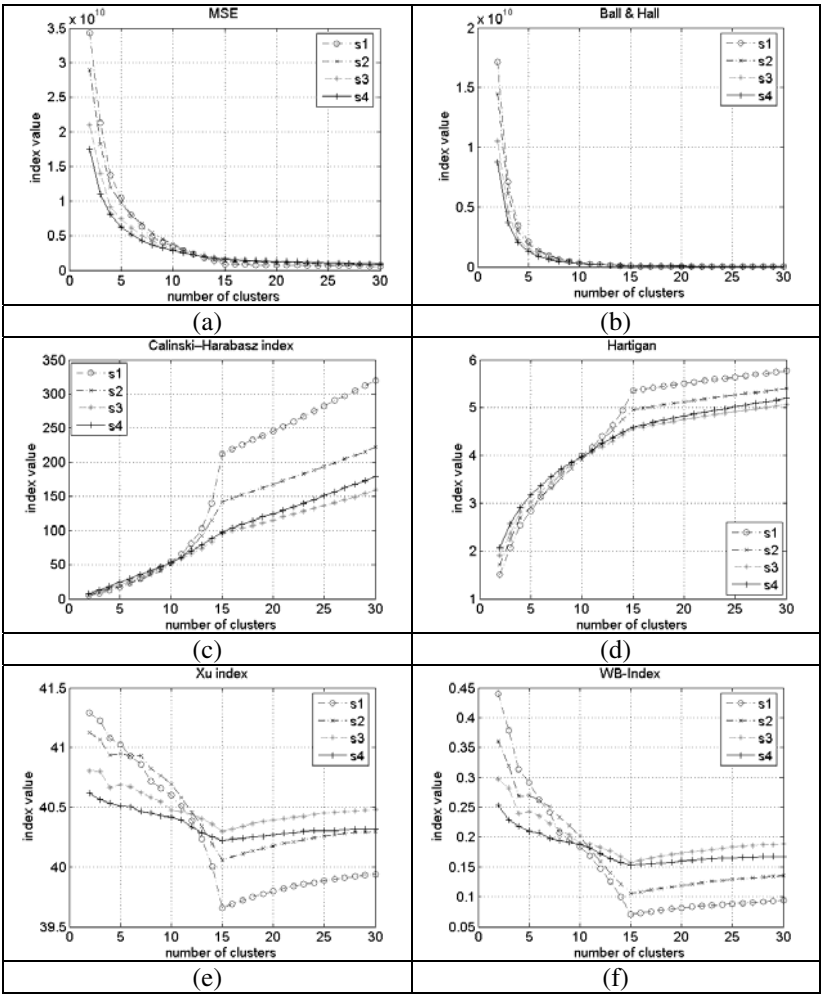


Fig. 2. (a). Comparison of SSW and SSW/SSB; (b)-(f). Comparison of several sum-of-square based indexes with four artificial data sets (s1-s4).

4 Experimental Results

In this paper, we test the methods with the data sets in table.2. The data sets s1 to s4 are generated with varying complexity in terms of spatial data distributions, which have 5000 vectors scattered around 15 predefined clusters with a varying degrees of overlap. The datasets a1 and R15 are generated in 2-dimensional Gaussian distribution. Iris and Breast are the real data sets obtained from the UCI Machine Learning Repository. Iris is a four-dimensional data set, containing three classes of 50 instances each, in which each class refers to a type of iris plant. The second real data set is the Wisconsin breast cancer data set (Wolberg and Mangasarian, 1990).

For purpose of comparison, we test five other classic measures:

- Dunn's index (DI)
- Davies-Bouldin's Index (DBI)
- Xie-Beni (XB)
- Bayes Information Criterion (BIC)
- Silhouette Coefficient (SC)

In the special case of m=1, SSW equals to SST. Clustering algorithm is therefore performed by m=[2,30] in the case of S1-S4, and m=[2,10] in the case of the real data sets.

Table 2. Information of the data sets in the experiments

DataSet	Size	Dimension	# of clusters	Generated
s1-s4	5000	2	15	artificial
a1	3000	2	20	artificial
R15	600	2	15	artificial
Breast	699	11	2	real
iris/Iris	150	4	3	real

Table 3. Results using the RLS (with 5000 RLS iterations and 2 K-means iterations)

DataSet	BH*	CH*	Har*	Xu	DI	DBI	XB	SC	BIC*	WB-INDEX
s1-s4	3	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
	3	**15**	4	**15**	7	**15**	**15**	**15**	4	**15**
	4	**15**	4	**15**	16	8	4	**15**	4	**15**
	3	**15**	3	**15**	25	13	13	**15**	5	**15**
a1	3	**20**	3	**20**	34	**20**	**20**	**20**	3	**20**
R15	3	**15**	**15**	**15**	2	**15**	**15**	**15**	8	**15**
Breast	3	3	3	NA	14	**2**	**2**	**2**	**2**	**2**
Iris	**3**	**3**	**3**	NA	2	2	2	9	6	**3**

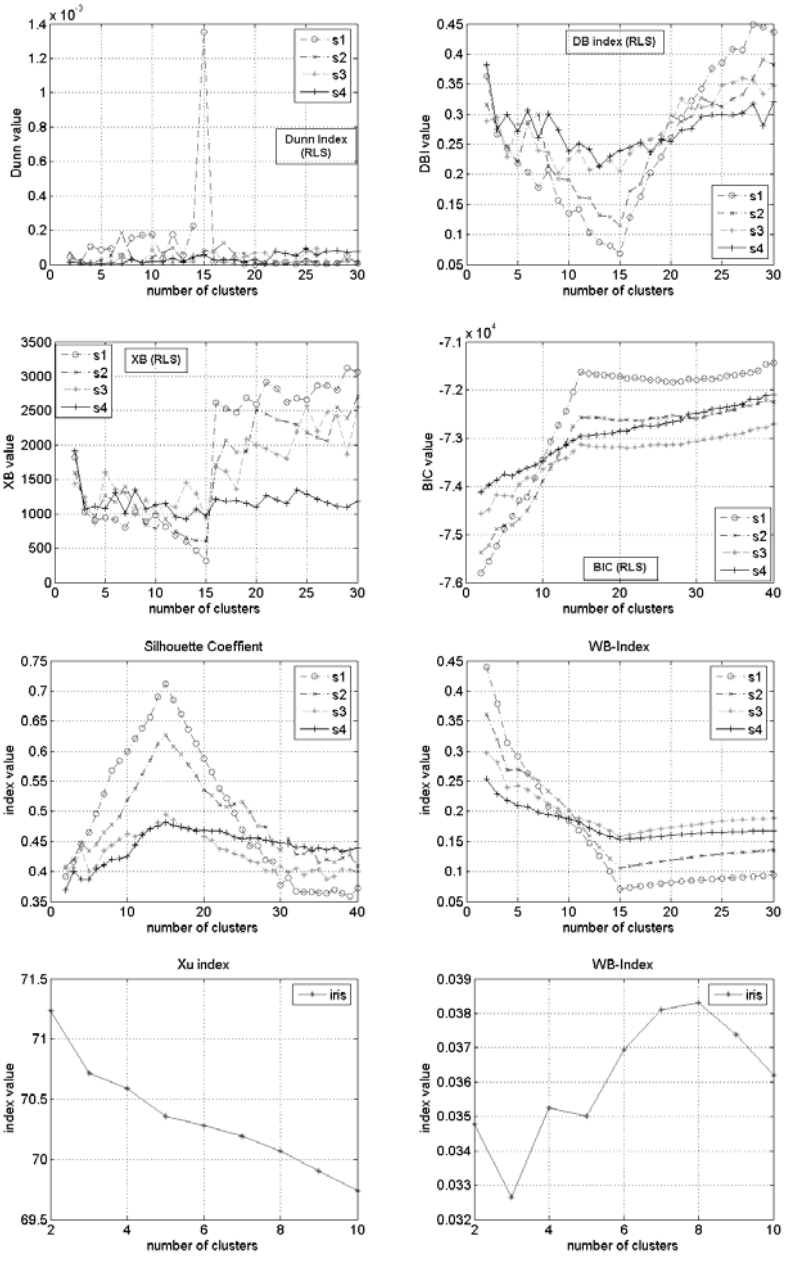


Fig. 3. The results with different validity indexes and data sets. The results of Xu's index and the proposed index on real data set Iris are on the last row. It is unable to find the minimum value of Xu's index as the optimal number of clusters as it is monotonously decreasing.

4 Significance Analysis

The results of the experiments with different clustering algorithms and data sets demonstrate that the proposed index can provide an accurate estimation of the number of clusters, which also shows the effectiveness of the cluster validity. Fundamentally, we can demonstrate the proposed index as it shows in the experiments. Moreover, we want to confirm the results in this section by further significance analysis.

4.1 Variability Analysis

With an uncertain distribution of the results, resampling method can be employed as a natural approach for the variability estimation associated with each index value. As in the process shows (Fig.1), we could resample on the original data set (X), get a new data set (X^*) and apply the new data set for the validation procedure again. Repeat the resampling B times, deal with the B times index values to get the statistical significance. However, the RLS clustering algorithm is designed with randomization, in which there is random swapping of the code vectors. Hence, we keep the data set unchanged and utilize the randomization of the clustering algorithm by running B times to analyze the results.

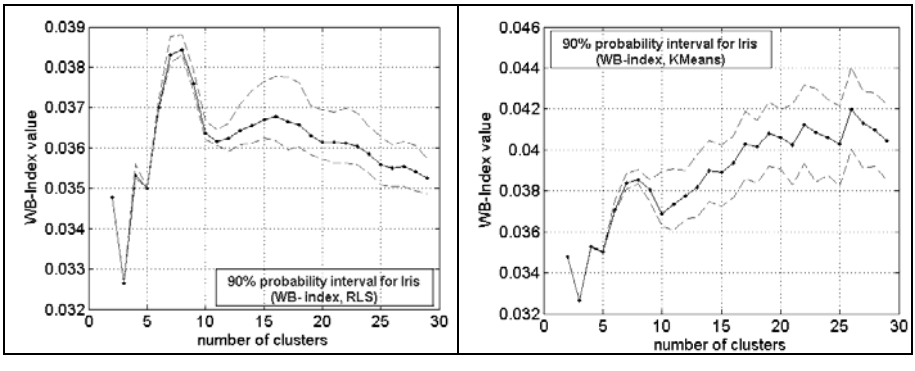


Fig. 4. 90% probability interval of the WB-index with the RLS and KMeans clustering on the data set Iris

Quartile range is one of the measures used to estimate variability. We use it into our scheme to analyze the variability of each index value. With the same setting of input parameters, fix the number of clusters, and run the clustering algorithm B times to get B values on the same number of clusters. Then the 5^{th} and 95^{th} percentiles of the B index values are calculated to get 90% probability range.

Iris data as a real data set is a representative to be tested. According to the results, only the proposed method and the BIC with knee detection get the correct number on Iris. In this case, both of the clustering algorithms with the same data set and index are tested. We run B = 100 times of the clustering algorithm with the same input parameters setting. The 90% probability interval with the RLS and K-means is shown

respectively in Fig.4, and the dash line is the boundary of the range. It is clear that the range m = [2, 3] strongly indicates the optimal number of clusters with the RLS clustering; and m = [2, 5] with the K-means clustering. The range of m is wider with the K-means clustering than the RLS clustering. Thus we can conclude that RLS clustering is more stable than the K-means and the variability of the K-means on m = 3 is convincingly larger than that of the RLS on the Iris.

4.2 Certainty Analysis

We develop another way to prove the certainty of the proposed index, which employs the resampling method. As Fig.1 shows, the effect of the validity index is affected by the data set and the clustering algorithm. In this case, resampling the data set cannot prevent the effect coming from the clustering algorithm. Hence, we process the resampling method on the partitions getting from the clustering to avoid this problem.

In the first run of the validity procedure, a set of partitions (P) is generated. Basically, this set of partitions is the optimal one according to the clustering algorithm. A WB-index value (WBI) is obtained on P. We permute the original partitions (P) by B times, get $\{P^*\}$, and recalculate the index values $\{WBI^*\}$. As the optimal value of the WB-index should be as small as possible, we can estimate the certainty by counting the probability that $WBI^* \leq WBI$.

$$P = \frac{No.(WBI^* \leq WBI)}{TotalNo.(WBI^*)} \tag{5}$$

The smaller the probability P is, the more certainty the method obtains. It is not practical to calculate all possible permutations due to the involved time. Generally, at least B =1000 times permutations should be done. In this paper, 1000 random permutations were performed on the partitions (Fig.5). It indicates the certainty of the index, as the observed optimal value is much smaller than any of the values obtained under permutation.

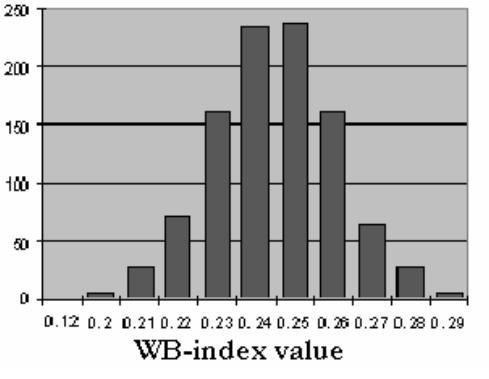


Fig. 5. Distribution of the WB-index on Iris data set (m=3) for 1000 permutations of the partitions with the RLS clustering. The "optimal" value of the WBI is very extreme by reference to this distribution (WBI = 0.032653).

5 Conclusions

We represented a framework with the resampling step for the estimation on the stability of the clustering algorithm and the variability of the validity index in cluster validity process. In addition, we proposed a new sum-of-squares based index which indicates simplicity and good prospect compared to other indexes. Based on the proposed index, we completed the whole process of the cluster validity.

References

1. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks 16(3), 645–678 (2005)
2. Milligan, G.W., Cooper, M.C.: An examination of procedures for determining the number of clusters in a data set. Psychometrika 50, 159–179 (1985)
3. Dimitriadou, E., Dolnicar, S., Weingassel, A.: An examination of indexes for determining the number of clusters in binary data sets. Psychometrika 67(1), 137–160 (2002)
4. Ball, G.H., Hubert, L.J.: ISODATA, A novel method of data analysis and pattern classification (Tech. Rep. NTIS No. AD 699616). Standford Research Institute, Menlo Park (1965)
5. Calinski, T., Harabasz, J.: A dendrite method for cluster analysis. Communication in statistics 3, 1–27 (1974)
6. Hartigan, J.A.: Clustering algorithms. Wiley, New York (1975)
7. Xu, L.: Bayesian Ying-Yang machine, clustering and number of clusters. Pattern Recognition Letters 18, 1167–1178 (1997)
8. Dunn, J.C.: Well separated clusters and optimal fuzzy partitions. Journal of Cybernetica 4, 95–104 (1974)
9. Davies, D.L., Bouldin, D.W.: Cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1(2), 95–104 (1979)
10. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. on Pattern Analysis and Machine Intelligence 13(8), 841–847 (1991)
11. Frayley, C., Raftery, A.: How many clusters? Which clustering method? answers via model-based cluster analysis. Technical Report no. 329, Department of Statistics, University of Washington (1998)
12. Kaufman, L., Rousseeuw, P.J.: Finding Groups in data. In: An Introduction to cluster analysis. Wiley, New York (1990)
13. Peck, R., Fisher, L., Ness, J.V.: Approximate confidence intervals for the number of clusters. Journal of the American Statistical Association 84(405), 184–191 (1989)
14. Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. In: Proceedings of the Pacific Symposium on Biocomputing, pp. 6–17 (2002)
15. Tibshirani, R., Walther, G.: Cluster validation by prediction strength. Journal of Computational & Graphical Statistics 14(3), 511–528 (2005)
16. Dudoit, S., Fridlyand, J.: A prediction-based resampling method for estimating the number of clusters in a dataset. Genome Biology 3(7) (June 2002)
17. Fränti, P., Kivijärvi, J.: Randomized local search algorithm for the clustering problem. Pattern Analysis and Applications 3(4), 358–369 (2000)

Paper **P4**

©2011 ISDA. Reprinted, with permission.
Q. Zhao, M. Xu and P. Fränti,
"Expanding external validity measures for determining the number of clusters", in *Int. Conf. on Intelligent Systems Design and Applications (ISDA'11)*,
pp. 931–936, Córdobar, Spain, 2011.

Extending external validity measures for determining the number of clusters

Qinpei Zhao
School of Computing
University of Eastern Finland
Joensuu, Finland
qinpei.zhao@uef.fi

Mantao Xu
School of Electronics & Information
Shanghai Dian Ji University
Shanghai, China
mantao.xu@gmail.com

Pasi Fränti
School of Computing
University of Eastern Finland
Joensuu, Finland
pasi.franti@uef.fi

***Abstract*—External validity measures in cluster analysis evaluate how well the clustering results match to a prior knowledge about the data. However, it is always intractable to get the prior knowledge in the practical problem of unsupervised learning, such as cluster analysis. In this paper, we extend the external validity measures for both hard and soft partitions by a resampling method, where no prior information is needed. To lighten the time burden caused by the resampling method, we incorporate two approaches into the proposed method: (i) extending external validity measures for soft partitions in a computational time of $O(M^2N)$; (ii) an efficient sub-sampling method with time complexity of $O(N)$. The proposed method is then applied and reviewed in determining the number of clusters for the problem of unsupervised learning, cluster analysis. Experimental results has demonstrated the proposed method is very effective in solving the number of clusters.**

***Keywords*-external cluster validity, clustering, subsampling, image segmentation**

I. Introduction

External validity measures are preferable for evaluating the goodness of clusterings when ground truth labels are available [1]. With the ground truth consisting of class labels assigned to the patterns, the ideal clustering is selected based on how well the cluster labels produced by the algorithm match. External measures are also used to compare the similarity of two clustering results.

Rand Index [2], [3], Jaccard coefficient, Fowlkes and Mallows index [4] are typical external measures, which evaluate the clustering quality by the similarity of the pairs of data objects in different partitions. A study of 16 external measures for K-means clustering has been conducted in [5]. According to the result of this survey, we only investigated Adjusted Rand Index in the experiments.

External measures are mainly designed for hard partitions. Researchers shed light on extensions of external measures for fuzzy results. A fuzzy extension of the Rand index has been introduced [6]. Other measures such as adjusted Rand Index, the Jaccard coefficient, the Fowlkes and Mallows index have also been derived from the same formulation. However, they are as computationally expensive as $O(M^2N^2)$, where M is the number of clusters and N is the data size. Thanks Michele's pioneer solution of fuzzy clustering, the computational cost of external measures has significantly reduced to $O(M^2N)$ time [7].

In clustering, however, prior knowledge of the data is usually not available. To overcome this difficulty, Rand Index was extended to calculate a pairwise stability [8], where the pairwise stability is calculated as the variability of the clustering results by resampling the original data or multiple initializations. For example, bootstrap resampling has been utilized in evaluating the fuzzy partition stability in [9], and its fuzzy extension has been introduced in [6]. However, these methods lead to a high time complexity in general.

Since the goal of image segmentation shares the commonalities with clustering, several clustering methods have been applied in image segmentation successfully [10]. A common way to evaluate the segmentation result is supervised evaluation, in which manually segmented reference images are used as ground truth. However, external information is difficult to acquire and require human assistance. Generating a reference image is also a subjective, and time consuming task. Even given the reference information, it is not guaranteed that the reference is unique. Considering the difficulty, a framework for a similarity measure is suggested in [11], where the measure is based on an objective comparison between the results from image segmentation algorithms and several manual segmentations.

In this paper, we mainly extend the external measures by a resampling method to the case of cluster analysis that no ground truth is available. The proposed method combined both the benefits of resampling method and fast implementation of external measures in determining the number of clusters, which is applicable for both hard and soft partitions in clustering problems. With numerous clustering algorithms and varies of image types, evaluation of the segmentation result is an open question. We employed the proposed method on segmentation evaluation to prove the validity of the method.

II. External measures

Clustering aims at partitioning a set of N and d-dimensional data points $X = \{x_1, x_2, ..., x_n\}$ into M

978-1-4577-1675-1 ©2011 IEEE

clusters. The partition is defined as:

$$P = [p_{ij}]_{N\times M}; \sum_{j=1}^{M} p_{ij} = 1 \quad (1)$$

Here P is a $N \times M$ partition matrix, p_{ij} represents the probability of the i^{th} point belonging to the j^{th} cluster. In hard clustering, p_{ij} is either 0 or 1, while in soft clustering $p_{ij} \in (0, 1)$. Given two partitions P and G, external validity measures are used to measure the similarity of two clusterings by the proportion of pairs of vectors that agree by belonging either to the same cluster or to different clusters in both partitions.

A. Hard partitions

External validity measures can be computed from the contingency matrix in $O(M^2 + N)$ time for hard partitions. A contingency matrix is defined as:

$$C_{ij} = \sum_{t=1}^{N} I(P(t) = i \wedge G(t) = j) \quad (2)$$

where I is the indicator function, t is the data point, and $i, j < M$ are the group labels. The quantities a, b, c, d are defined as follows:

$$\begin{aligned} a &= \sum_{i=1}^{M}\sum_{j=1}^{M} C_{ij}^2 - N \\ b &= \sum_{j=1}^{M}(\sum_{i=1}^{M} C_{ij})^2 - \sum_{i=1}^{M}\sum_{j=1}^{M} C_{ij}^2 \\ c &= \sum_{i=1}^{M}(\sum_{j=1}^{M} C_{ij})^2 - \sum_{i=1}^{M}\sum_{j=1}^{M} C_{ij}^2 \\ d &= \sum_{i=1}^{M}\sum_{j=1}^{M}(C_{ij}\sum_{l\neq i}^{M}\sum_{s\neq j}^{M} C_{ls}) \end{aligned} \quad (3)$$

These calculate the number of points that belongs to the same cluster in P and G (a); belongs to the same cluster in P but to different in G (b); inverse of b (c); are in different groups in P and G (d). Terms a and d measure the amount of agreement of P and G, whereas terms b and c measure the amount of disagreement. The Adjusted Rand index is now derived by:

$$ARI = \frac{2 \times (a \times d - b \times c)}{(c \times c + b \times b + 2 \times a \times d + (a + d) \times (c + b))} \quad (4)$$

The definitions of Rand index, Jaccard coefficient and Fowlkes-Mallows indices are all based on Eq. 3, see [6], [7] for the exact definition.

B. Efficient extension to soft partitions

An efficient extension of external measures into soft partitions in [7] is based on an update definition of contingency matrix. In soft partitions, each point has a membership/probability value to each cluster. The calculation of contingency matrix for soft partitions is defined as:

$$C_{ij} = \sum_{t=1}^{N} (P_{ti} + G_{tj})^{\alpha} \quad (5)$$

where, t is the data point, i, j are the number of cluster. The value α is used to boost the influence of higher memberships and reduced the influence of lower memberships. We discuss its setting in Section IV. Given two soft partitions of the same data set, the contingency matrix is calculated according to Eq. 5. The calculations of a, b, c, d are the same as in Eq. 3 with the only difference of the calculation of a, which is defined as follows:

$$a = \sum_{i=1}^{M}\sum_{j=1}^{M} C_{ij}^2 - \sum_{i=1}^{M}\sum_{j=1}^{M} C_{ij} \quad (6)$$

The time complexity of the soft version is $O(M^2N)$.

III. Determining the number of clusters

For determining the number of clusters, the basic idea is to test whether the points in a data set are randomly structured or not. Resampling techniques such as bootstrapping [12], subsampling [13], cross validation [14], sampling by Monte Carlo method [15] have been utilized as a solution for this problem. They allow one to simulate the process of estimating the probability density function of a validity measures using random numbers, i.e. the resampling-based method discover the structure of the data by simulation. However, as a non-parametric method the resampling method requires high computation.

We propose a resampling-based method for determining the number of clusters in a more efficient way. The idea of this method is to estimate $index(k)$ by comparing an index value on the original data I_x with an expectation under index values on an appropriate null reference distribution of the data I_u. The estimated optimal number of clusters is the value that minimizes $index(k)$.

$$index(k) = E_b[I_u] - I_x \quad (7)$$

where E_b denotes expectation under a sample with size B from the reference uniform distribution. This idea is similar with the gap-statistic [15]. The most significant difference is that the proposed method is applied to external measures, which are working differently with internal measures. The proposed method is applicable both to hard and soft clusterings. To speed up the method, an efficient extension of soft external measures and a sub-sampling method are employed.

The proposed method is described in Algorithm 1. First, we perform a sub-sampling algorithm [16] with $O(N)$ time complexity on the original data set to reduce the computation. The parameters setting is the same as in [16]. Clustering

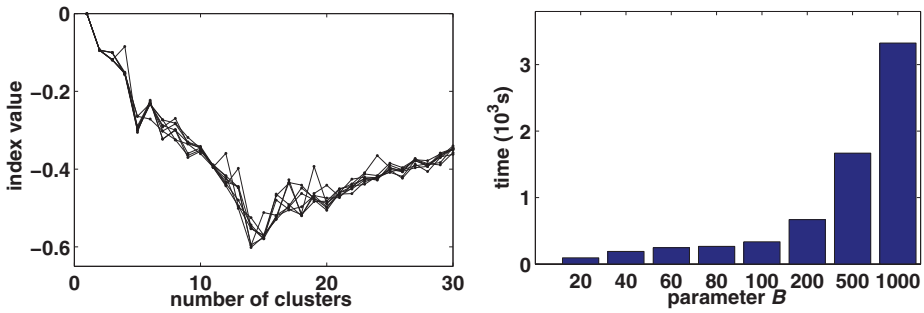


Figure 1. The settings of parameter *B* make little difference on the performance of the proposed index (left) while the processing time increases with the increment of *B* value (right).

algorithm is run on the sub-sampled data X_s, and P_x is the result. We compute an index value I_x of the defined external index between a reference partition G and P_x. Here, G is built according to the intuition about the clustering structure of the data set that the data set is not randomly collected. Let $c = \lfloor N/M \rfloor$, the reference partition $G_{N\times M}$ is generated by:

$$[G]_{ij} = \begin{cases} 1, if \quad i > (j-1)\times c \quad \& \quad i < j \times c + 1 \\ 0, otherwise \end{cases} \tag{8}$$

Next, B synthetic data sets X_b are generated in the area of the sub-sampled data (for each dimension independently) by a uniform distribution. The same clustering algorithm is run on these data sets, and let P_{ub} be the resulting clustering. We compute index values I_{ub} of the same external index between the reference partition G and P_{ub}. Index values I_{ub} are the approximation of the probability density function of the defined external index. We define $I^* = \sum_{b=1}^{B} I_{ub}/B$ as a reference of I_x.

Input: $X = \{x_1, x_2, ..., x_n\}$, K_{max}
Output: K_{opt}

```
Xs = subsampling(X) ;
for k = 2 : K_max do
    Set reference labels G = [g_ij]_{N×M} ;
    P_x = CLUSTER(Xs) ;
    I_x = ExternalIndex(P_x, G) ;
    for b = 1 : B do
        Generate reference data X_b uniformly ;
        P_ub = CLUSTER(X_b) ;
        I_ub = ExternalIndex(P_ub, G) ;
    end
    index(k) = I* − I_x ;
end
K_opt = min(index) ;
return K_opt
```

Algorithm 1: Pseudocode of the proposed method

Finally, I_x and I_{ub} are obtained for different number of clusters within the range $k \in [2, K_{max}]$, where a rule of thumb of K_{max} is $K_{max} \sim (N/2)^{1/2}$ [17]. Thus, $index(k) = I^* - I_x$ is calculated under different k, and $K_{opt} = argmin_k\{index(k)\}$.

Table I
DESCRIPTION OF THE DATA SETS, D IS DIMENSIONALITY, N IS DATA SIZE AND M IS NUMBER OF CLUSTERS.

Name	D	N	M	Generated
Touching	2	73	2	artificial
rdata3	2	300	3	artificial
S1-S4	2	5000	15	artificial
Iris	3	150	3	real
wine	13	178	3	real(Normalized)
wdbc	30	569	2	real(Normalized)
Zernike	47	2000	10	real(Normalized)
image	3	116*261	NA	real

IV. EXPERIMENTS

Experiments were performed on several real and synthetic data sets (Table I). The data set S1-S4 consists of 5000 vectors and 15 Gaussian clusters with different degree of cluster overlapping. The rdata3 is generated under Gaussian distribution with three smaller groups of data points. Touching contains two connecting clusters. The real data sets are obtained from UCI Machine Learning Repository [18]. All real data sets instead of Iris are normalized by statistical normalization. The image in (Fig. 4) in YUV color space is used for image segmentation. We test the proposed method on K-means (KM), EM and Fuzzy C-means (FCM) clustering algorithms for hard and soft clustering.

For the setting of parameter *B*, we run the proposed method with increasing *B* values. As shown in Fig. 1, the increment of *B* value increases the processing time while it brings little effect on the index value. Thus *B* in Algorithm 1 is set to 20 to get less processing time.

Spearman's rank correlation [19] is a non-parametric measure of statistical dependence between two variables. To decide the setting of α, we calculate the Spearman's rank correlation among ARI for hard partitions and ARI for soft

partitions in different α settings in Table II. As shown in the table, it has very high correlation among the ARI values on $\alpha = 10$, $\alpha = 15$ and $\alpha = 20$. However, it has very low correlation among the values when $\alpha = 1$ and the others. The correlation of the ARI values on hard partitions and soft partitions is the highest when $\alpha = 5$. Thus, we set $\alpha = 5$ in this paper.

Table II
SPEARMAN'S RANK CORRELATION AMONG ARI FOR HARD PARTITIONS AND ARI FOR SOFT PARTITIONS IN DIFFERENT α SETTINGS.

	hard	α=1	α=5	α=10	α=15	α=20
hard	1	0.65	**0.87**	0.79	0.78	0.76
α=1	0.65	1	0.64	0.57	0.53	0.51
α=5	0.87	0.64	1	0.95	0.93	0.91
α=10	0.79	0.57	0.95	1	0.99	0.98
α=15	0.78	0.53	0.93	0.99	1	1
α=20	0.76	0.51	0.91	0.98	1	1

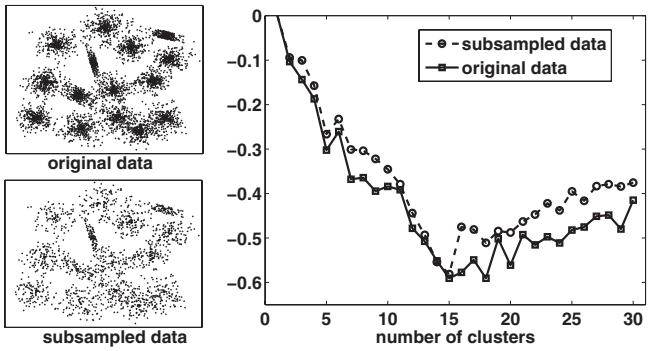


Figure 2. Original and sub-sampled data distribution (left) and the results from the proposed method on both data sets (right).

A. Sub-sampling algorithm

First, we need to verify if the sub-sampling algorithm affects the final result. As shown in Fig. 2, the original data size is sampled from 5000 to 1724 data points so that the data structure is preserved while the density is reduced. The result of the proposed method on the sub-sampled data has similar trend as that on the original data in Fig. 2. It indicates that the sub-sampled data works well in our method, although variation exists.

Table III
THE PROCESSING TIME WITHOUT AND WITH SUB-SAMPLING ON DIFFERENT PARTS IN THE PROPOSED METHOD. THE RUNNING TIME FOR THE SUB-SAMPLING PROCEDURE IS 0.09 SECONDS.

	without	with	reduced
External measures	0.13s	0.08s	38%
K-means	8.41s	1.82s	78%
EM	28.43s	8.42s	70%
FCM	42.12s	14.22s	66%

The time costs in Table III are for external measures and clustering algorithms in Algorithm 1 for data set S2 with 15 clusters. The sub-sampling method reduces time cost 38%-78% on different parts in the proposed method according to Table III. Compared to the running time for sub-sampling procedure, which is 0.09s, the reduced time is much more than that. The sub-sampling method can reduce remarkable running time in resampling method since the clustering algorithms employed in the method need to be repeated multiple times.

B. Determining the number of clusters

We tested the proposed method on the data sets in Table I. The hard partitions are resulted from K-means and the soft partitions of FCM and EM algorithm by taking the cluster with the maximal membership value.

An example on data S2 is shown in Fig. 3, where the data distribution with partitioning from FCM is displayed. The running time of the proposed method varies from different clustering algorithms, where K-means is the fastest and EM is the slowest. Thus, the running time depends highly on the choice of the clustering algorithm. Compare hard and soft clusterings, for example, hard and soft partitions from FCM and EM, the computation time have little difference as Fig. 3 indicates.

The index values of the proposed method with the increasing number of clusters are plotted, where the minimal values of the curve indicate the number of clusters. For data set S2, the proposed method reveals the structure of the data set on hard partitions from different clustering algorithms. Results from soft partitions work similar as those from hard partitions with higher variance. The main reason is that external measures on hard partitions are more robust than that of soft partitions.

To validate the proposed method, we listed the determined number of clusters for the data presented in Table I by the proposed method and two internal measures [20] in Table IV. Calinski-Harabsz (CH) index is popular as an internal measure, which is based on within and between cluster variance. Xie and Beni proposed a validity index (XB) for fuzzy clustering, which considered the data set, geometric distance measure, distance between cluster centroids and more importantly on the fuzzy partition generated by any fuzzy algorithm used. We employed K-means results for Calinski-Harabsz index and FCM results for Xie-Beni index. The bold-faced numbers in Table IV represents the correctly determined number of clusters.

For determining the number of clusters, the proposed method works well on real data sets and small Gaussian-distributed data sets. For higher overlapped data S3 and S4, the proposed method works well. In general, it has better performance on hard partitions than soft ones. Internal measures have less accurate result on real data sets, but Xie-Beni index works well on artificial data sets.

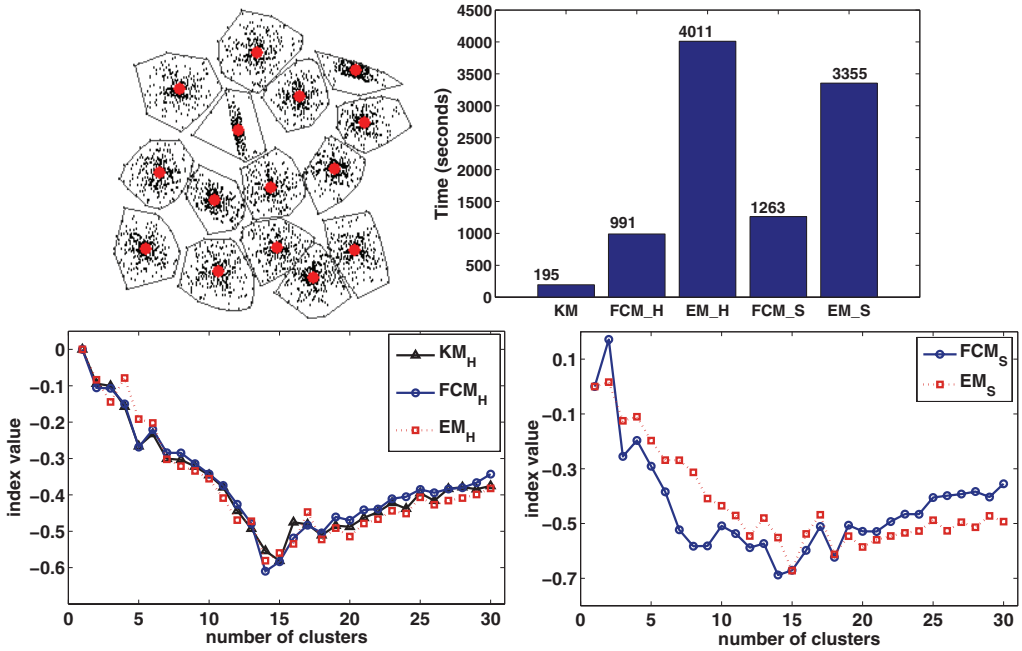


Figure 3. (a) a clustering on data set S2 from FCM; (b) a comparison on the running time of the proposed method on different soft and hard clusterings; (c) and (d) the index value of the proposed method (hard and soft clustering respectively) on the increasing number of clusters.

Table IV
THE NUMBER OF CLUSTERS DETERMINED BY THE PROPOSED METHOD FOR HARD AND SOFT PARTITIONS.

Data	KM_H	FCM_H	EM_H	FCM_S	EM_S	XB	CH
Iris	**3**	**3**	**3**	**3**	2	2	2
wine	**3**	**3**	**3**	**3**	**3**	2	2
wdbc	**2**	**2**	**2**	6	8	**2**	**2**
Zernike	**10**	5	**10**	4	**10**	2	2
image	3	3	3	5	5	2	2
rdata3	**3**	**3**	**3**	**3**	2	**3**	10
Touching	**2**	**2**	**2**	3	6	**2**	**2**
S1	17	14	17	13	18	**15**	**15**
S2	**15**	14	14	14	**15**	**15**	20
S3	**15**	**15**	**15**	**15**	**15**	4	6
S4	**15**	**15**	**15**	**15**	**15**	**15**	16

C. Unsupervised evaluation of image segmentation

For image segmentation, K-means, FCM and EM algorithms are conducted. For simplification, the segmentation results are represented by the clustering labels. We use the proposed method to determine the number of clusters for images. The proposed index and BIC (Bayesian Information Criterion) are used to evaluate the segmentation results, see Fig. 4, where the image in YUV color space is segmented into 3 clusters by different algorithms. The evaluation under different numbers of clusters (from 2 to 10) are shown in Fig. 5. There is a strong indication on three clusters by the proposed method, whereas BIC on EM algorithm has no clear suggestion.

Figure 4. An image in YUV color space and image segmentations of three clusters by KM, EM and FCM.

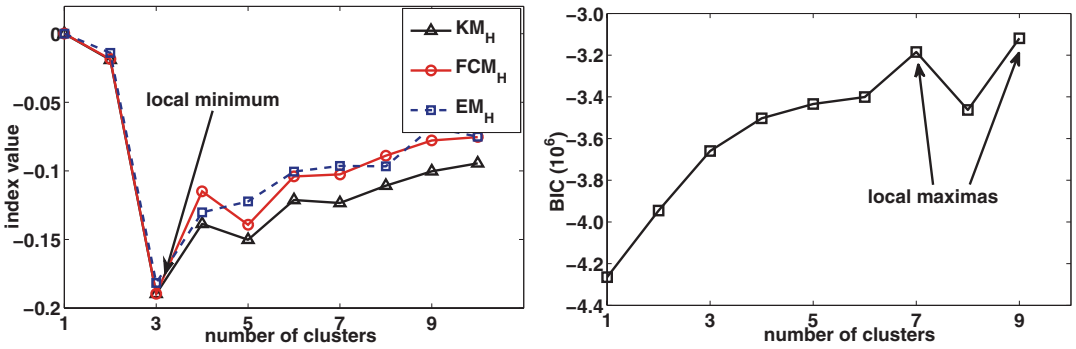


Figure 5. Evaluation results of the proposed method (left) and BIC (right) for the image.

V. Conclusion

We extended the external measures for both hard and soft partitions in clustering problems when no prior information is available. To the computational efficiency of resampling method, a state-of-art sub-sampling method is implemented and applied instead. Experimental results indicate that the proposed method are very effective in solving the number of clusters in practice, for example, the unsupervised evaluation of image segmentation. The proposed method can be envisioned as a general approach that can be applicable to other clustering algorithms and external measures.

References

[1] B. E. Dom, "An information-theoretic external cluster-validity measure," *Research Report RJ 10219, IBM*, 2001.

[2] W.M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846–850, 1971.

[3] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clustering comparison: Is a correction for chance necessary?," *ICML'09*, pp. 1073–1080, 2009.

[4] E.B. Fowlkes and C.L. Mallows, "A method for comparing two clusterings," *Journal of the American Statistical Association*, vol. 75, pp. 553–569, 1983.

[5] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," *KDD'09*, pp. 877–886, 2009.

[6] R.J.G.B. Campello, "A fuzzy extension of the rand index and other related indexes for clustering and classification assessment," *Pattern Recognition Letters*, vol. 28, no. 7, pp. 833–841, 2007.

[7] C. Michele and M. Antonio, "A fuzzy extension of some classical concordance measures and an efficient algorithm for their computation," *Int'l Conf. on Knowledge-Based Intelligent Information and Engineering Systems*, pp. 755–763, 2008.

[8] L.I. Kuncheva and D.P. Vetrov, "Evaluation of stability of k-means cluster ensembles with respect to random initialization," *IEEE TPAMI*, vol. 28, no. 11, pp. 1798–1808, 2006.

[9] M. Falasconi, A. Gutierrez, M. Pardo, G. Sberveglieri, and S. Marco, "A stability based validity method for fuzzy clustering," *Pattern Recognition*, vol. 43, no. 4, pp. 1292–1305, 2010.

[10] C. Carson, S. Belongie, H. Greenspan, and J.Malik, "Blobworld: Image segmentation using expectation-maximization and its application to image querying," *IEEE TPAMI*, vol. 24, no. 8, pp. 1026–1038, 2002.

[11] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," *IEEE TPAMI*, vol. 29, pp. 929–944, 2007.

[12] M.H.C. Law and A.K. Jain, "Cluster validity by bootstrapping partitions," *Technical Report MSU-CSE-03-5, Dept. of Computer Science and Engineering, MSU, Michigan, USA*, 2003.

[13] E. Levine and E. Domany, "Resampling method for unsupervised estimation of cluster validity," *Neural Computation*, vol. 13, pp. 129–137, 2001.

[14] W.M. Abd-Elhafiez A. Farag M. El-Melegy, E.A. Zanaty, "on cluster validity indexes in fuzzy and hard clustering algorithms for image segmentation," *ICIP'07*, pp. 5–8, 2007.

[15] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society*, vol. 63, pp. 411–423, 2001.

[16] T. Hasan, Y. Lei, A. Chandrasekaran, and J.H.L. Hansen, "A novel feature sub-sampling method for efficient universal background model training in speaker verification," *ICASSP'10*, pp. 4494–4497, 2010.

[17] K. V. Mardia, J. T. Kent, and J. M. Bibby, "Multivariate analysis," *Academic Press*, 1979.

[18] A. Asuncion and D.J. Newman, "UCI machine learning repository," http://archive.ics.uci.edu/ml/, 2007.

[19] Jerome L. Myers and Arnold D. Well, "Research design and statistical analysis (second edition ed.)," *Lawrence Erlbaum*, p. 508, 2003.

[20] Q. Zhao, M. Xu, and P. Fränti, "Sum-of-square based cluster validity index and significance analysis," *ICANNGA*, pp. 313–322, 2009.

Paper **P5**

Q. Zhao and P. Fränti,
"Centroid Ratio for Pairwise Random Swap Clustering Algorithm",
manuscript

Centroid Ratio for Pairwise Random Swap Clustering Algorithm

Qinpei Zhao, Pasi Fränti, *Senior Member, IEEE*

Abstract—Clustering based on swap strategy is to improve prototype-based clustering. The swap strategy can be either random or deterministic. Design of the swap strategy is critical because wrong design can make the algorithm either stuck into a local optima or in low efficiency. In this work, a new swap strategy is proposed in a novel *Pairwise Random Swap* clustering algorithm. The swap strategy is based on a cluster-level evaluation measure, so called *Centroid Ratio*. The measure is used to compare two clusterings based on centroids, whereas other measures are based on partitions, or both partitions and centroids. It has low time complexity and is applicable for detecting unstable or incorrectly located centroids. The centroid ratio is shown to highly correlate to external indices and mean sqaure error (MSE). The pairwise random swap clustering algorithm employing centroid ratio as the swap strategy has low time complexity and comparable MSE values. An empirical study on synthetic and real data indicates that the proposed clustering algorithm works more efficiently than *Random Swap*, *Deterministic Random Swap*, *Repeated k-means* and *k-means++*.

Index Terms—data clustering, random /deterministic swap, clustering evaluation, k-means

1 Introduction

Prototype-based clustering is a typical clustering method for finding a sequence of prototypes that best fit the data with unknown structure. For example, a single prototype (centroid) is used to represent a cluster in *k-means* [1], which has been widely applied for data grouping in real applications not only because it has low computation and memory space requirements but it also achieves good result in most cases. However, it is known to be sensitive to initialization.

A common way to address the initialization problem is to run k-means multiple times with a different set of randomly chosen initial parameters [2] and to choose the best solution as a result. We call this variant *repeated k-means* (RKM). For different data sets, proper number of repetitions for RKM is an empirical choice. Several other methods have been developed, which are based on stochastic global optimization such as *simulated annealing* [3] and *genetic algorithms* [4]. These methods have not gained wide acceptance because of their high time complexity. A *global k-means* algorithm (GKM) [5] is an incremental approach that dynamically adds one cluster center at a time through a deterministic global search procedure. The search procedure consists of N (data size) executions of the k-means algorithm from suitable initial positions. *K-means++* [6] chooses initial values (seeds) for k-means, and improves both the speed and accuracy of k-means. It is $\Theta(\log M)$-competitive with the optimal clustering [6], i.e. $E[\phi] \leq 8(\log M + 2)\phi_{OPT}$ where ϕ indicates the cost function and M represents the number of clusters.

Swap-based clustering algorithm [7] is a local search heuristic to find optimal centroids. In each iteration, a swap strategy is employed to look for a pair of centroids, of which one is to be removed, and the other is inserted to lead an improved solution. If better prototypes are found, the swap is made and the procedure is repeatedly performed after a fine-tuning step by k-means. Swap-based clustering is simple to implement and has good quality of results independent on the initialization. The swap strategy could be either random or deterministic. A *random swap* means choosing a cluster to be removed and a location to be inserted randomly. The random swap algorithm has linear dependency on the number of data vectors (N) but quadratic on the number of clusters (M), and inverse dependency on the dimensionality (D) according to [8].

A problem concerning *deterministic swap* is to design a criteria for selecting a prototype to be removed and a location to be inserted. The balance between the removal and addition steps is a key factor for a good performance. The candidate to be removed could be the one with the smallest size or variance. The new location of the prototype can be chosen by considering the locations of all possible data vectors as in *J-means* [9]. Both J-means and random swap achieve good clustering result but they do not always work efficiently, since they generate a large number of candidate solutions during swapping. Therefore, deterministic swap-based methods have been considered by selecting a centroid to be swapped as the one increasing the cost function value least [10], [11], or by adding one cluster [5] and merging two existing

• *Q. Zhao and P. Fränti are with the School of Computing, University of Eastern Finland, FI 80110.*
E-mail: qinpei.zhao@uef.fi

clusters [12] following the spirit of agglomerative clustering. The deterministic swap strategy is usually based on clustering structures such as prototypes and partitions. Even though the correct clustering can be obtained by much fewer swaps compared to J-means and random swap, the time complexity for each deterministic swap is at least $O(MN)$ where N is the data size, M is the number of clusters. Therefore, the lower time complexity of a single swap reduces the overall efficiency of a deterministic swap-based clustering.

In this paper, we first propose a cluster validity index called *centroid ratio*, which can be used to compare two clusterings and find unstable and incorrectly located centroids in the clusterings. The index has a time complexity of $O(M^2)$ because it relies on centroids only. Then we design a pairwise random swap clustering algorithm, which employs centroid ratio as a swap strategy.

In clustering evaluation, partitions at point level are often used for evaluating clusterings by external indices [13] such as *Rand index, Jaccard coefficient*. Since these evaluation measures are at point level, they provide high accuracy but their time complexity are related to both $O(M)$ and $O(N)$, typically $O(MN)$. There is very little work done on clustering evaluation based on centroids only. Centroids represent a global structure of prototypes, and by using them, the centroid ratio can provide a global evaluation on the clustering. Utilizing only centroids in evaluation reduces the time complexity to $O(M^2)$. In this paper, we show that the proposed centroid ratio has high correlation with other evaluation measures.

As the centroid ratio can find incorrectly located centroids in two clusterings, we utilize this property and propose a novel deterministic swap based clustering called *Pairwise Random Swap* (PRS) clustering algorithm. The centroid ratio is used to select a cluster to be swapped and it is also used as a stopping criterion in the algorithm. We compare the PRS algorithm to other algorithms such as random swap clustering (RS), deterministic random swap clustering (DRS), repeated k-means (RKM) and k-means++ (KM++) on a variety of data sets in Section 4. The experimental result indicates that the proposed algorithm requires 26% to 96% less processing time than the second fastest algorithm RS and avoids the local optimality problem better than the other non-random swap strategies.

2 Related Work

2.1 k-means

Given $X = \{x_1, x_2, ..., x_N\}$ as a set of N points in a d-dimensional Euclidean space to be clustered, we define C and P as a specific partition of these points into M clusters, where $C = \{c_1, c_2, ..., c_M\}$ presents the centroids and $P = \{p_1, p_2, ..., p_N\}$ the point level partitions. A cost function is used to evaluate the quality of the clustering. There is no universal function for all clustering problems, and the choice of the function depends on the application. We consider the clustering as an optimization problem, and *mean squared error* (MSE) is the most common cost function, calculated as:

$$f = \frac{1}{N}\sum_{i=1}^{N} \|x_i - C\|^2 \quad (1)$$

K-means (Algorithm 1) is the most famous clustering algorithm, which aims to partition N objects into M clusters so that each object belongs to the cluster with the minimum Euclidean distance to the cluster centroid.

Input: X, M
Output: C, P, MSE
$c_j = x_i | i = random(1, N)$, $0 \le j \le M$;
while *! convergence* **do**
  $p_i \leftarrow \underset{1 \le j \le M}{\operatorname{argmin}} \|x_i - c_j\|^2$, $\forall i \in [1, N]$;
  $c_j \leftarrow (\sum_{p_i=j} x_i)/(\sum_{p_i=j} 1)$;
  $MSE = \frac{1}{N}\sum_{i=1}^{N} \|x_i - C\|^2$;
end
return C, P, MSE ;

Algorithm 1: *k-means* algorithm

It is known that k-means suffers from the initialization. With different initial solutions, k-means converges to different local minima, which makes the final result unstable. Previous work on improving the clustering results based on standard k-means employs different strategies [2], [3], [4], [5], [6], [7], [14], of which the swap-based approach is simple but effective.

2.2 Swap-based clustering

In swap-based clustering, centroids are perturbed by a certain strategy in order not to get stuck into a local minima. A swap is accepted if it improves the clustering quality. This trial-and-error approach is simple to implement and very effective in practice.

Random Swap algorithm (RS), originally called *Randomized Local Search* [7], is based on randomization where a randomly selected centroid is swapped to another randomly selected location. After that, a local repartition is performed and the clustering is fine-tuned by two k-means iterations. Pseudocode of the random swap algorithm is described in Algorithm 2.

To ensure a good clustering quality, the number of iterations T for random swap should be set large enough to find successful swaps. For a more accurate analysis, with a given confidence level q, the number of iterations T for a successful swap has an estimated

bound [8] as follows:

$$T = \Theta\left(-\frac{M^2}{\alpha^2} \cdot \ln q\right) \quad (2)$$

where M is the number of clusters, and α is the number of neighbor clusters.

Input: X, M
Output: C, P, MSE
$C \leftarrow InitializeCentroids(X)$;
$P \leftarrow OptimalPartition(X, C)$;
for T *times* **do**
  $C^{new} \leftarrow RandomSwap(C)$;
  $P^{new} \leftarrow LocalRepartition(P, C^{new})$;
  $KmeansIteration(P^{new}, C^{new})$;
  if $f(P^{new}, C^{new}) < f(P, C)$ **then**
    $(P, C) \leftarrow P^{new}, C^{new}$;
  end
end
$MSE = \frac{1}{N}\sum_{i=1}^{N} \|x_i - C\|^2$;
return C, P, MSE ;

Algorithm 2: Pseudocode of *Random Swap* algorithm

Deterministic swap aims at finding good swaps by a systematic analysis rather than in a trial-and-error manner. In general, the clustering can be found in a few swaps only if the algorithm knows the centroid that should be swapped and the location where it should be relocated.

Several heuristic criteria have been considered for selection of the centroids to be swapped, but simple criteria such as selecting the clusters with the smallest size or variance does not work very well in practice. Other approaches remove one cluster [11], or merge two existing clusters as in agglomerative clustering [12]. The deterministic removal takes N distance calculations for each of the M clusters. Thus, the overall time complexity of the deterministic removal step becomes $O(MN)$.

The replacement location of the swapped centroid can be chosen by considering locations of all possible data points, however it would be very inefficient. In order to find the correct location, the task can be divided into two parts: select an existing cluster and select a location within this cluster. One heuristic selection is to choose the cluster that has the largest distortion (Eq. 1). The exact location within the cluster can be chosen considering the following heuristics: 1) current centroid of the cluster with small movement; 2) furthest data point; 3) middle point of the current centroid and furthest data point; 4) random.

With the random and deterministic swap strategies, an analysis combining the deterministic heuristic with random swap was conducted in [15].

3 METHODOLOGY

3.1 Centroid Ratio

Design of swap criterion is based on three elements: data set, point level partitions and centroids. Mean square error (MSE) is a conventional criterion for evaluating clustering, which is calculated by these three elements. External indices [13], however, utilize only partitions by comparing the given clustering against ground truth. The ground truth is usually built by using human assessors or output of another clustering. External indices count pairs of points on the agreement or disagreement of two partitions. The evaluation measures are well studied in literature [13], [16], [17].

The criterion such as MSE uses quantities and features inherent in the dataset, which gives a global level of evaluation. Since it relates to points and clusters, time complexity is at least $O(MN)$. The partition-based criteria are based on pointwise evaluation of two partitions, which give the time complexity of $O(N^2)$ usually. The time complexity of point-pair measures can be reduced to $O(N + M^2)$ [18] by a contingency matrix.

There is little research on cluster level evaluation measure, which is based on centroids only. As an important structure of clustering, centroid reveals the allocation of clusters. Two clusterings $\{X, P_1, C_1\}$ and $\{X, P_2, C_2\}$ from k-means are shown in Fig. 1, where centroids and partitions have high correlation with each other. The partition shows little difference (left) at the border of the clusters while centroids also display little difference on the location. For incorrectly located centroids (right), the partitions differ greatly. The evaluation of the clustering can be performed on either partition P or centroids C. Motivated by this, we introduce a cluster level criterion in this section.

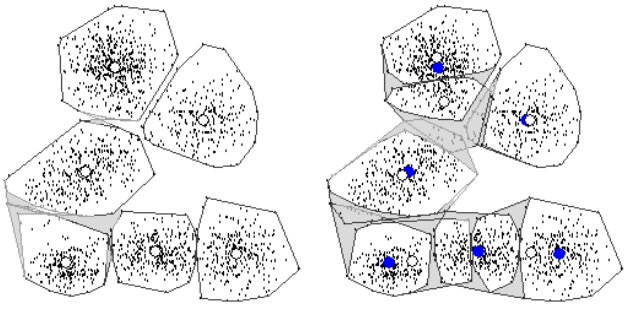


Fig. 1. Clusterings from k-means, showing the connection between centroids and partitions.

Let $C_1 = \{c_{11}, c_{12}, ..., c_{1M}\}$ and $C_2 = \{c_{21}, c_{22}, ..., c_{2M}\}$ be the centroids of two clusterings C_1 and C_2 respectively and $|C_1| = |C_2|$.

A pairing problem between two sets of centroids can be represented by a bipartite graph in which the vertex classes are the centroids in C_1 and C_2

separately, and centroids in C_1 are joined by edges to centroids in C_2.

Definition The *Nearest Pairing* of two sets of centroids (C_1 and C_2) can be stated in graph-theoretic terms as the minimum matching of a given bipartite graph where nodes correspond to the centroids, edges connect centroids from different clusterings, and edge cost stands for the centroid distance.

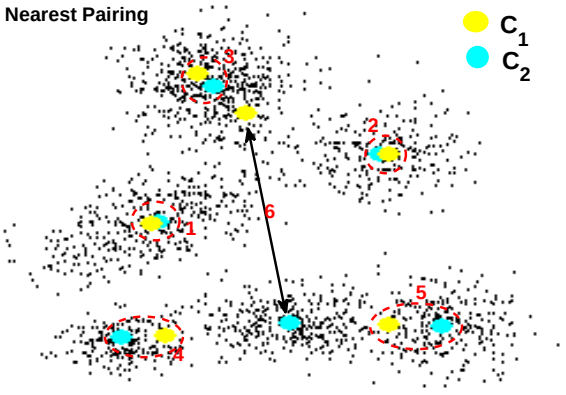


Fig. 2. Nearest pairing of two clusterings C_1 and C_2.

Definition *Pair Ratio* for centroid i, $PR(i)$, is the degree of matching between centroid i from C_1 and C_2 after nearest pairing.

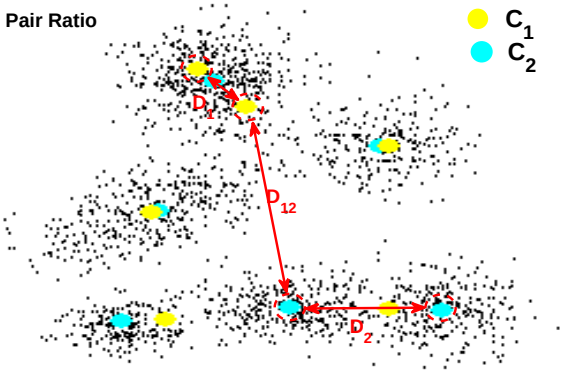


Fig. 3. Calculate *Pair Ratio* for one pair of centroids.

The minimum matching in nearest pairing is solved in a greedy way. For each i,j, where $1 < i < M$, $1 < j < M$, we consider they are paired if c_{2j} is the closest centroid to c_{1i} out of $\{c_{21}, c_{22}, ..., c_{2M}\}$. We thus iterate M times the operations:

$$\begin{aligned} \{i,j\} &= \underset{c_{1i}\in C_1, c_{2j}\in C_2}{\operatorname{argmin}} \|c_{1i} - c_{2j}\|^2 \\ C_1 &\leftarrow C_1 \backslash \{c_{1i}\} \\ C_2 &\leftarrow C_2 \backslash \{c_{2j}\} \end{aligned} \tag{3}$$

For paired centroids $c_{1i} \in C_1$ and $c_{2j} \in C_2$, we define the distances:

$$\begin{aligned} D_1(i) &= \min_{c_{1s}\in C_1} \|c_{1i} - c_{1s}\|^2 \\ D_2(i) &= \min_{c_{2s}\in C_2} \|c_{2j} - c_{2s}\|^2 \\ D_{12}(i) &= \|c_{1i} - c_{2j}\|^2 \end{aligned} \tag{4}$$

The value of D_{12} is the distance of the matched centroids in two clustering results C_1 and C_2. D_1 is the nearest distance of two centroids in the same set of centroids C_1 and similarly, D_2 is the nearest distance in C_2. The centroids in two clustering sets are strictly matched when $D_{12} = 0$. We consider centroid i is stable or correctly located when $D_{12} \leq D_1$ and $D_{12} \leq D_2$. Thus, the *Pair Ratio* for a centroid i of clustering C_1 with respect to C_2 (see Fig. 3) is defined by:

$$PR(i) = \frac{D_{12}(i)}{D_1(i)} \times \frac{D_{12}(i)}{D_2(i)} \tag{5}$$

A centroid i is considered as stable or correctly located when $PR(i) \leq 1$. For unstable and incorrectly located centroids, $PR(i) > 1$.

Definition The similarity S between two clusterings C_1 and C_2 is:

$$S(C_1, C_2) = 1 - \sum_{i=1}^{M} \gamma_i / M \tag{6}$$

where, $\gamma_i = \begin{cases} 1 & \text{if } PR(i) > 1 \\ 0 & \text{otherwise} \end{cases}$, and here S value is in the range of $[0, 1]$, where 1 indicates a completely match of two clusterings while 0 indicates a complete mismatch.

Definition S_{id} is a set of incorrectly located centroids and $S_{id} = \{i | PR(i) > 1\}$. Given T sets of clustering results, the degree of stability of centroid i is defined as:

$$stability(i) = \frac{\sum_{t=1}^{T} \sum_{s=1}^{T} (1 - \gamma_i)_{\{C_t, C_s\}}}{T^2} \tag{7}$$

If the stability is 1, the centroid i is correctly located among T sets of clustering results.

Definition *Centroid Ratio* is defined as a combination of Pair Ratio (PR) and the similarity S, where PR finds incorrectly located centroids and the S value indicates the similarity of two clusterings.

Intuitively, the clustering is stable with respect to a data set in case a global optimal exists [19], and the difference among clustering results is little. The stability of cost function in clustering algorithm implies the stability of actual centroids of clusters. We show in the following lemma that the similarity S is highly correlated to the difference of MSE in Euclidean space.

Lemma 3.1: Assume that the data x in each cluster lies in an Euclidean ball in $\mathbb{R}^d$, i.e. $x \subset \mathbb{R}^d$. Let P be the probability measure function and $f_{C_1} =$

$\frac{1}{N}\sum_{i=1}^{M}\sum_{x_j \in c_{1i}} \|x_j - c_{1i}\|^2$. For $\epsilon > 0$, suppose that the difference between two clustering results

$$\|f_{C_1} - f_{C_2}\|_{L_1(P)} \leq \epsilon \tag{8}$$

where $L_1(P)$ stands for L_1 norm of P. In case of stable clusterings, the difference between two clusterings is little ($\epsilon \to 0$) and $\|f_{C_1} - f_{C_2}\| \xrightarrow{P} 0$. Then we can prove

$$S(C_1, C_2) \xrightarrow{P} 1 \tag{9}$$

Proof: Suppose the distance $\|c_{1i} - c_{2j}\| = R$, and consider $B(c_{1i}, R/2)$ be a ball of radius $R/2$, centered at c_{1i}. So for $x \in B(c_{1i}, R/2)$, we have $\|x - c_{1i}\|^2 \leq \|x - c_{2j}\|^2$, and for any x, $\|x - c_{1i}\|^2 \leq f_{C_1}$. From Eq. 8,

$$\begin{aligned}
&\|f_{C_1} - f_{C_2}\|_{L_1(P)} = \int |f_{C_1}(x) - f_{C_2}(x)|\, dP(x) \\
&\text{for } x \in B(c_{1i}, R/2), \\
&\geq \int_{B(c_{1i},R/2)} |f_{C_1}(x) - f_{C_2}(x)|\, dP(x) \\
&= \int_{B(c_{1i},R/2)} \left| f_{C_1}(x) - \|x - c_{2j}\|^2 \right| dP(x) \\
&= \int_{B(c_{1i},R/2)} \|x - c_{2j}\|^2 - f_{C_1}(x) dP(x) \\
&\geq \int_{B(c_{1i},R/2)} (\|x - c_{2j}\|^2 - \|x - c_{1i}\|^2) dP(x) \qquad (10) \\
&\text{since } \|x - c_{2j}\|^2 \leq (R/2)^2, \\
&\geq \int_{B(c_{1i},R/2)} (R/2)^2 - \|x - c_{1i}\|^2\, dP(x) \\
&\geq a_1 \int_0^{R/2} ((R/2)^2 - r^2) r^{d-1} dr \\
&= a_1 (\frac{1}{2})^{d+2} [\frac{1}{d} - \frac{1}{d+2}] R^{d+2} \\
&= a R^{d+2} \leq \epsilon
\end{aligned}$$

where, a is a constant. Thus, we get $R \leq (\epsilon/a)^{1/(d+2)}$. Suppose the clustering is stable, i.e. $\epsilon \to 0$, we get $R \to 0$, which implies that the difference between two sets of centroids for certain clustering approaches to 0, i.e. $\|c_{1i} - c_{2j}\| \xrightarrow{P} 0$. For M centroids of C_1 and C_2, $S(C_1, C_2) \xrightarrow{P} 1$. □

The above lemma indicates the relationship between the proposed similarity and the difference of MSE values of two clusterings C_1 and C_2. MSE value reflects a global view, but there is no way to track detail information of each point through it. External indices such as Rand index can compare two clusterings pointwisely, but they can not give information directly on clusters. The proposed centroid ratio can reveal the information at a cluster level, which is able to give a global evaluation and detect unstably or incorrectly located centroids. In Section 3.2, we introduce a clustering algorithm employing centroid ratio, which can improve the local optimal problem of k-means in an efficient way.

3.2 Pairwise Random Swap algorithm

The pairwise random swap algorithm (PRS) takes a given data set X and the number of clusters M as its input. It starts by generating two centroids sets (C_1, C_2) and MSE values (MSE_1, MSE_2) from conventional k-means as described in Algorithm 3. Then, we calculate the pair ratio value $PR(i)$ for each paired centroids to get a set of incorrectly located centroids S_{id} and the similarity value $S(C_1, C_2)$ according to Eq. 6. We perform *Swap* function (Algorithm 4) to get an improved solution, in which we randomly swap the detected centroids c_{1j} and c_{2j} in C_1 and C_2 ($j \in S_{id}$) and fine-tune the result by k-means. The algorithm stops when the similarity of two centroid sets S is 1, which indicates that the centroids of two clusterings are matched. The final solution of the PRS algorithm is the centroid set that has lower MSE value, i.e. $\min(MSE_1, MSE_2)$.

Input: X, M
Output: C, MSE
Two initializations: I_1, I_2;
(C_1, MSE_1) = *k-means*(X, I_1, M);
(C_2, MSE_2) = *k-means* (X, I_2, M);
Calculate $S_{id} = i|PR(i) > 1$ and $S(C_1, C_2)$;
while $S \neq 1$ **do**
  ($C_1^{'}, C_2^{'}, MSE_1^{'}, MSE_2^{'}$) = *Swap*($X$, M, C_1, C_2, MSE_1, MSE_2, S_{id});
  $MSE_1 = MSE_1^{'}$; $MSE_2 = MSE_2^{'}$;
  $C_1 = C_1^{'}$; $C_2 = C_2^{'}$;
  Calculate $S_{id} = \{i|PR(i) > 1\}$ and $S(C_1, C_2)$;
end
return $\min(MSE_1, MSE_2)$ and corresponding C_1 or C_2;

Algorithm 3: *Pairwise Random Swap* clustering algorithm

Input: X, m, C_1, C_2, MSE_1, MSE_2, S_{id}
Output: $C_{r1}^{'}, C_{r2}^{'}$ and $MSE_{r1}^{'}, MSE_{r2}^{'}$
$MSE_{r1}^{'}$ = MSE_1 + 1;
while $MSE_{r1}^{'} > MSE_1$ **do**
  $C_{r1} \leftarrow$ random swap S_{id} on C_1;
  ($C_{r1}^{'}, MSE_{r1}^{'}$) = *k-means*(X, C_{r1}, m);
end
$MSE_{r2}^{'}$ = MSE_2 + 1;
while $MSE_{r2}^{'} > MSE_2$ **do**
  $C_{r2} \leftarrow$ random swap S_{id} on C_2;
  ($C_{r2}^{'}, MSE_{r2}^{'}$) = *k-means*(X, C_{r2}, m);
end
return $C_{r1}^{'}, C_{r2}^{'}$ and $MSE_{r1}^{'}, MSE_{r2}^{'}$;

Algorithm 4: Function of *Swap*

On occasion, the initial centroid sets C_1 and C_2 are completely matching but the partition is local optimal, i.e. $S(C_1, C_2) = 1$ and $S_{id} \in \emptyset$ at the beginning, in

TABLE 1
Summary of time complexities on one iteration of deterministic swap. *RD* represents random removal and deterministic addition; *DR*, deterministic removal and random addition and *DD*, deterministic removal and deterministic addition. *PRS* is for the proposed PRS algorithm.

	RD	DR	DD	PRS
Removal	$O(1)$	$O(MN)$	$O(MN)$	$O(M^2)$
Addition	$O(N)$	$O(1)$	$O(N)$	$O(1)$
fine-tuning	$O(sN)$	$O(sN)$	$O(sN)$	$O(sN)$
Total	$O(sN+N)$	$O(sN+MN)$	$O(sN+MN)$	$O(sN+M^2)$

which case the PRS algorithm performs a random swap on the centroids.

The proposed algorithm is a type of deterministic swap clustering (DR) since centroids to be swapped are chosen by the centroid ratio and the allocated position is random. The time complexity of the removal step is $O(M^2)$ and $O(1)$ for addition step. Although the swap heuristic is capable of moving out of a local minimum, it may take a long time to move near to a local minimum. Thus, it is profitable to use k-means for fine-tuning after the swap heuristic [20]. A note for the PRS algorithm is that k-means can be substituted by other prototype-based clustering algorithms.

3.3 Efficiency Analysis

The efficiency of a swap-based clustering algorithm depends on two issues: how many iterations (swaps) are needed and how much time each iteration consumes.

In random swap, the swap step is completely random so it needs a large number of iterations to provide a good quality of result. It takes $O(sN)$ (s is the number of neighboring clusters on average) at least for each iteration with a fast variant of k-means for fine-tuning [21]. The main bottleneck of random swap is that the number of iterations T has quadratic dependency on the number of clusters M (Eq. 2), which increases the overall time complexity.

The selection criterion for swapping in *DR* is to find clusters that increase cost function (MSE) least when they are swapped. In *DD*, the centroid to be removed is chosen by calculating removal cost, and the addition is made within the cluster of the highest distortion. In this case, the number of iterations is limited because the algorithm will stop whenever there is no improvement. However, the time required for each iteration is high. It takes $O(MN)$ for finding the minimum removal cost, $O(N)$ for the addition cost and $O(sN)$ for the local partition and fine-tuning, so the total time complexity of one iteration in *DD* is $O(sN+MN)$. Time complexities for variants of deterministic swap are summarized in Table 1. As shown in the table, time complexities of the existing deterministic strategies are either related to $O(N)$ or $O(MN)$. The time complexity of the swap strategies is the only difference in the total time complexity of variants.

In the proposed method, the algorithm needs $O(M^2)$ to find incorrectly located centroids and $O(1)$ for addition. The main computation comes from repartitioning and fine-tuning by the k-means iterations, which takes $O(sN)$. The total time complexity is $O(k_2(k_1sN+M^2))$, where k_1 is the number of iterations for k-means and k_2 is the repeated times of centroid ratio step. It is shown by experiment that the selection of k_1 affects very little on the final result and the algorithm can stop in less than M runs of centroid ratio, i.e. $k_2 \leq M$.

To sum up, random swap needs a large number of iterations to provide a good quality of clustering. The deterministic swap needs less number of iterations, but it takes more time for each iteration than random swap. For the variants of deterministic swap, the main computation coming from the local partitioning is the same. However, the time complexity of deterministic strategies differs and the number of iterations depends on the swap strategies.

The time complexity for global k-means is $O(TM^2N^2)$ with incrementally adding one cluster at a time through a deterministic global search, where T is an average k-means iterations. The kmeans++ has an additional procedure for choosing initial cluster centers, which adds $O(MN)$ to the time complexity of the standard k-means.

4 Experiments

We tested the algorithms using both synthetic and real data sets from various sources as summarized in Table 2.

TABLE 2
Attributes of the data sets used in our experiments. For the data sets where the number of clusters is unknown, the model sizes used in the experiments are shown in parenthesis.

Name	Dimensionality	Data Size	#Clusters
Synthetic data sets			
S1-S4 [11]	2	5000	15
Aggregation [22]	2	788	7
R15 [23]	2	600	15
BIRCH1-BIRCH2 [24]	2	100000	100
Real data sets			
CM [25]	9	68040	NA(20)
CT [25]	16	68040	NA(20)

The synthetic data sets are two dimensional and contain a known number of clusters, which are easy from visualization point of view. The ground-truth labels are known for S1 to S4[1], which have gradually more overlapping clusters. In S1 the overlap is the smallest whereas in S4 the overlap is the greatest. BIRCH sets [24] are large data sets with 100 clusters among 100,000 data points. BIRCH1 contains clusters in regular grid structures, BIRCH2 has clusters at a Sine curve. R15 [23] is generated as 15 similar 2-D Gaussian distributions that are positioned in rings. Data Aggregation (A7) [22] consists of seven perceptually distinct groups of points, where there are non-Gaussian clusters. The distribution of two dimensional data sets are shown in Fig. 4.

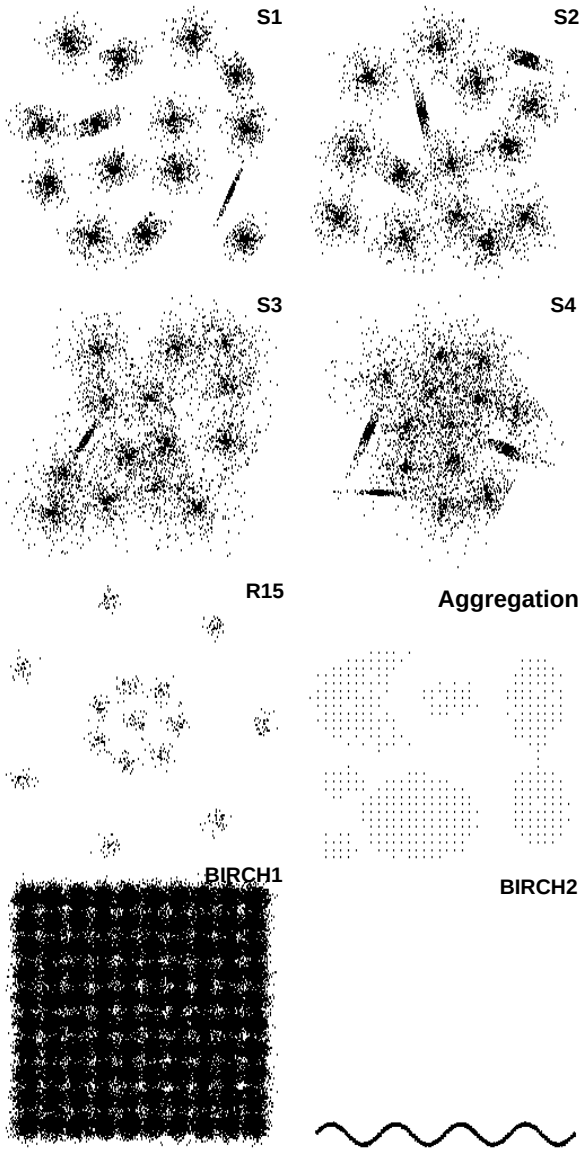


Fig. 4. Visualization of two-dimensional data sets.

The real data sets are the color moments (CM) and co-occurrence texture (CT) data sets from [26]. It is unknown whether the data is clustered. We selected the number of components of CM and CT as 20 in the experiment, because the number of clusters of these two data sets are unknown and the problem of determining the number of clusters is out of our scope.

4.1 Centroid Ratio Validity

We study the validity of the proposed centroid ratio in this Section. To compare with other clustering evaluation measures, we define consistency in terms of similarity between their rankings on a number of clustering results. The compared measures include Rand index (RI), Adjusted Rand index (ARI), Jaccard coefficient (Jac), Fowlkes and Mallows index (FM) and ΔMSE. The similarity is based on the Spearman's rank correlation, which is a non-parametric measure of statistical dependence between two variables. The clustering results are obtained from 50 runs of standard k-means clustering on data set S2 with 15 clusters until convergence. The ground-truth labels are known for the data set S2.

TABLE 3
Spearman's rank correlation on different clustering validity measures.

	RI	ARI	Jac	FM	-ΔMSE	CR
RI	1	1	1	1	0.90	0.96
ARI	1	1	1	1	0.90	0.96
Jac	1	1	1	1	0.90	0.96
FM	1	1	1	1	0.90	0.96
-ΔMSE	0.90	0.90	0.90	0.90	1	0.94
CR	0.96	0.96	0.96	0.96	0.94	1

The correlation in Table 3 indicates that external measures have very high correlation with each other. The proposed centroid ratio has higher correlation with external measures than ΔMSE. By the high correlation of centroid ratio and other measures, we conclude that the centroid ratio is valid for clustering evaluation.

According to the definition for stability degree of centroids in Section 3.1, we tested the stability of centroids in standard k-means and Global k-means (GKM) [5] separately. We repeated $T = 10$ runs and pairwisely calculated the degree of stability from ten clusterings by Eq. 7. The degree of stability for each centroid in k-means and GKM is shown in Fig. 5, centroids $2, 4, 5, 7, 9, 10, 14$ are not stable from k-means, while it is stable in GKM for all centroids. The degree of the stability is reflected on each centroid, for example, centroid 7 is the most unstable centroid in k-means. According to the stability of centroids from different algorithms, we can then conclude the stability of the algorithms.

1. http://cs.joensuu.fi/sipu/datasets/

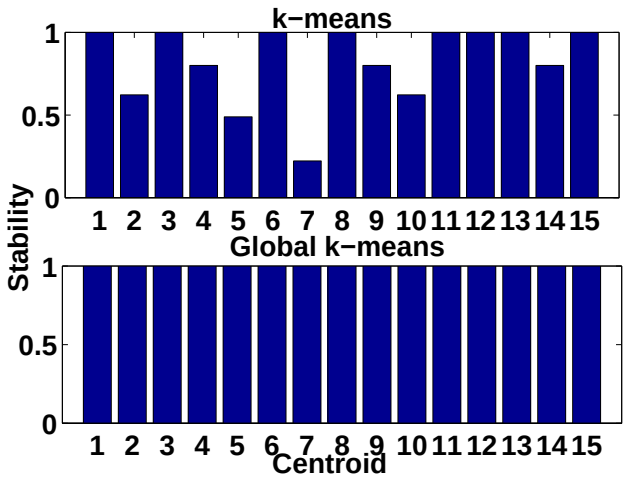


Fig. 5. The stability of each centroid and finding unstable centroids by centroid ratio.

4.2 Validity of Pairwise Random Swap algorithm

We compare PRS with other variants of k-means, including repeated k-means (RKM) and kmeans++ (KM++) [6]. We also compare it with the Random Swap algorithm (RS) and Deterministic Random Swap algorithm (DRS). The clustering algorithms[2] are implemented in C and tested under the same environment.

Swapping iterations are needed in RS and DRS and repetitions are needed for RKM and KM++ to guarantee good performance. We summarized the parameter setting in the experiments in Table 4. The number of swapping iterations in RS can refer to Eq. 2. For RKM and KM++, the number of repetitions is selected experimentally. All algorithms employ k-means, the number of iterations of k-means in RS and DRS is set two and running until converge in RKM and KM++.

TABLE 4
Parameter settings for algorithms: RS, DRS, RKM and KM++. The numbers in the table represent the number of iterations for swaps and repetitions in the experiments.

Data	RS / DRS	RKM / KM++
S1-S4	130	130
R15	130	130
A7	60	60
BIRCH1	1400	300
BIRCH2	10000	300
CM	2000	300
CT	2000	300

We study the relationship between the number of k-means iterations and clustering result by MSE values and processing time in Fig. 6. We repeated PRS 50 times on each number of iterations for k-means. The differences of MSE values among the runs with different numbers of k-means iterations are less than 0.0007%, which is negligible. The processing time has variance on each run with different numbers of iterations. However, a larger number of k-means iterations does not necessarily lead to a better result and higher processing time according to Fig. 6. Thus, we set k-means iterations in PRS to 10.

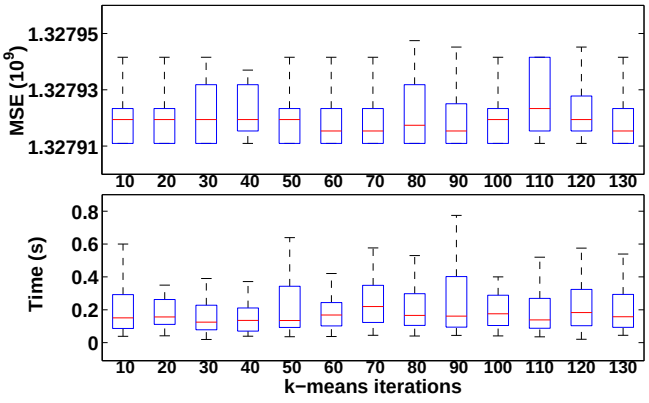


Fig. 6. MSE and time on different number of k-means iterations in PRS on data set S2.

We observe from the experiment on several synthetic data sets that the probability is 100% for $k_2 \leq M$ (see Fig. 7). The experiment is to run 100 times of PRS on data sets S1-S4, Aggregation and R15.

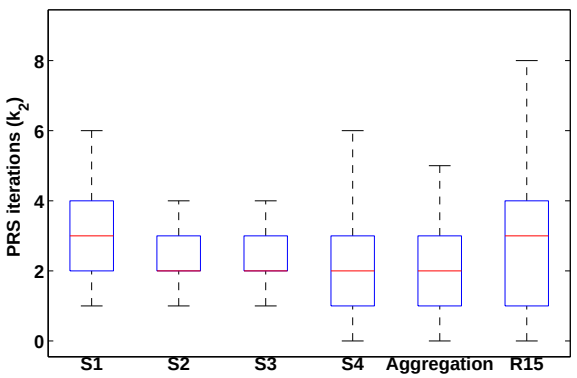


Fig. 7. Boxplot to show the PRS iterations for different synthetic data sets. The probability is 100% for $k_2 \leq M$, where $M = 15$ for S1-S4 and R15, and $M = 7$ for Aggregation.

One way to compare the performance of the methods is to plot the *MSE* values with increasing time. With enough processing time, the time-distortion figure can be used to check estimated quality at the time axis.

We performed 50 runs of each algorithm on each data set to study their average performance (see Fig. 9). Box plots of *MSE* values and processing time are used to reflect the performance of the algorithms in average. Each box includes minimum, median (the central red line), the 25th and 75th percentiles (the edges of the box) and maximum values.

2. http://cs.joensuu.fi/sipu/soft/

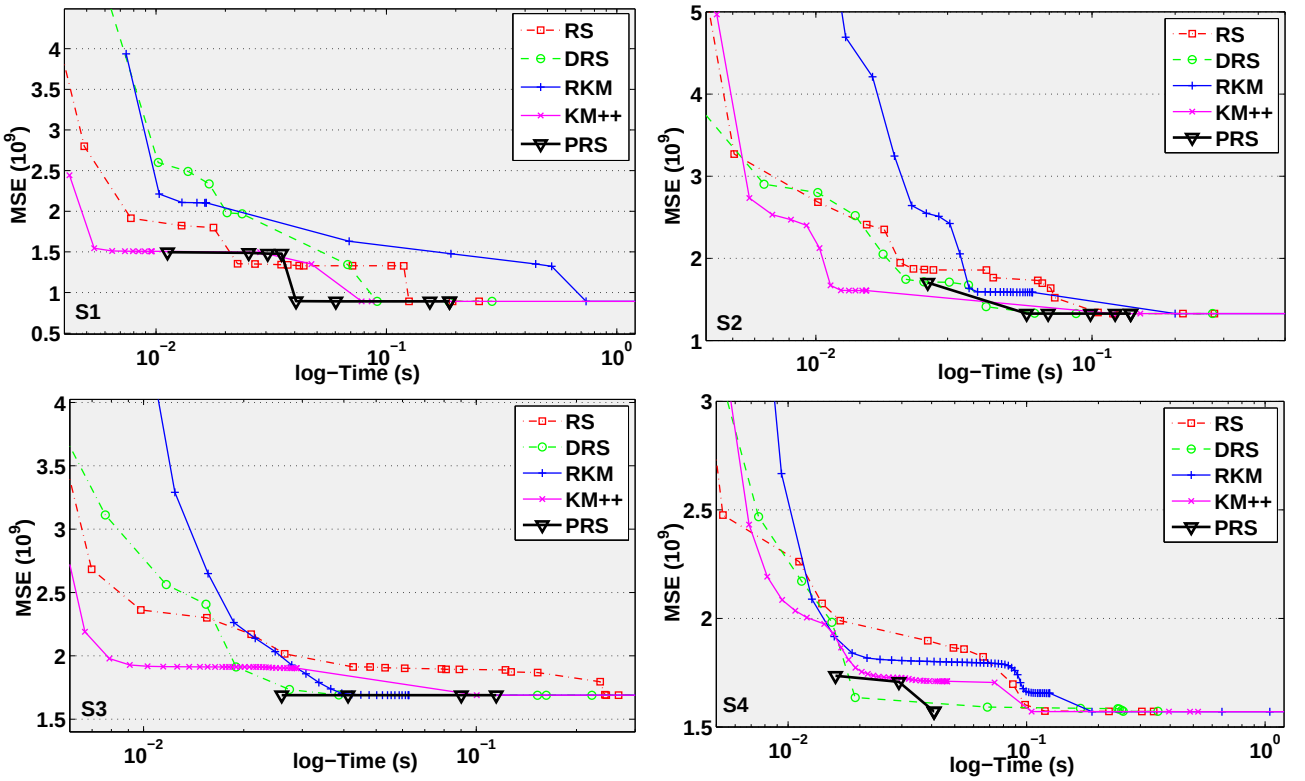


Fig. 8. MSE values with increasing time from clustering algorithms on S-sets.

The time-distortion plots on S-sets (S1-S4) are compared in Fig. 8. Among the clustering algorithms (RS, DRS, PRS, RKM, KM++), PRS works best in terms of the MSE value and processing time. Because of the stopping criterion, PRS stops while the other algorithms are still running. DRS reaches the local minimum faster than RS because DRS stops whenever there is no improvement and RS stops when the number of iterations has been reached. Deterministic selection converges faster than random selection. RKM is the most inefficient algorithm since not every repetition helps on the final result and a waste of computation exists in RKM. For example, too many repetitions are not improving the result on S4 (see Fig. 8). KM++ reaches local minimum comparably fast as RS, DRS and PRS, and the setting of repetitions for KM++ is over-set in the experiment. This arises the question that how many iterations are proper for RKM and KM++ in order to obtain a good performance in an efficient way.

As shown in the box plot for S-sets (Fig. 9), enough running time guarantees good performance of RKM and KM++. The degree of overlapping of S-sets increases the running time of RKM and KM++ and brings minor effect on swap-based clustering algorithms. The running times of RS is stable. The swapping candidates in deterministic swap depends on the selection criterion. Thus, both DRS and PRS have high variance on the processing time. PRS is a good choice according to its *MSE* values and processing time.

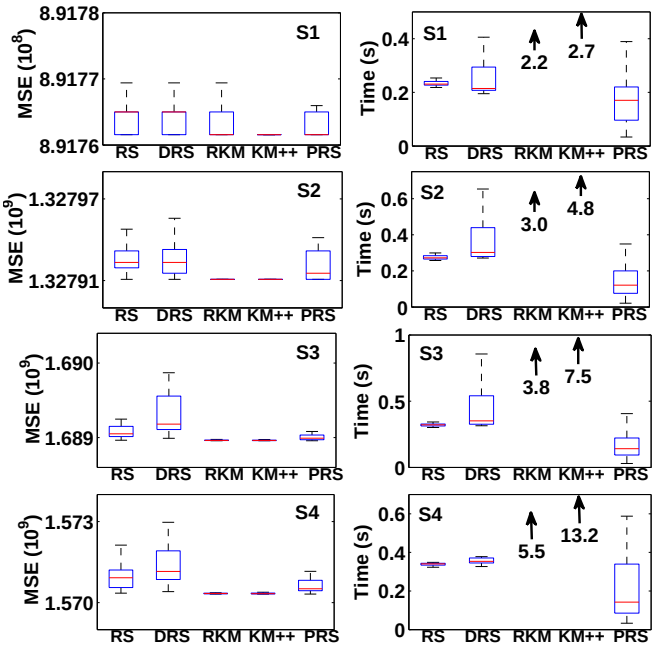


Fig. 9. Box plots of S-sets including minimum, 25th percentile, median, 75th percentile and maximum. The central red line represents median value, the edges of the box are the 25th and 75th percentiles.

CM and CT contain multi-dimensional data. When the data is high-dimensional, the feature space is usually sparse [27]. The standard k-means algorithm

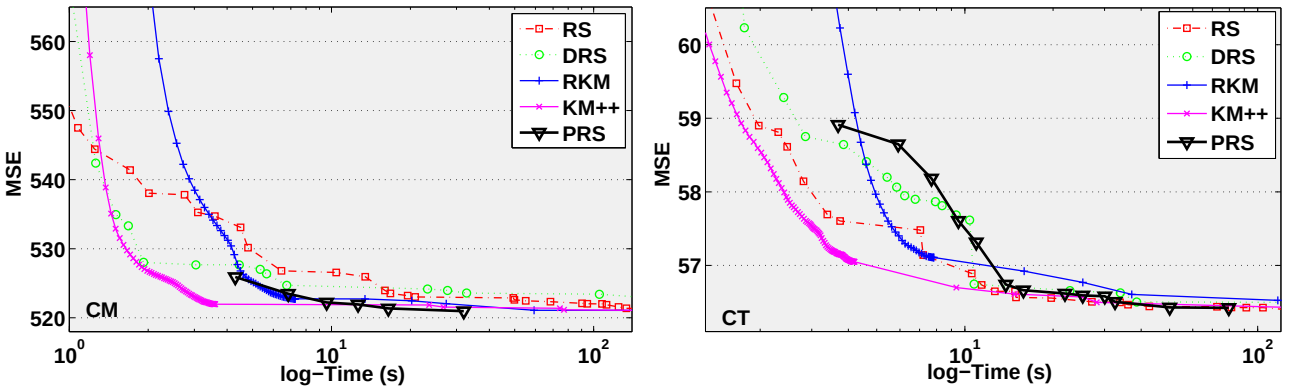


Fig. 10. MSE values with increasing time from clustering algorithms on CM and CT.

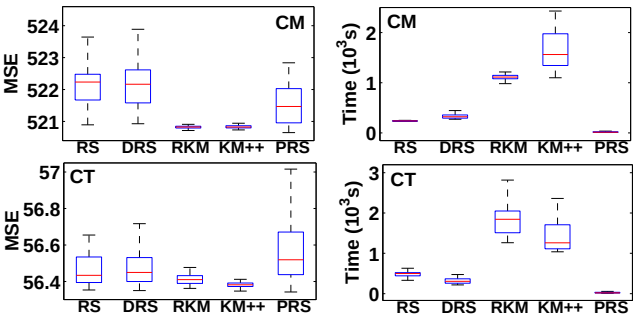


Fig. 11. Box plots of data sets CM and CT.

TABLE 5
Summary of the median processing times (in seconds).

	RS	DRS	RKM	KM++	PRS
S1	0.23	0.21	2.22	0.50	**0.2**
S2	0.27	0.30	2.97	4.59	**0.1**
S3	0.32	0.35	3.80	6.78	**0.1**
S4	0.34	0.35	5.47	12.93	**0.1**
A7	<**0.1**	<**0.1**	<**0.1**	<**0.1**	<**0.1**
R15	<**0.1**	<**0.1**	0.121	0.117	<**0.1**
BIRCH1	262	173	2413	1787	**134**
BIRCH2	315	413	535	539	**126**
CM	237	321	1112	1562	**14**
CT	497	306	1845	1261	**19**

for cluster analysis often do not work well in high dimensional spaces. Thus, the algorithms employing k-means are restricted by the performance of k-means. RKM has a little bit better result than KM++ on data CM, while KM++ works better than RKM on CT. In terms of MSE values, RKM and KM++ work better than swap-based algorithms on both CM and CT (Fig. 10). However, the running times of RKM and KM++ are higher than those for swap-based algorithms. For highly separated data space, the probability of getting a good swap is relatively low, which explains the high variance of MSE values for RS, DRS and PRS. PRS performs better than RS and DRS on CM; however, PRS is not stable on CT (Fig. 11). In terms of the processing time, PRS is still the most efficient one among the tested algorithms.

A summary table on processing time in Table 5 presents numerical results on different algorithms. PRS requires 26% to 96% less processing time than the others on different data sets.

4.3 An Application on Image Color Quantization

The most straightforward application of clustering algorithms in image processing is color quantization. When the input data set is the color space of images, clustering points in three-dimensional space are treated as standard color quantization. After the clusters have been located, typically the points in each cluster are averaged to obtain the representative color to which all colors in that cluster are mapped.

We compare the proposed clustering algorithm with other popular clusterings on the images[3] for color quantization. The images are in RGB color space with size of 481*321 pixels. In order to speed up the running time for all of the clustering algorithms, we reduce the amount of image data by a subsampling method [28]. The subsampling method can reduce the size of image from 14% to 42% while the running time reduced 55% to 94%. The difference of MSE values for original images and subsampled images is from -23% to 19%. Based on the numbers, we conclude that the subsampling method is applicable in color quantization.

The proposed method is compared to the algorithms including random local search (RS), Fuzzy c-means (FCM) and Genetic algorithm (GA) and kmeans++ (KM++). The evaluations of the clusterings by mean square error (MSE) and peak signal-to-noise ratio (PNSR) are listed in Table 6. Comparing the MSE and PNSR values, there is no clustering algorithm that works for all images. The performance is equally distributed among the algorithms and images. The proposed algorithm has best performance at the running time. A visualization of quantization results

3. http://cs.joensuu.fi/~zhao/DATA/

TABLE 6
MSE, PNSR (dB) and processing time (second) of different clusterings on subsampled images at quantization level 32.

image		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MSE	RS	173	118	80	256	68	492	100	**38**	**101**	174	162	**82**	**76**	229	12
	FCM	191	151	159	267	69	417	97	40	114	201	194	90	85	238	17
	GA	177	118	**79**	260	67	399	102	**38**	103	171	158	82	77	233	**11**
	KM++	**134**	**100**	80	**188**	**53**	236	69	41	123	**110**	144	93	84	**208**	16
	PRS	140	101	107	189	58	**202**	**67**	43	137	114	**129**	93	90	212	16
PSNR	RS	25.7	27.4	29.1	24.0	29.8	21.2	28.1	**32.4**	**28.1**	25.7	26.0	**29.0**	**29.3**	24.5	**37.5**
	FCM	25.3	26.4	26.1	23.9	29.7	21.9	28.3	32.1	27.6	25.1	25.3	28.6	28.8	24.4	35.7
	GA	25.7	27.4	**29.2**	24.0	29.8	22.1	28.1	**32.4**	28.0	25.8	26.2	29.0	**29.3**	24.5	**37.5**
	KM++	**26.9**	28.1	29.1	**25.4**	**30.9**	24.4	29.7	32.0	27.2	**27.7**	26.6	28.4	29.0	**24.9**	36.1
	PRS	26.7	**28.8**	27.9	**25.4**	30.5	**25.1**	**29.9**	31.8	26.8	27.6	**27.0**	28.4	28.6	**24.9**	36.1
time	RS	152	133	78	150	125	139	129	203	112	117	105	89	131	162	42
	FCM	73	47	41	66	80	52	62	74	73	58	51	108	58	58	24
	GA	642	712	407	833	889	789	785	4353	530	623	577	685	666	756	259
	KM++	226	174	96	207	184	174	201	298	158	158	144	111	183	222	47
	PRS	**17**	**46**	**13**	**4**	**5**	**34**	**13**	**17**	**42**	**16**	**17**	**3**	**19**	**19**	**34**

(a) original image (b) PRS (c) KM++

(d) RS (e) FCM (f) GA

Fig. 12. Sample quantization results for image 11 at quantization level 32. The main difference is shown in the red circles.

from different algorithms is shown in Fig. 12.

5 Conclusion

We proposed a novel evaluation criterion so-called centroid ratio based on centroids in prototype-based clustering, which compares two clusterings and detect unstably/incorrectly located centroids. The centroid ratio highly correlates with external indices and MSE. Since the proposed measure can detect incorrectly located clusters, it is employed as a swap criterion in Pairwise Random Swap algorithm. The algorithm is compared with other algorithms, such as Random Swap, Deterministic Random Swap, Repeated k-means, and k-means++. It is the most efficient method among these algorithms according to the experimental results.

In practice, users suffer from a problem of parameter setting of algorithms when there is little prior knowledge about data and algorithm. In the proposed algorithm, it is not necessary to set any parameters except the number of clusters.

References

[1] A.K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31(8), pp. 651–666, 2010.

[2] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, pp. 264–323, 1999.

[3] G.P. Babu and M.N. Murty, "Simulated annealing for selecting optimal initial seeds in the k-means algorithm," *Journal of Pure and Applied Mathematics*, vol. 25, pp. 85–94, 1994.

[4] K. Krishna and M.N. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, pp. 433–439, 1999.

[5] A. Likas, N. Vlassis, and J.J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, pp. 451–461, 2003.

[6] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, p. 10271035, 2007.

[7] P. Fränti and J. Kivijärvi, "Randomized local search algorithm for the clustering problem," *Pattern Analysis and Applications*, vol. 3(4), pp. 358–369, 2000.

[8] P. Fränti, O. Virmajoki, and V. Hautamäki, "Probabilistic clustering by random swap algorithm," *IAPR Int. Conf. on Pattern Recognition (ICPR'08)*, 2008.

[9] P. Hansen and N. Mladenovic, "J-means: A new local search heuristic for minimum sum-of-squares clustering," *Pattern Recognition*, vol. 34, pp. 405–413, 2001.

[10] B. Fritzke, "The LBG-U method for vector quantization - an improvement over LBG inspired from neural networks," *Neural Processing Letters*, vol. 5(1), pp. 35–45, 1997.

[11] P. Fränti and O.Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition*, vol. 39, pp. 761–765, 2006.

[12] H. Frigui and R. Krishnapuram, "Clustering by competitive agglomeration," *Pattern Recognition*, vol. 30(7), pp. 1109–1119, 1997.

[13] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," *KDD'09*, pp. 877–886, 2009.

[14] S.S. Khan and A. Ahmad, "Cluster center initialization algorithm for k-means clustering," *Pattern Recognition Lett.*, vol. 25, pp. 1293–1302, 2004.

[15] P. Fränti and O.Virmajoki, "On the efficiency of swap-based clustering," *Int. Conf. on Adaptive and Natural Computing Algorithms (ICANNGA'09)*, pp. 303–312, 2009.

[16] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, pp. 107–145, 2001.

[17] E. Dimitriadou, S. Dolnicar, and A. Weingassel, "An examination of indexes for determining the number of clusters in binary data sets," *Psychometrika*, vol. 67, pp. 137–160, 2002.

[18] C. Michele and M. Antonio, "A fuzzy extension of some classical concordance measures and an efficient algorithm for their computation," *Proc. of the 12th Int'l Conf. on Knowledge-Based Intelligent Information and Engineering Systems*, pp. 755–763, 2008.

[19] A. Rakhlin and A. Caponnetto, "Stability of k-means clustering," *Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA*, vol. 19, 2007.

[20] N.S. Netanyahu C.D. Piatko R. Silverman T. Kanungo, D.M. Mount and A.Y. Wu, "A local search approximation algorithm for k-means clustering," *Computational Geometry*, vol. 28, pp. 89–112, 2004.

[21] P. Fränti T. Kaukoranta and O. Nevalainen, "A fast exact gla based on code vector activity detection," *IEEE Trans. Image Processing*, vol. 9(8), pp. 1337–1342, 2000.

[22] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discov. Data*, vol. 1, pp. 1556–4681, 2007.

[23] E. Backer C.J. Veenman, M.J.T. Reinders, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24(9), pp. 1273–1280, 2002.

[24] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, vol. 1(2), pp. 141–182, 1997.

[25] A. Asuncion and D.J. Newman, "UCI machine learning repository," http://archive.ics.uci.edu/ml/, 2007.

[26] M. Ortega, Y. Rui, K. Chakrabati, K. Porkaew, S. Mehrotra, and T.S. Huang, "Supporting ranked boolean similarity queries in MARS," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 6, pp. 905–925, 1998.

[27] S. Dasgupta and L. Schulman, "A probabilistic analysis of EM for mixture of separated, spherical Gaussians," *Journal of Machine Learning Research*, pp. 203–226, 2007.

[28] T. Hasan, Y. Lei, A. Chandrasekaran, and J.H.L. Hansen, "A novel feature sub-sampling method for efficient universal background model training in speaker verification," *ICASSP'10*, pp. 4494–4497, 2010.

Paper **P6**

Q. Zhao, V. Hautamäki, I. Kärkkäinen and P. Fränti, "Random swap EM algorithm for Gaussian mixtures models", *manuscript*.

Random swap EM algorithm for Gaussian mixture models

Qinpei Zhao*, Ville Hautamäki, Ismo Kärkkäinen, Pasi Fränti

Abstract

Expectation maximization (EM) algorithm is a popular way to estimate the parameters of Gaussian mixture models. Unfortunately, its performance highly depends on the initialization. We propose a random swap EM for the initialization of EM. Instead of starting from a completely new solution in each repeat as in repeated EM, we make a random perturbation on the solution before continuing EM iterations. The removal and addition in random swap are simpler and more natural than split and merge or crossover and mutation operations. The most important benefit of random swap is its simplicity and efficiency. RSEM needs only the number of swaps as a parameter in contrast to complicated parameter-setting in Genetic-based EM. We show by experiments that the proposed algorithm is 9%-63% faster in computation time compared to the repeated EM, 20%-83% faster than split and merge EM except in one case. RSEM is much faster but has lower log-likelihood than GAEM for synthetic data with a certain parameter setting. The proposed algorithm also reaches comparable result in terms of log-likelihood.

Key words: Expectation Maximization; Random Swap EM; Gaussian Mixture Model; Split and Merge EM; Genetic-based EM; data clustering;

*School of Computing, University of Eastern Finland, Finland, FI-80101, Tel: +358 132517962, Email: qinpei.zhao@uef.fi

1. Introduction

Maximum likelihood (ML) estimation of the *Gaussian mixture models* (GMMs), does not lead to a closed form solution. However, if the estimation problem is reformulated in terms of so called latent or hidden variables, a numerical gradient ascent approach can be used. As the latent variables cannot be observed directly, *expectation maximization* (EM) [1, 2] algorithm iteratively refines the ML estimate by first calculating the expectation of the posterior of the latent variables, while keeping the parameters fixed. While keeping the posteriors fixed, the algorithm then computes the maximum of the parameters. This iterative process is guaranteed to converge.

EM has two well known deficiencies. First, user needs to know in advance the number of Gaussian components. Second deficiency is that the quality depends on the initial parameters. A number of methods have been proposed to attack both problems simultaneously [3, 4]. However, such a solution needs to change the optimization cost. In general, we assume that the problem of the number of components can be solved by a validity index, and therefore, we do not consider the number of components as a parameter to be optimized.

Initial parameters are needed for the first E-step. Unfortunately, not all initial parameters lead to the same unique solution when the algorithm converges [5]. Especially for Gaussian mixture models, log-likelihood landscape is multimodal [6]. A common way to address this problem is to run EM multiple times with different randomly chosen initial parameters [5] and pick the best solution as the result. We call this variant *repeated EM* (REM). The strategy gives good stability with respect to the log-likelihood and reduces dependency on the initialization [7]. However, the solution space is searched inefficiently in REM, because after each

restart it can take a long time to converge without any guarantee that it leads to an improved solution. Running time can be improved by computing in each iteration a bound on the locally optimal log-likelihood and stopping early if the bound shows no improvement [8].

Assuming that a complete restart is not necessary, search strategy based on changing only a part of the converged model can be utilized. One such strategy is to split one component into two and merge two other components [4, 9, 10, 11, 12]. A method utilizing this strategy is called *split and merge EM* (SMEM) [10], which searches systematically the best choice for the three components: one for split ($O(MN)$ operation, N is the data size and M is the number of components) and two for merge ($O(M^2N)$ operation). The choice is based on how well components match the local density of the data. Algorithm will terminate when no split and merge candidate brings improvement. Systematic approach needs to consider $O(M^3)$ triplets in total. In practice, the number of candidates searched is set lower than the number of all possible triplets.

Genetic-based EM (GAEM) [13] improves the repeated EM by considering a parallel set of solutions (populations) instead of sequential ones. Operations such as crossover, mutation and selection are applied to the population iteratively. A single-point crossover, which exchanges components between two populations is employed. Mutation selects the components with similar parameters and swaps them to random positions. A new generation of populations is finally obtained by a selection operation. There are five parameters involved in the algorithm. In general, GAEM can achieve a good result by a proper set of parameters.

Some other algorithmic strategies employed to escape a local maximum are: competitive learning [4], incremental clustering implemented in *greedy EM* (GEM) [14],

stochastic variants such as *stochastic EM* (SEM) [15] and *Monte Carlo EM* (MCEM) [16].

In this work, we use randomization instead of systematic search to select the component. Preliminary results of the proposed method were published in [17, 18]. In the proposed algorithm, *random swap EM* (RSEM), replaces the split and merge -operations by more general addition and removal -operations. Proposed operations are simple and efficient. Removing a component, which is an $O(1)$ time operation, is more straightforward than merging and only one component is involved. Creation of a new component is also simpler than splitting a component, where split is usually ill-posed (i.e., more variables than equations). GAEM has five parameters, all of which affect the running time and performance. Proposed method is thus simpler and easier to adapt to different datasets and applications.

In RSEM, randomly selected component is swapped to a new location in the feature space and the weight and covariance matrices are updated. The time complexity is $O(NM)$, which is the same as one EM iteration. Even though more iterations are needed by random swap approach due to its trial-and-error nature, the total number of candidates is significantly less than by systematic search such as SMEM or repeated EM. After the swap is performed, EM is iterated until convergence. New solution is accepted only if it improves the previous one. In principle, RSEM algorithm terminates when none of the possible NM swaps result in an improved solution [19]. However, a fixed number of swaps is sufficient in practice.

2. EM algorithm and its Variants

In this section, we first describe the existing methods that are compared to the proposed method, which is presented in Section 3.

2.1. EM algorithm

EM algorithm can be used to estimate *maximum likelihood* (ML) parameters of many different types of parametric densities. For GMMs, the goal is to maximize the following log-likelihood:

$$L(\Theta) = \log p(\boldsymbol{X}|\Theta) = \sum_{i=1}^{N} \log \sum_{j=1}^{M} \alpha_j \mathcal{N}(\boldsymbol{x}_i|\Theta_j), \tag{1}$$

where $\mathcal{N}(.|.)$ is Gaussian distribution, $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ is the observed d-dimensional data-set of N vectors, Θ is the GMM and $\Theta_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ are the mean vector and covariance matrix of the jth Gaussian, respectively. Finally, α_j is the mixture weight of the jth component. The parameters α_j must satisfy the following constraints:

$$\sum_{j=1}^{M} \alpha_j = 1, \quad \text{and}, \quad \alpha_j \geq 0, \quad j = 1, ..., M. \tag{2}$$

Unfortunately, closed-form solution of the (1) is not possible [1], since it contains the log of the sum. Maximization is then performed on the expectation of the complete-data log-likelihood, given posterior density of the latent variables [1]. This function is usually called the Q-function, and can be written in a concrete form for Gaussian mixtures as:

$$Q(\Theta, \Theta^{\mathrm{t}-1}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \tau_{ij} \left\{ \log \alpha_j + \log \mathcal{N}(\boldsymbol{x}_i|\Theta_j) \right\}. \tag{3}$$

$\Theta^{\mathrm{t}-1}$ are parameters estimated in the previous iteration. Maximization of Eq. (3), in terms of Θ can be performed easily, by keeping the posterior probabilities τ_{ij} fixed. Then, given estimated parameters, posterior probability of $\boldsymbol{x}_i$ from component j, τ_{ij} can be calculated as follows:

$$\tau_{ij} = \frac{\mathcal{N}(\boldsymbol{x}_i|\Theta_j)\alpha_j}{\sum_{l=1}^{M} \mathcal{N}(\boldsymbol{x}_i|\Theta_l)\alpha_l} \tag{4}$$

To find an initial set of parameters in EM algorithm, one possibility is to randomly select mean vectors and set equal weights and whole data covariance matrix for all components [20]. A more common practice is to first run k-means on the dataset to get hard partitioning. The initial mean vectors are directly the cluster centroids, partition covariance is the component covariance matrix and proportion of vectors in each partition is the component weight. Several short runs of k-means starting with random initial solutions each followed by a long run of EM is recommended in [7].

EM suffers from the local maximum problem [6]. A standard solution for the initialization problem (REM) is to repeat random initializations with k-means followed by EM [7]. The best performing solution, in terms of log-likelihood, is retained. This introduces a new parameter, the number of repeats. From the linearity of expectation, it is expected that the number of EM iterations in REM is multiplied by the number of repetitions. It means that the model quality can be improved by increasing the number of repetitions, but at the cost of linearly increasing the processing time.

2.2. Split-and-Merge EM

One strategy to overcome the sensitivity to initialization of EM algorithm is to identify parts of the solution that do not fit well to the data, and revise the solution by making local changes. When working in the component domain, we can change the solution by splitting a component into two and by merging two components into one. *Split and merge EM* (SMEM) [10] makes a systematic search through all possibilities for split and merge after which the algorithm selects the best candidates and performs the operations.

SMEM algorithm searches among the candidates composed of combinations

of all components i, j and k until the likelihood value improves. The candidates are sorted by the merge and split criteria. Merge criterion is based on the correlation of posterior probabilities of components i and j. The split criterion is based on the Kullback-Leibler divergence between component k and the local data density.

$$\begin{aligned} \mathrm{JMerge}(i,j) &= \frac{\tau_i(\Theta)^T \tau_j(\Theta)}{||\tau_i(\Theta)||||\tau_j(\Theta)||} \\ \mathrm{JSplit}(k) &= \int f_k(\boldsymbol{X},\theta_k) \log \frac{f_k(\boldsymbol{X},\theta_k)}{p_k(\boldsymbol{X},\theta_k)} dx \end{aligned} \tag{5}$$

where, $\tau_i(\Theta) = (\tau_{1i}(\Theta), ..., \tau_{Ni}(\Theta))$ is an N-dimensional vector consisting of the posterior probabilities for the ith component. T denotes the transpose operation and $1 < k \neq i \neq j < M$. The $f_k(\boldsymbol{X},\theta_k)$ is the local data density around the component k and the $p_k(\boldsymbol{X},\theta_k)$ is the empirical distribution. The merged components are combined linearly and the split component is split by adding constant movements on the original parameters. Then a partial EM step is performed on the merge and split candidate.

The original acceptance rule, line 7 in Algorithm 1, used the Q-function, instead of $L(\Theta)$ [10]. However, it was found in [21] that by doing so the global maximum might be accidentally rejected. In our experiments, we therefore use improvement of the log-likelihood as the acceptance rule.

A practical problem of split and merge approach is that the split and merge operations are not straightforward to design. The assumption behind split-and-merge approach is that only the components of the triplet (i, j, k) are affected and the rest of the model is unchanged. Merge operation has a closed-form solution when we assume that the distributions are Gaussian. However, it is not possible to find a unique solution to the problem of splitting one component into two.

One alternative was proposed in [12], where one randomly selected singular value decomposition basis vector of the covariance matrix is used to compute two new covariance matrices. It is also used in combination with the original mean vector to generate two new mean vectors.

Input: Data Set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$

Output: Parameters $\Theta = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ and log-likelihood $L(\Theta)$

```
[Θ_0, L(Θ_0)] ← EM(X);
while candidates left to process do
    Sort candidates (i, j, k)_{C_max} by JMerge and JSplit (equation 5);
    for c = 1 to C_max do
        [Θ', L(Θ')] ← partialEM((i, j, k)_c);
        [Θ*, L(Θ*)] ← EM(X, Θ');
        if (L(Θ*) > L(Θ)) then
            Θ = Θ*; L(Θ) = L(Θ*);
        end
    end
end
return Θ, L(Θ)
```

Algorithm 1: SMEM algorithm

Furthermore, due to the split and merge operations, $C_{\max} = M(M-1)(M-2)/2$ candidate triplets are generated. A systematic search through all possible triplets leads to $O(M^4 N I_{\mathrm{EM}})$ time complexity, where I_{EM} is the number of EM iterations needed to reach convergence after perturbation. Final processing time can be reduced by considering only top $C_{\max}$ candidates. In [10], $C_{\max}$ was set to

five. We first experimentally find suitable $C_{\max}$ before comparing SMEM to other methods.

Input: Data Set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, I_{EM}, I_g, I_p, p_c, t_{corr}

Output: Parameters $\Theta = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ and log-likelihood $L(\Theta)$

1 $[\Theta_p[I_p], L[I_p]] \leftarrow$ Initialization($\boldsymbol{X}$);
2 **for** *GAEM-iteration*=1 *to* I_g **do**
3 $[\Theta_p[I_p], L_1[I_p]] \leftarrow$ EM($\Theta_p[I_p]$, I_{EM});
4 $\Theta_c[H] \leftarrow$ crossover($\Theta_p[I_p]$, p_c); $H = I_p * p_c$;
5 $[\Theta_c[H], L_2[H]] \leftarrow$ EM($\Theta_c[H]$, I_{EM});
6 $[\Theta_s[I_p], L_1(\Theta_s[I_p])] \leftarrow$ Select ($\Theta_p[I_p]$, $\Theta_c[H]$, $L_1[I_p]$, $L_2[H]$);
7 $\Theta_s[I_p] \leftarrow$ mutation($\Theta_s[I_p]$, t_{corr});
8 $\Theta_p[I_p] \leftarrow \Theta_s[I_p]$;
9 **end**
10 execute lines 3 to 6 once;
11 $[\Theta, L] \leftarrow$ EM($\Theta_s[best]$, I_{EM});
12 return Θ, L

Algorithm 2: GAEM algorithm

2.3. Genetic-based EM

Genetic-based EM (GAEM) for learning Gaussian mixture models is proposed in [13]. Original design of GAEM includes the model selection. However, number of components M is left as a user defined parameter in our task definition. So, we have modified the algorithm by keeping M fixed and removing the part where decision regarding M is made. Also, instead of MDL criterion we use log-likelihood during the selection in Algorithm 2.

In GAEM, the single point crossover operator selects a component index. First child gets components before the index from first parent and from the index onwards from the second parent, and vice versa for the second child. Mutation operator selects components that model the data points similarly by using posterior probabilities (i.e., $JMerge(i, j)$). If there is a correlation above a given parameter limit, the components are moved to random positions. New generation is selected from parent and child populations.

There are two deficiencies in GAEM. One is that the algorithm involves multiple solutions (population). When the population size (I_p) is large enough, a good result is achieved but it increases the running time linearly. The other one is the parameters. For crossover, mutation and selection steps, parameters are involved. In crossover, a probability p_c determines the number of offsprings after crossover. A threshold for correlation coefficient t_{corr} between components is set for mutation. There are also parameters for GAEM iterations I_g and EM iterations I_{EM}.

3. Random Swap EM

The idea of the *random swap EM* (RSEM) algorithm is to alternate between simple perturbation to the solution by random swap and convergence towards nearest optimum by the EM algorithm. A random swap consists of removal and addition operations.

RSEM is presented in Algorithm 3. The initialization is performed as in the EM algorithm, described in Section 2.1. After the solution has been initialized, we perform t random swap iterations (called RS-iterations). During each iteration, a component is removed, a new one is added and the resulting solution is converged towards nearest optimum using EM algorithm. The best solution, in terms of log-

likelihood, is maintained as the starting point for the subsequent RS-iteration.

Input: Data Set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$

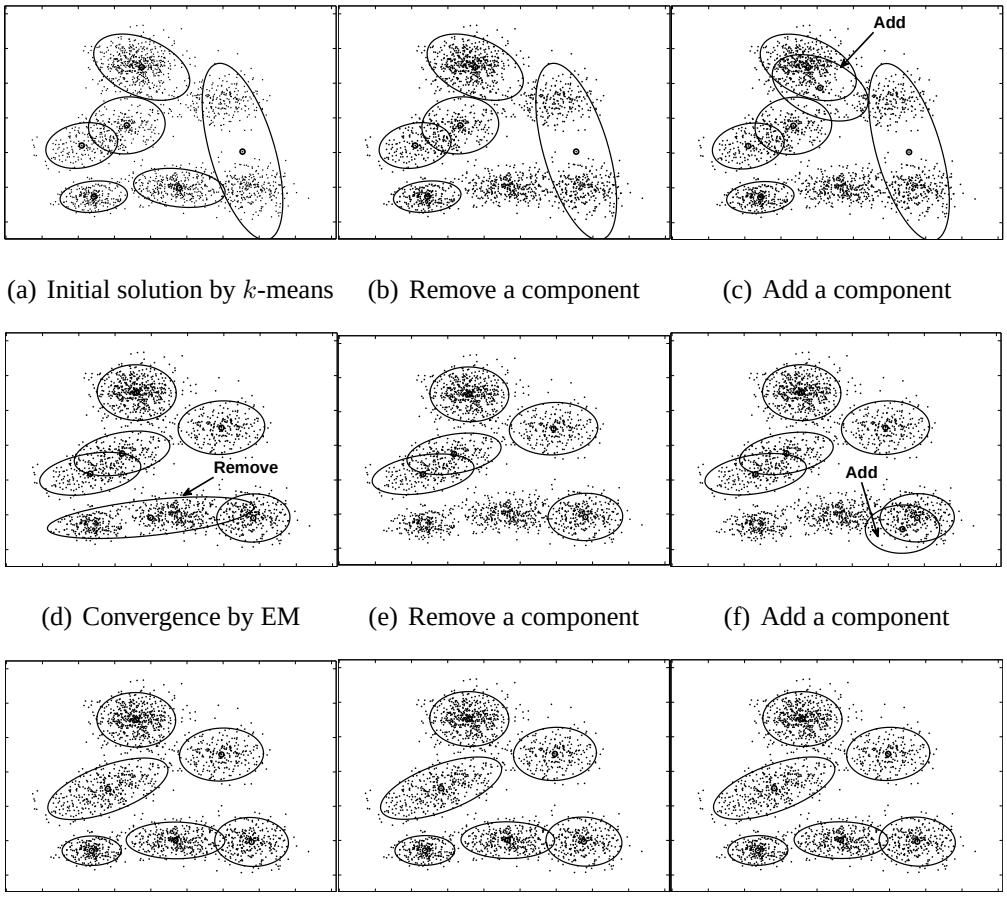
Output: Parameters $\Theta = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ and log-likelihood $L(\Theta)$

$[\Theta_0, L(\Theta_0)] \leftarrow$ Initialization($\boldsymbol{X}$);
for *RS-iteration*=1 *to* t **do**
    $r = \mathrm{U}(1, M)$, remove rth component;
    $p = \mathrm{U}(1, N)$, add at pth position (see equation 7);
    normalize weights $\boldsymbol{\alpha}$ to sum to 1;
    new parameters $\Theta^{\mathrm{s}} = \{\alpha^{\mathrm{s}}, \boldsymbol{\mu}^{\mathrm{s}}, \boldsymbol{\Sigma}^{\mathrm{s}}\}$;
    $[\Theta^{\mathrm{st}}, L(\Theta^{\mathrm{st}})] \leftarrow$ EM($\boldsymbol{X}, \Theta^{\mathrm{s}}$);
    if $L(\Theta^{\mathrm{st}}) > L(\Theta)$ **then**
        $\Theta = \Theta^{\mathrm{st}}$;
        $L(\Theta) = L(\Theta^{\mathrm{st}})$;
    end
end
return $\Theta, L(\Theta)$

Algorithm 3: RSEM algorithm

The removal operation is done by selecting a component r randomly among M components from uniform distribution, $r = \mathrm{U}(1, M)$. This is a constant-time operation.

The location of the new component is decided by selecting one data point, $\boldsymbol{x}_p, p = \mathrm{U}(1, N)$ and setting it as the mean vector of the new component. The new component is therefore more likely to be placed in areas of high point density, such as cluster centers, than areas of low point density.



(a) Initial solution by k-means (b) Remove a component (c) Add a component

(d) Convergence by EM (e) Remove a component (f) Add a component

(g) Convergence by EM (h) After 10 iterations of RSEM (i) Ground truth Gaussians

Figure 1: Result by RSEM for a two-dimensional Gaussian mixture density estimation problem. (a) An initial solution by 10 runs of k-means, (b)-(c) Removal and addition operation for the 1^{st} iteration , (d) Convergence by EM, (e)-(g) The procedures on the 3^{rd} iteration, (h) The final result by RSEM with 10 iterations, (i) Ground-truth Gaussians.

The best solution found so far, in terms of log-likelihood, is always used as the starting point for the next iteration. If a swap and EM iterations fail to produce a better solution than the starting point, the new solution is discarded. Swap will decrease the log-likelihood of the solution but it can also change the solution so

that iterating EM will move it towards different optimum.

The technique has been successfully applied to clustering with centroid model [22, 23, 24]. We observed that the effect of a bad initialization is diminished when random swap is used. We therefore expect random swap to yield good results with Gaussian mixture models, too.

The solution is fine-tuned with EM algorithm, so reasonable values for the weight and covariance matrix are sufficient. Suppose the current likelihood function $L(\Theta^{\mathrm{t}})$ at RS-iteration t is obtained by EM. Let r be the component selected for removal, and keep the rest of the components unchanged. The posterior probability is updated as follows:

$$\tau_{ij}^{\mathrm{s}} = \frac{\alpha_j^t \mathcal{N}(\boldsymbol{x}_i|\Theta_j^t)}{\sum_{l=1,l\neq r}^{M} \alpha_l^t \mathcal{N}(\boldsymbol{x}_i|\Theta_l^t)} \tag{6}$$

The equations for the new parameters of the r^{th} component are:

$$\begin{aligned} \boldsymbol{\mu}_r^{\mathrm{s}} &= \boldsymbol{x}_p \\ \alpha_r^{\mathrm{s}} = \alpha_r^{\mathrm{t}} \quad or \quad \alpha_r^{\mathrm{s}} &= \sum_{l=1,l\neq r}^{M} \left(\sum_{i=1}^{N} \tau_{il}^{\mathrm{s}} \right) \alpha_l^{\mathrm{t}} \\ \boldsymbol{\Sigma}_r^{\mathrm{s}} = \boldsymbol{\Sigma}_r^{\mathrm{t}} \quad or \quad \boldsymbol{\Sigma}_r^{\mathrm{s}} &= \sum_{k=1,k\neq r}^{M} \left(\sum_{i=1}^{N} \tau_{ik}^{\mathrm{s}} \right) \boldsymbol{\Sigma}_k \end{aligned} \tag{7}$$

In order to retain a valid Gaussian mixture model after the swap operation, weights $\alpha_i, 1 \leq i \leq M$ are normalized so that they sum up to 1. The time complexity of the addition operation is linear with respect to the model size M. After each swap, the new parameters Θ^{s} are set as initial solutions for EM. After EM has converged, we get a new likelihood value $L(\Theta^{\mathrm{st}})$ and we compute $\Delta L = L(\Theta^{\mathrm{st}}) - L(\Theta^{\mathrm{t}})$, If the difference is positive, the new parameter estimate replaces the previous best solution. Otherwise the new parameter estimate is discarded. This process is

repeated until all possible swap pairs are tried out and none is left to improve the solution. However, as a practical matter we restrict the total number of swaps to a user selectable number of RS iterations t. An example of RSEM algorithm operating on data is illustrated in Fig. 1.

To ensure a good solution, the number of iterations t for random swap should be set large enough so that there are enough successful swaps. Given the number of components M, the probability of selecting a component to be removed is $1/M$. The probability of selecting a point to be added is also $1/M$. Only if the point is inside one cluster, it will be a successful addition because EM can fine-tune the location even after then. Therefore it is not necessary to find near-optimal location during creation of a component. For a good swap to occur, a badly-placed component must be chosen and a location from the area where the component needs to move must also be chosen. Hence the probability of a single good swap is at least $1/M^2$, and $t > M^2$.

4. Summary of Iterative Methods

4.1. Comparing REM and RSEM

RSEM is faster than REM if it converges with fewer iterations after a swap. We prove in [18] that the increment of Q-function value by randomly swapping a component in RSEM is greater than that by a random restart on all components in REM, which leads to the fact that processing time of RSEM is less than REM for reaching the optimal result. We will approximate log-likelihood by the Q-function as in [8].

Theorem 4.1. *A random swap limits* $Q(\Theta^s, \Theta^{t-1}) - Q(\Theta^t, \Theta^{t-1})$ *into the lower*

and upper bounds of $[-\frac{N\alpha_r^t}{2}d, \frac{N\alpha_r^t}{2}d]$*, where* d *is the Mahalanobis distance between the swapped centroids* μ_r^t *and* μ_r^s*.*

Theorem 4.2. *For REM and RSEM, if* $d < \frac{1}{3}$*, the probability of* $Q(\Theta^s, \Theta^{t-1}) - Q(\Theta^t, \Theta^{t-1}) > Q(\Theta, \Theta^{t-1}) - Q(\Theta^t, \Theta^{t-1})$ *is* 1*. If* $d \geq \frac{1}{3}$*, the probability is* $\frac{1}{2} + \frac{1}{6d}$*.*

We see that the farther the new component is from the original, the closer to $P = 1/2$ we approach. However, REM will not have a higher probability than RSEM to reach a high Q-function value.

4.2. Comparison of time complexities

The time complexities of the algorithms are shown in Table. 1. M and N are the number of clusters and data vectors, respectively. S represents the number of REM repetitions, the number of RSEM swaps and the number of SMEM iterations with improvement. Parameters I_1, I_2 and I_3 are the iteration counts of EM convergence in the algorithms and C in SMEM indicates the number of candidates, which is set $C = 20$ in our experiments. Parameter I_g is the number of generations, I_{EM} is the number of EM iteration used in GAEM and I_p is the population size.

REM and RSEM have similar strategies. The difference is in the number of EM iterations to converge in both methods. Since not every run of EM contributes to the final result in REM, the proposed RSEM algorithm, which changes only a part of the solution, achieves better or same result faster than REM. This is shown theoretically in [18] and experimentally in Section 5. For SMEM, the number of SMEM iterations with improvement S takes a major role in the time complexity of SMEM. It highly depends on the size of search space caused by the the number of candidates C. RSEM is faster than SMEM when $I_2 \leq M + CI_3$. The merge

Table 1: Time complexity analysis on the methods.

			total
REM	EM	$O(I_1MN)$	$O(SI_1MN)$
SMEM	merge	$O(MN + M^2N)$	$O(S(M^2N + CI_3MN))$
	split	$O(MN + N \log N)$	
	EM	$O(3N)$	
GAEM	mutation	$O(M^2)$	$O(I_gI_p^2I_{EM}MN)$
	crossover	$O(I_pM)$	
	EM	$O(I_p^2MNI_{EM})$	
RSEM	removal	$O(1)$	$O(SI_2MN)$
	addition	$O(MN)$	
	EM	$O(I_2MN)$	

operation in SMEM takes much more time than removal in RSEM. Thus, RSEM is faster than SMEM in most cases. In GAEM, number of generation I_g plays a similar role as S, then RSEM is faster than GAEM if an average EM iterations are less than $I_p^2 I_{EM}$. On the other hand, we can also restrict EM iterations in RSEM to I_{EM}, then extra computations caused by GAEM is quadratic to population size.

5. Experimental Results

We tested the algorithms[1] using both synthetic and real data sets from various sources summarized in Table 2. We divide the sets into two categories. The first category is synthetic data sets. These are fairly small and contain a known number of clusters. In the tests, we match the number of components with the number

[1] `http://cs.joensuu.fi/sipu/soft/`

of clusters whenever the number of clusters is known. The second category is large data sets obtained from UCI Machine Learning Repository [25]. We set the number of components to 15 for CM and 20 for CT.

Table 2: Attributes of the data sets used in our experiments.

Data sets	Name	Dimension	Data Size	No. of Clusters
Synthetic	S1-S4 [26]	2	5000	15
	R15 [27]	2	600	15
Real	CM [28]	9	68040	15
	CT [28]	16	68040	20

In all experiments Gaussian mixture models are restricted to diagonal covariance matrices. The baseline algorithm is the REM algorithm. Initialization of the GMM for each repetition is described in Section 2.1. RSEM is given one random initial solution and the same number of RS-iterations is performed as the number of random solutions given to REM. The EM algorithm or partial EM algorithm is allowed to iterate until convergence (threshold = $1.53e-05$), except in GAEM, $I_{EM} = 3$.

The number of candidates $C_{\max}$ considered in each SMEM round is fixed to 20 as it seems to provide the best accuracy and processing time trade-off (see supplementary[2]). Increasing the number of candidates closer to maximum $O(M^3)$ does not improve the accuracy at all. SMEM algorithm immediately accepts a candidate that results in a better solution. When none of the $C_{\max}$ candidates result in improvement, the algorithm stops.

[2] `http://cs.joensuu.fi/~zhao/Software/supplementary1.pdf`

For GAEM, both I_p and I_g affect the running time. However, the result in terms of log-likelihood depends more on I_p. An experiment on different combinations of I_p and I_g on data S2 is conducted (see the supplementary file). The number of generations helps little to improve the log-likelihood, which however brings high running time. Thus, we select $I_g = 10$. The population size I_p affects the result clearly. It seems the log-likelihood is stable when $I_p > 20$ for S2. However, since the running time of GAEM (proportional to I_p^2) depends highly on I_p, we choose $I_p = 15$ to reduce the running time. The crossover probability $p_c = 0.8$ and $t_{corr} = 0.95$ following the setting in [13].

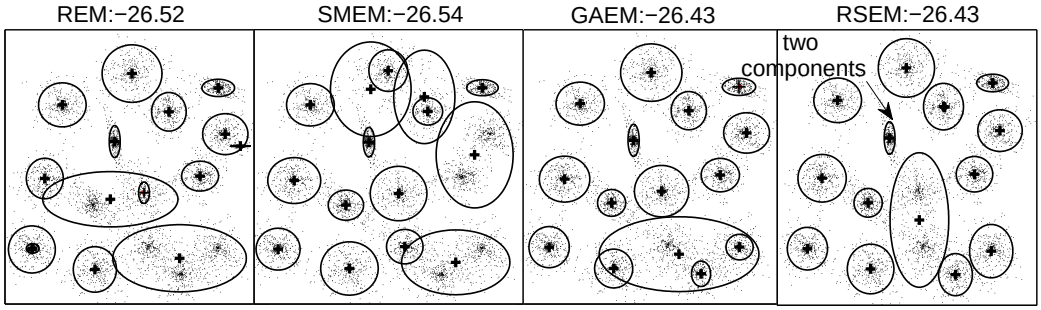


Figure 2: Gaussian models on data S2 estimated from EM variants.

We demonstrate the Gaussian models estimated from REM, SMEM, GAEM and RSEM on data set S2 in Fig. 2. The experiment is conducted by 20 repetitions. The average among them in terms of log-likelihood is shown. The models are displayed as ellipses. REM and SMEM are clearly worse in parameter estimation than GAEM and RSEM.

For S1 to S4, ground-truth distributions are available. For comparing the GMMs obtained from different EM variants, we calculate the squared Euclidean distance between estimated and ground-truth GMMs using the closed-form solu-

tion in [29]. The distance values are the average of 50 results. There are two out of four cases that RSEM is closer to ground-truth than competing methods even though log-likelihood is the best in all cases. It implies that in terms of parameter estimation by likelihood is not always a good proxy. The goal metric, however in the present work is log-likelihood.

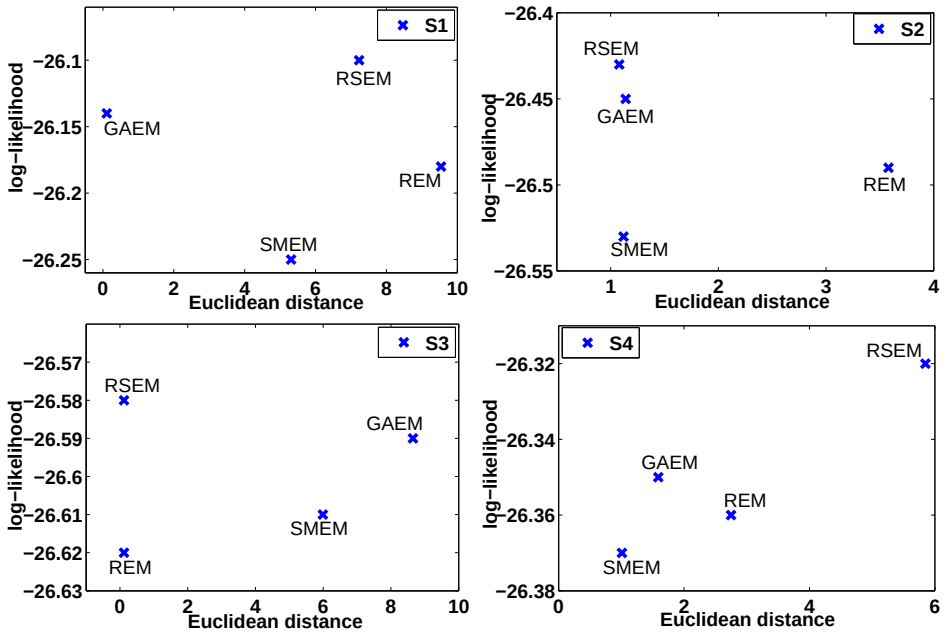


Figure 3: Squared Euclidean distance between ground-truth GMM and estimated solutions vs. log-likelihood values.

To obtain robust estimates of average log-likelihood and CPU time values, each algorithm is repeated 50 times. A summary on the mean log-likelihood values is presented in Table 3 and processing time in Table 4. Statistical tests run on the distributions of log-likelihood values and processing times showed that the processing time follows Gaussian distribution while log-likelihoods do not. Furthermore, the shapes of the log-likelihood distributions differ from each other.

Hence statistical significance tests such as t-test or normal rank-sum test can not be used for log-likelihoods. Thus, we performed t-test only on the processing time of RSEM and other three methods (REM, SMEM and GAEM) respectively to emphasize that RSEM is significantly faster than the EM variants with comparable or better log-likelihood. We use an asterisk (*,$p < 0.05$) to indicate the significant difference between RSEM and other EM variants.

Table 3: Summary of the mean log-likelihood values.

	S1	S2	S3	S4	R15	CM15	CT20
REM	-26.20	-26.51	-26.63	-26.37	-6.48	-10.34	-3.64
SMEM	-26.25	-26.53	-26.61	-26.38	-6.57	-10.35	-3.65
GAEM	**-26.11**	**-26. 43**	**-26.59**	-26.34	**-6.35**	-10.35	-3.65
RSEM	-26.15	-26.45	-26.60	**-26.34**	-6.43	**-10.33**	**-3.63**

Table 4: Summary of the mean processing times (seconds).

	S1	S2	S3	S4	R15	CM15	CT20
REM	3.18*	3.94*	4.59*	4.07*	0.32*	794*	2551*
SMEM	2.29*	2.80*	3.34*	4.38*	0.29*	2267*	961
GAEM	7.09*	6.82*	6.45*	6.59*	1.13*	**157**	**315**
RSEM	**1.27**	**1.66**	**1.71**	**1.70**	**0.21**	355	1568

In processing time SMEM can vary greatly. The variance mainly comes from the $C_{\max}$ candidates. The algorithm stops if there is no improvement among the candidates, which decreases the running time in some cases. This is also reflected in log-likelihoods for CT data set. SMEM is capable of improving the initial solutions according to log-likelihood, but the effort needed varies greatly, resulting

in large variation in running times. The other algorithms are not affected much by the data set. Difference in running time between REM and RSEM is explained by the need to improve the entire model in REM versus the smaller changes in RSEM.

GAEM has good performance in terms of log-likelihood, however, it is much slower than RSEM for synthetic data. For real data, the running time is faster than RSEM, however the log-likelihood is worse. This is a major difficulty in using GAEM in practical applications. How to set parameters for a new dataset in such way that quality of the solution is maintained while processing time is kept in control. In contrast, RSEM offers simplicity to users. If processing time is not an issue, RSEM can be run until convergence, and then no parameter is required.

6. Conclusions

We proposed a random swap EM algorithm in order to get rid of the tendency of the standard EM algorithm to get stuck in a local maximum. The proposed RSEM indicates that it is not necessary to start from the beginning in each restart as it does in the repeated EM. The RSEM is also shown to be simpler and more efficient than other EM variants. The removal and addition operations in RSEM are more general and simpler than split and merge operations in SMEM. They use less parameters than crossover and mutation in GAEM, where crossover involves two populations at a time and a criterion is needed in mutation. Comparing the proposed algorithm to the REM, we found that RSEM reached higher or comparable level of log-likelihood 9%-63% faster, which was proved by a bound derived from formulas. RSEM is also easier to implement and more efficient than the split-and-merge EM (20%-83% faster). Genetic EM has good performance, however, the

complicated parameter setting makes it less useful in practice.

The number of swaps is a key parameter in the proposed method, which decides the performance of RSEM. As a future work, we plan to investigate ways to automatically select the number of swaps, as well as theoretical support for random swap strategy in Gaussian mixture models.

References

[1] Bishop, C.: Pattern Recognition and Machine Learning. Springer Verlag (2006)

[2] Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of Royal Statistical Society B **39** (1977) 1–38

[3] Figueiredo, M., Jain, A.: Unsupervised learning of finite mixture models. IEEE Transactions on Pattern Analysis and Machine Intelligence (2002) 381–396

[4] Zhang, B., Zhang, C., Yi, X.: Competitive EM algorithm for finite mixture models. Pattern Recognition **37** (2004) 131–144

[5] McLachlan, G., Peel, D.: Finite Mixture Models. John Wiley &Sons (2000)

[6] Mclachlan, G., Krishnan, T.: The EM algorithm and extensions. John Wiley &Sons (1996)

[7] Biernacki, G., Celeux, C., Govaert, G.: Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. Computational Statistics and Data Analysis **41** (2003) 561–575

[8] Zhang, Z., Dai, B., Tung, A.: Estimating local optimums in EM algorithm over Gaussian Mixture Model. Proceedings of the 25th International Conference on Machine Learning (2008) 1240–1247

[9] Li, Y., Li, L.: A split and merge EM algorithm for color image segmentation. IEEE Intl. Conf. on Intelligent Computing and Intelligent Systems (2009) 395–399

[10] Udea, N., Nakano, R., Gharhamani, Z., Hinton, G.: SMEM algorithm for mixture models. Neural Computation **12** (2000) 2109–2128

[11] Wang, H., Luo, B., Zhang, Z.: Estimation for the number of components in a mixture model using stepwise split-and-merge EM algorithm. Pattern Recognition Letters **25**(16) (2004) 1799–1809

[12] Zhang, Z., Chen, C., Sun, J., Chan, K.: EM algorithms for Gaussian mixtures with split-and-merge operation. Pattern Recognition **36**(9) (2003) 1973–1983

[13] Pernkopf, F., Bouchaffra, D.: Genetic-based em algorithm for learning gaussian mixture models. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(8) (2005) 1344–1348

[14] Verbeek, J., Vlassis, N., Krose, B.: Efficient greedy learning of Gaussian mixture models. Neural Computation **15**(12) (2003) 469–485

[15] Celeux, G., Diebolt, J.: The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. Computational Statistics Quaterly **2** (1985) 73–82

[16] Wei, G., Tanner, M.: A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. Journal of the American Statistical Association **85** (1990) 699–704

[17] Zhao, Q., Hautamäki, V., Kärkkäinen, I., Fränti, P.: Randon swap EM algorithm for finite mixture models in image segmentation. In: Proc. IEEE Int. Conf. on Image Processing, Cairo, Egypt (November 2009) 2397–2400

[18] Zhao, Q., Hautamäki, V., Fränti, P.: Rsem: An accelerated algorithm on repeated em. 2011 Sixth International Conference on Image and Graphics (2011) 135–140

[19] Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C., Silverman, R., Wu, A.Y.: A local search approximation algorithm for k-means clustering. Computational Geometry: Theory and Aplications **28** (2004) 89–112

[20] Figueiredo, M., Jain, A.: Unsupervised learning of Finite Mixture Models. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(3) (2002) 381–396

[21] Minagawa, A., Tagawa, N., Tanaka, T.: SMEM algorithm is not fully compatible with maximum-likelihood framework. Neural Computation **14**(6) (2002) 1261–1266

[22] Fränti, P., Kivijärvi, J.: Randomized local search algorithm for the clustering problem. Pattern Analysis and Applications **3**(4) (2000) 358–369

[23] Fränti, P., Virmajoki, O., Hautamäki, V.: Probabilistic clustering by random swap algorithm. 19th Int. Conf. on Pattern Recognition **55**(4) (2008) 287–314

[24] Merz, P.: An iterated local search approach for minimum sum-of-squares clustering. Advances in Intelligent Data Analysis V (2003) 1–38

[25] Asuncion, A., Newman, D.: UCI machine learning repository. `http://archive.ics.uci.edu/ml/` (2007)

[26] Fränti, P.: Clustering data sets. `http://cs.joensuu.fi/sipu/datasets/` (2009)

[27] Veenman, C., Reinders, M., Backer, E.: A maximum variance cluster algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(9) (2002) 1273–1280

[28] Ortega, M., Rui, Y., Chakrabati, K., Porkaew, K., Mehrotra, S., Huang, T.: Supporting ranked boolean similarity queries in MARS. IEEE Transactions on Knowledge and Data Engineering **10**(6) (1998) 905–925

[29] Helen, M., Virtanen, T.: Query by example of audio signals using euclidean distance between gaussian mixture models. ICASSP'07 (2007) 225–228

Paper **P7**

©2009 IEEE. Reprinted, with permission.
Q. Zhao, V. Hautamäki, I. Kärkkäinen and P. Fränti, "Random swap EM algorithm for finite mixtures models in image segmentation", in *IEEE Int. Conf. on Image Processing*, pp. 2397–2400, Cairo, Egypt, 2009.

RANDOM SWAP EM ALGORITHM FOR FINITE MIXTURE MODELS IN IMAGE SEGMENTATION

Qinpei Zhao[*1], *Ville Hautamäki*[1], *Ismo Kärkkäinen*[2], *Pasi Fränti*[1,3]

[1]Department of Computer Science, University of Joensuu, Finland
[2]Institute for Infocomm Research, A*STAR, Singapore
[3]School of Computer Science, Fudan University, China

ABSTRACT

The Expectation-Maximization (EM) algorithm is a popular tool in estimating model parameters, especially mixture models. As the EM algorithm is a hill-climbing approach, problems such as local maxima, plateau and ridges may appear. In the case of mixture models, these problems involve the initialization of the algorithm and the structure of the data set. We propose a random swap EM algorithm (RSEM) to overcome these problems in Gaussian mixture models. Random swaps are repeatedly performed in our method, which can break the configuration of the local maxima and other problems. Compared to the strategies in other methods, the proposed algorithm has relative improvements on log-likelihood value in most cases and less variance than other algorithms. We also apply RSEM to the image segmentation problem.

Index Terms— EM algorithm, unsupervised learning, mixture models, image segmentation

1. INTRODUCTION

As a standard method for the fitting of finite mixture models, particularly normal mixture models, the use of EM algorithm [1] has been demonstrated for the analysis of data from a wide variety of fields. However, the EM algorithm has several drawbacks: it is sensitive to initialization, it is known to get stuck at local optimal solutions, and sometimes it converges to the boundary of the parameter space [2].

To overcome the problems mentioned above, many variants of the EM algorithm have been proposed. Several methods for choosing sensible starting values have been discussed in [3]. To address the inappropriate distribution of the components in data space when locally trapped, Ueda and Nakano proposed the Split and Merge EM (SMEM) algorithm [4]. It escapes the local maxima in many situations and performs better than DAEM [5]. However, the split and merge operation in the SMEM algorithm is an ill-posed problem according to [6]. Greedy EM (GEM) algorithm was proposed in [7]. It reduces the problem of learning a k-component mixture model to a sequential learning of two-component model, and it offers a mechanism of dynamically allocating new components outside the overpopulated center regions. In addition, its time complexity is lower than that of SMEM.

In this paper, we propose a randomized version of the EM algorithm to overcome the limitations of the EM variants as mentioned above. The motivation of the random swap EM (RSEM) can be considered as random perturbations of the results generated by EM. The reasons for introducing the randomization are: the random perturbations prevent the proposed algorithm from staying near the unstable or hyperbolic fixed points of EM, as well as from its stable fixed points corresponding to insignificant local maxima of the likelihood function. Consequently, the slow convergence of the EM algorithm can also be avoided. Moreover, because of the randomization in the algorithm, it becomes less sensitive to its initialization, which is one of the main reasons that cause the local maxima problem.

*The author is funded by Nokia Foundation, Finland.

2. EM ALGORITHM AND ITS VARIANTS

2.1. *Conventional EM algorithm*

We briefly review the main features of the EM algorithm [8].

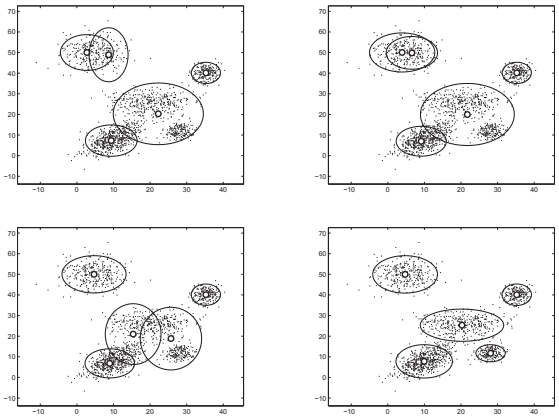
The log-likelihood function for *complete data* $Z = \{X, Y\}$, where $Y = \{y_1, .., y_n, ..., y_N\}$ is the observed data, $X = \{x_1, .., x_n, ..., x_N\}$ is the missing information, is defined as:

$$L(\Theta|Z) = \log p(X, Y; \Theta) = \log \left(p(Y|X; \Theta)p(X)\right) \quad (1)$$

N is the data size and Θ is an unknown parameter set and $p(X, Y; \Theta)$ is the joint probability density of Z and $p(X)$ is the probability density of X. The log-likelihood $L(\Theta|Z)$ is unobservable since $p(X)$ is unknown.

The log-likelihood function for *incomplete data* can be defined as:

$$L(\Theta|Y) = \log \prod_{n=1}^{N} p(y_n|\Theta) = \sum_{n=1}^{N} log \sum_{m=1}^{M} \alpha_m g_m(y_n; \theta_m) \quad (2)$$



(a) Initializations (b) EM results

Fig. 1. The effect of initialization on EM algorithm: two different initial solutions (left); two EM results (right).

Here M is the number of components, α_m is the mixing proportion of the m^{th} component, which satisfy the constraints:

$$\sum_{m=1}^{M} \alpha_m = 1, and, \alpha_m \geq 0, m = 1, ..., M. \quad (3)$$

The $g_m(y_n; \theta_m)$ is a d-dimensional density model corresponding to the m^{th} component, where θ_m is the parameters of it, and y_n is the n^{th} observed data point.

It is difficult to optimize Equation 2 because it contains the log of the sum. Hence the expectation function of $L(\Theta|Z)$ given Y is defined as Q function in the EM algorithm: $Q(\Theta; \Theta^{(k)}) = E\{L(\Theta|Z)|Y, \Theta^{(k)}\}$. The EM algorithm is as follows:

1. Expectation-step: calculate Q function $Q(\Theta; \Theta^{(k)})$;
2. Maximization-step: choose Θ to be any value which maximizes $Q(\Theta; \Theta^{(k)})$.

It may not be numerically feasible to find the value of Θ that globally maximizes the Q function. That is, one chooses $\Theta^{(k+1)}$ to increase the Q function $Q(\Theta; \Theta^{(k)})$ over its value at $\Theta = \Theta^{(k)}$, rather than to globally maximize it over all Θ. Under suitable regularity conditions, $\Theta^{(k)}$ converges to a stationary point of $L(\Theta)$. However, when there are several stationary points (local maxima, minima, saddle points), $\Theta^{(k)}$ does not necessarily converge to a significant local maximum of $L(\Theta)$ [9].

In practice, the EM algorithm has been observed to be extremely slow in some applications. As mentioned in [10], slow convergence appears when the proportion of missing information is high. Moreover, the convergence rate of EM depends on the initial positions. When the log-likelihood surface is littered with saddle points and sub-optimal maxima, final solution of EM greatly depends on its initial solution. For example, as shown in Fig.1, with two different initial solutions generated by k-means and the same data set, EM gives different final solutions. A solution could be a random initialization with several runs of EM itself or k-means. Common practice with EM is to first iterate k-means and then use the partitions to initialize component weights and covariance matrices. This has the advantage that we do not need to make educated guesses about the initial covariance matrices or use a scaled version of the covariance matrix of the entire data set.

2.2. *Variants of the EM algorithm*

Split and merge strategies have been often used in image segmentation. An iterative split-and-merge algorithm is used to generate codebook in vector quantization [11]. Splitting is applied in the X-means algorithm [12] to to overcome the problems existing existing in K-means.

With the split and merge operations in EM algorithm, the criteria for choosing the candidates should be considered. There are many ways to decide the criteria for splitting and merging based on the data set and algorithm itself. Consequently, finding a proper criterion is one of the constraints in SM operations. In the proposed method, random swap (RS) is an operation to randomly pick a component and swap it to another randomly selected place. The original configuration of the whole solution is perturbed in an uncertain way by RS operation without changing the number of components. We will see the different results coming from these two operations in the next section.

Two existing variants of EM algorithm are introduced for the purpose of comparisons. One is SMEM algorithm [4] with unchanged number of components. The other is the GEM algorithm [7], which only employs addition operation and the number of components increases with each addition. The key point of SMEM is

$$Q^* = Q_i^* + Q_j^* + Q_k^* + \sum_{m \neq i,j,k} Q_m^* \quad (4)$$

Here, Q is the Q-function mentioned in Section 2.1, and m represents the components. SMEM performs split and merge on the i^{th}, j^{th}, k^{th} components until a better result is achieved (i.e. Q^* improves).

The GEM employs the strategy that starts with one component and optimally adds new components one after another. Let $f_m(y) = \sum_{m=1}^{M} \alpha_m g_m(y_n; \theta m)$. The optimal new component $\phi(y_n; \theta)$ is found by log-likelihood, and corresponding mixing weight needs to satisfy:

$$\{\theta^*, \alpha^*\} = \arg\max_{\alpha} \sum_{n=1}^{N} \log[(1-\alpha) f_m(y_n) + \alpha \phi(y_n; \theta)] \quad (5)$$

The Greedy EM algorithm repeats these two steps until a stopping criterion is met: insert a new component to split the original one and apply EM until convergence.

Table 1. Parameters estimated from different EM-based algorithms

Parameters: Θ	True values: Θ^*	Initial values: Θ^0	EM: $\Theta^\wedge$	SMEM: $\Theta^\wedge$	RSEM: $\Theta^\wedge$
α_1	0.333	0.333	0.318	0.005	0.332
α_2	0.333	0.333	0.379	0.694	0.336
α_3	0.333	0.333	0.303	0.301	0.332
μ_1	(0, -2)	(-1, 0)	(-1.175, -0.054)	(0.482, -2.316)	(-0.086, -2.023)
μ_2	(0, 0)	(0, 0)	(-0.049, 0.006)	(-0.035, 0.929)	(-0.06, -0.02)
μ_3	(0, 2)	(1, 0)	(-1.126, 0.105)	(-0.1, -2.05)	(-0.01, 2.09)
Σ_1	(2, 0; 0, 0.2)	(1, 0; 0, 1)	(1.41, 0; 0, 2.99)	(0.43, 0; 0, 4.6)	(1.81, 0; 0, 0.17)
Σ_2	(2, 0; 0, 0.2)	(1, 0; 0, 1)	(0.80, 0; 0, 3.06)	(2.08, 0; 0, 1.53)	(2.20, 0; 0, 0.19)
Σ_3	(2, 0; 0, 0.2)	(1, 0; 0, 1)	(1.38, 0; 0, 2.93)	(1.81, 0; 0, 0.16)	(1.96, 0; 0, 0.21)

2.3. *Random Swap EM*

As EM starts with a random (potentially poor) solution, the final result of EM highly depends on the initialization. To avoid the sensitivity to initialization, RSEM randomly picks a component and relocates it to a random position. The parameters of the components will be modified accordingly. The procedure of randomization breaks up the configuration of the previous step, which can overcome the problems of EM. Moreover, the underlying EM characteristic helps to find good estimates of the parameters in a comparatively small number of iterations. The algorithm can be described as follows:

1. Initialization by several runs of k-means;

2. Perform a number of RSEM iterations and select the best solution.

Randomly pick up one component (i) and one position among the points (y_x). Update the Q function as follows:

$$Q^* = Q_i^{'} + \sum_{j \neq i} Q_j \tag{6}$$

Here $Q_i^{'}$ is the updated Q function of the randomly selected component where the parameters are $\Theta_i^{'} = \{\alpha_i^{'}, \mu_i^{'}, \Sigma_i^{'}\}$. The new parameters are: $\mu_i^{'} = y_x$, $\alpha_i^{'} = \sum_{j=1, j\neq i}^{M} (\alpha_j * P_j)$, where P_j is the posterior probability and the covariance $\Sigma_i^{'}$ is kept unchanged. Maximize the Q function to find Θ, if the log-likelihood improves, $Q = Q^*$; otherwise, continue;

In split and merge operations, the candidates have to be chosen in order to find the optimal one. The number of candidates affects the time complexity of the algorithm. Similarly, the criteria for merge and splitting also influence the time complexity. However, random swap will not increase time complexity since randomly picking up and changing position is a linear time operation with respect to model size. The time complexity of SMEM algorithm is $O(N^2 + M^5)$. If the number of component is large, the time complexity of SMEM will be a problem. Because of splitting and partial update, the time complexity of the Greedy EM is $O(M^2 N)$. In contrast to the SM based methods, the time complexity of RSEM is only $O(kMN)$, where k is the number of swapped iterations used in the algorithm.

3. EXPERIMENTAL RESULTS & APPLICATIONS

3.1. *Split and Merge EM vs. Random Swap EM*

We found out that SMEM algorithm is not compatible with the log-likelihood framework, which is also mentioned in [13]. In this case, a simple data set is generated according to the true parameters (Θ^*) shown in Table 1 to display the effectiveness of our algorithm. With the initial value Θ^0, the final results from EM, SMEM and RSEM are listed in the table. As we can see, the parameters estimated by RSEM are closer to the true parameters than those by other algorithms.

3.2. *Greedy EM vs. Random Swap EM*

As GEM changes the number of components, we set the maximum number allowed as the same as the true number of components. A comparison of log-likelihoods between conventional EM, GEM and RSEM is shown in Table 2. The data sets (Q1, Q2, S1, S4) are artificially generated with normal distribution (no. of components are 6,3,15,15 respectively). According to the mean and standard deviation value of the log-likelihood from 50 runs, the results indicate that RSEM has two significant digits' difference on likelihood for Q1,Q2 and S1 and at least 10% less standard deviation than the GEM.

3.3. *Image Segmentation*

We applied the proposed RSEM algorithm to color image segmentation on standard test images[1]. YUV color space performs better than RGB in our experiments, thus we display the segmentation results of "pepper" in YUV color space by EM, SMEM and RSEM algorithm in Fig.2. The number of components is set as 12, which also indicates the number of colors in the algorithms. As shown in the figure, EM is the fastest algorithm with the lowest log-likelihood value. RSEM is the compromise between EM and SMEM, and it has visibly better results than EM. We compared the number of segments obtained from the results of the three algorithms, where the

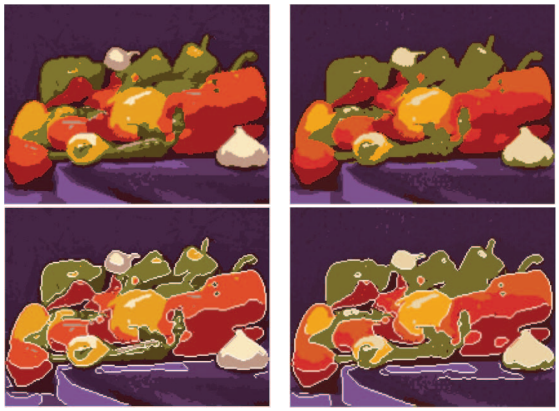
[1] $www.imageprocessingplace.com/root\_files\_V3/image\_databases.htm$

Table 2. Log-likelihood comparisons of RSEM, GEM and EM.

Data / Method	Q1		Q2		S1		S4	
	mean	std	mean	std	mean	std	mean	std
GEM	-6.874	0.0103	-3.598	0.0827	-26.308	0.4347	-26.76	0.19
RSEM	-6.795	0.0019	-3.422	0.0254	-26.117	0.0191	-26.38	0.026
EM	-6.873	0.0668	-3.572	0.013	-26.26	0.081	-26.377	0.033

segments are the objects in the image. RSEM has less segments than SMEM as shown in the figure, where more segments indicate unnecessary objects are detected in our case.

(a) Original Image (b) EM



(c) SMEM (d) RSEM

Fig. 2. Image segmentation results. (a). Original image and result of 80 segments by Canny operator (b). EM algorithm with 57 segments(CPU time: 2.31s, log-likelihood: -11.3162) (c). SMEM algorithm with 59 segments(CPU time: 24.97s, log-likelihood: -11.2564) (d). RSEM algorithm with 45 segments(CPU time: 8.55s, log-likelihood: -10.8137)

4. CONCLUSIONS

The proposed RSEM algorithm is capable of estimating the parameters better, and with randomly perturbing the configuration of the results from the EM algorithm, it overcomes existing problems of the EM algorithm with lower time complexity than other tested methods. Experiments conducted on different data sets and different methods demonstrated that the proposed algorithm made a clear improvement over the conventional EM algorithm. Meanwhile, the application in color image segmentation also gives promising results. We believe that the proposed method is widely applicable.

5. REFERENCES

[1] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of Royal Statistical Society B*, vol. 39, pp. 1–38, 1977.

[2] B. Zhang, C. Zhang, and X. Yi, "Competitive EM algorithm for finite mixture models," *Pattern Recognition*, vol. 31, no. 1, pp. 131–144, 2004.

[3] G. Biernacki, C. Celeux, and G. Govaert, "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models," *Computational Statistics and Data Analysis*, vol. 41, pp. 561–575, 2003.

[4] N. Udea, R. Nakano, Z. Gharhamani, and G. Hinton, "SMEM algorithm for mixture models," *Neural Computation*, vol. 12, pp. 2109–2128, 2000.

[5] N. Ueda and R. Nakano, "Deterministic annealing EM algorithm," *Neural Networks*, vol. 11, no. 2, pp. 271–282, 1998.

[6] Z.H. Zhang, C.B. Chen, J. Sun, and K.L. Chan, "EM algorithms for gaussian mixtures with split-and-merge operation," *Pattern Recognition*, vol. 36, no. 9, pp. 1973–1983, 2003.

[7] J.J. Verbeek, N. Vlassis, and B. Krose, "Efficient greedy learning of gaussian mixture models," *Neural Computation*, vol. 15, no. 12, pp. 469–485, 2003.

[8] G.J. Mclachlan and T. Krishnan, *The EM algorithm and extensions*, John Wiley - Sons, 1996.

[9] G. Celeux, D. Chauveau, and J. Diebolt, "On stochastic versions of the EM algorithm," INRIA technical report no.2514, 1995.

[10] C.F. Jeff Wu, "On the convergence properties of the EM algorithm," *Ann. Statist*, vol. 11, no. 1, pp. 95–103, 1983.

[11] T. Kaukoranta, P. Franti, and O. Nevalainen, "Iterative split-and-merge algorithm for vector quantization codebook generation," *Optical Engineering*, vol. 37, no. 10, pp. 2726–2732, 1998.

[12] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," *Proceedings of the 17th Int. Conf. on Machine Learning*, pp. 727–734, 2000.

[13] A. Minagawa, N. Tagawa, and T. Tanaka, "SMEM algorithm is not fully compatible with maximum-likelihood framework," *Neural Computation*, vol. 14, no. 6, pp. 1261–1266, 2002.