Springer Theses
Recognizing Outstanding Ph.D. Research

Junjie Wu

Advances in K-means Clustering

A Data Mining Thinking

Springer

Junjie Wu

Advances in K-means Clustering

A Data Mining Thinking

Doctoral Thesis accepted by
Tsinghua University, China, with substantial expansions

Springer

Author
Prof. Dr. Junjie Wu
Department of Information Systems
School of Economics
and Management
Beihang University
100191 Beijing
China

Supervisor
Prof. Jian Chen
Department of Management Science
and Engineering
School of Economics and Management
Tsinghua University
100084 Beijing
China

ISSN 2190-5053 ISSN 2190-5061 (electronic)
ISBN 978-3-642-29806-6 ISBN 978-3-642-29807-3 (eBook)
DOI 10.1007/978-3-642-29807-3
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012939113

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Parts of this book have been published in the following articles

Wu, J., Xiong, H., Liu, C., Chen, J.: A generalization of distance functions for fuzzy c-means clustering with centroids of arithmetic means. IEEE Transactions on Fuzzy Systems, forthcoming (2012) (Reproduced with Permission)

Xiong, H., Wu, J., Chen, J.: K-means clustering versus validation measures: A data distribution perspective. IEEE Transactions on Systems, Man, and Cybernetics—Part B 39(2): 318–331 (2009) (Reproduced with Permission)

Wu, J., Xiong, H., Chen, J.: Adapting the right measures for k-means clustering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 877–885 (2009) (Reproduced with Permission)

Wu, J., Xiong, H., Chen, J.: COG: Local decomposition for rare class analysis. Data Mining and Knowledge Discovery 20(2): 191–220 (2010) (Reproduced with Permission)

To my dearest wife Maggie, and our lovely son William

Supervisor's Foreword

In recent years people have witnessed the fast growth of a young discipline: *data mining*. It aims to find unusual and valuable patterns automatically from huge volumes of data collected from various research and application domains. As a typical inter-discipline, data mining draws work from many well-established fields such as database, machine learning, and statistics, and is grounded in some fundamental techniques such as optimization and visualization. Nevertheless, data mining has successfully found its own way by focusing on real-life data with very challenging characteristics. Mining large-scale data, high-dimensional data, highly imbalanced data, stream data, graph data, multimedia data, etc., have become one exciting topic after another in data mining. A clear trend is, with increasing popularity of Web 2.0 applications, data mining is being advanced to build the next-generation recommender systems, and to explore the abundant knowledge inside the huge online social networks. Indeed, it has become one of the leading forces that direct the progress of *business intelligence*, a field and a market full of imagination.

This book focuses on one of the core topics of data mining: cluster analysis. In particular, it provides some recent advances in the theories, algorithms, and applications of K-means clustering, one of the oldest yet most widely used algorithms for clustering analysis. From the theoretical perspective, this book highlights the negative uniform effect of K-means in clustering class-imbalanced data, and generalizes the distance functions suitable for K-means clustering to the notion of point-to-centroid distance. From the algorithmic perspective, this book proposes the novel SAIL algorithm and its variants to address the zero-value dilemma of information-theoretic K-means clustering on high-dimensional sparse data. Finally, from the applicative perspective, this book discusses how to select the suitable external measures for K-means clustering validation, and explores how to make innovative use of K-means for other important learning tasks, such as rare class analysis and consensus clustering. Most of the preliminary works of this book have been published in the proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), and IEEE International Conference on Data Mining (ICDM), which indicates a strong data-mining thinking of the research in the book. This book is also heavily based on Dr. Wu's Doctoral Thesis completed in Research Center for Contemporary Management (RCCM), Key Research Institute of Humanities and Social Sciences at Universities, Tsinghua University, which won the award of National Excellent Doctoral Dissertation of China in 2010, but with a substantial expansion based on his follow-up research. In general, this book brings together the recent research efforts of Dr. Wu in the cluster analysis field.

I believe both the researchers and practitioners in the cluster analysis field and the broader data mining area can benefit from reading this book. Moreover, this book shows the research track of Dr. Wu from a Ph.D. student to a professor, which may be of interest particularly to new Ph.D. students.

I want to compliment Dr. Wu for having written such an outstanding book for the data mining community.

Tsinghua University,
China, March 2012

Jian Chen
Research Center for Contemporary Management

Acknowledgments

This book was partially supported by the National Natural Science Foundation of China (NSFC) under Grants 70901002, 71171007, 70890080, 70890082, 71028002, and 71031001, the Doctoral Fund of Ministry of Education of China under Grant 20091102120014, a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (granted to Dr. Junjie Wu), and the Program for New Century Excellent Talents in University (granted to Dr. Junjie Wu).

The author would also like to thank the supports from various research centers, including: (1) Beijing Key Laboratory of Emergency Support Simulation Technologies for City Operations, Beihang University; (2) Research Center for Contemporary Management, Key Research Institute of Humanities and Social Sciences at Universities, Tsinghua University; (3) Jiangsu Provincial Key Laboratory of E-Business.

Some internal funds of Beihang University also provided important supports to this book, including: (1) The publication fund for graduate students' English textbooks; (2) The high-quality curriculum construction project for "Advanced Applied Statistics" (for graduate students); (3) The high-quality curriculum construction project for "Decision Support and Business Intelligence" (for undergraduate students).

Contents

Chapter 1
Cluster Analysis and K-means Clustering: An Introduction

1.1 The Emergence of Data Mining

The phrase "data mining" was termed in the late eighties of the last century, which describes the activity that attempts to extract *interesting patterns* from data. Since then, data mining and knowledge discovery has become one of the hottest topics in both academia and industry. It provides valuable business and scientific intelligence hidden in a large amount of historical data.

From a research perspective, the scope of data mining has gone far beyond the database area. A great many of researchers from various fields, e.g. computer science, management science, statistics, biology, and geography, have made great contributions to the prosperity of data mining research. Some top annual academic conferences held specifically for data mining, such as KDD (ACM SIGKDD International Conference on Knowledge Discovery and Data Mining),[1] ICDM (IEEE International Conference on Data Mining),[2] and SDM (SIAM International Conference on Data Mining),[3] have become the main forums and prestigious brands that lead the trend of data mining research, and have a farreaching influence on big sharks such as Google, Microsoft, and Facebook in industry. Many top conferences in different research fields are now open for the submission of data mining papers. Some A^+ journals in management field, such as Management Science, Information Systems Research, and MIS Quarterly, have also published business intelligence papers based on data mining techniques. These facts clearly illustrate that data mining as a young discipline is fast penetrating into other well-established disciplines. Indeed, data mining is such a hot topic that it has even become an "obscured" buzzword misused in many related fields to show the advanced characteristic of the research in those fields.

[1] http://www.kdd.org/.

[2] http://www.cs.uvm.edu/~icdm/.

[3] http://www.informatik.uni-trier.de/~ley/db/conf/sdm/.

DOI: 10.1007/978-3-642-29807-3_1, © Springer-Verlag Berlin Heidelberg 2012

From an application perspective, data mining has become a powerful tool for extracting useful information from tons of commercial and engineering data. The driving force behind this trend is the explosive growth of data from various application domains, plus the much more enhanced storing and computing capacities of IT infrastructures at lower prices. As an obvious inter-discipline, data mining discriminates itself from machine learning and statistics in placing ever more emphasis on data characteristics and being more solution-oriented. For instance, data mining has been widely used in business area for a number of applications, such as customer segmentation and profiling, shelf layout arrangement, financial-asset price prediction, and credit-card fraud detection, which greatly boost the concept of business intelligence. In the Internet world, data mining enables a series of interesting innovations, such as web document clustering, click-through data analysis, opinion mining, social network analysis, online product/service/information recommendation, and location-based mobile recommendation, some of which even show appealing commercial prospects. There are still many applicative cases of data mining in diverse domains, which will not be covered any more. An interesting phenomenon is, to gain the first-mover advantage in the potentially huge business intelligence market, many database and statistical software companies have integrated the data mining module into their products, e.g. SAS Enterprise Miner, SPSS Modeler, Oracle Data Mining, and SAP Business Object. This also helps to build complete product lines for these companies, and makes the whole decision process based on these products transparent to the high-end users.

1.2 Cluster Analysis: A Brief Overview

As a young but huge discipline, data mining cannot be fully covered by the limited pages in a monograph. This book focuses on one of the core topics of data mining: cluster analysis. Cluster analysis provides insight into the data by dividing the objects into groups (clusters) of objects, such that objects in a cluster are more similar to each other than to objects in other clusters [48]. As it does not use external information such as class labels, cluster analysis is also called unsupervised learning in some traditional fields such as machine learning [70] and pattern recognition [33].

In general, there are two purposes for using cluster analysis: understanding and utility [87]. Clustering for understanding is to employ cluster analysis for automatically finding conceptually meaningful groups of objects that share common characteristics. It plays an important role in helping people to analyze, describe and utilize the valuable information hidden in the groups. Clustering for utility attempts to abstract the prototypes or the representative objects from individual objects in the same clusters. These prototypes/objects then serve as the basis of a number of data processing techniques such as summarization, compression, and nearest-neighbor finding.

Cluster analysis has long played an important role in a wide variety of application domains such as business intelligence, psychology and social science, information retrieval, pattern classification, and bioinformatics. Some interesting examples are as follows:

- **Market research.** Cluster analysis has become the "killer application" in one of the core business tasks: marketing. It has been widely used for large-scale customer segmentation and profiling, which help to locate targeted customers, design the 4P (product, price, place, promotion) strategies, and implement the effective customer relationship management (CRM) [12, 13].
- **Web browsing.** As the world we live has entered the Web 2.0 era, information overload has become a top challenge that prevents people from acquiring useful information in a fast and accurate way. Cluster analysis can help to automatically categorize web documents into a concept hierarchy, and therefore provide better browsing experience to web users [43].
- **Image indexing.** In the online environment, images pose problems of access and retrieval more complicated than those of text documents. As a promising method, cluster analysis can help to group images featured by the bag-of-features (BOF) model, and therefore becomes a choice for large-scale image indexing [97].
- **Recommender systems.** Recent year have witnessed an increasing interest in developing recommender systems for online product recommendation or location-based services. As one of the most successful approaches to build recommender systems, collaborative filtering (CF) technique uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users [86]. One of the fundamental tools of CF, is right the clustering technique.
- **Community Detection.** Detecting clusters or communities in real-world graphs such as large social networks, web graphs, and biological networks, is a problem of considerable interests that has received a great deal of attention [58]. A range of detection methods have been proposed in the literature, most of which are borrowed from the broader cluster analysis field.

The above applications clearly illustrate that clustering techniques are playing a vital role in various exciting fields. Indeed, cluster analysis is always valuable for the exploration of unknown data emerging from real-life applications. That is the fundamental reason why cluster analysis is invariably so important.

1.2.1 Clustering Algorithms

The earliest research on cluster analysis can be traced back to 1894, when Karl Pearson used the moment matching method to determine the mixture parameters of two single-variable components [78]. Since then, tremendous research efforts have been devoted to designing new clustering algorithms for cluster analysis. It has been pointed out by Milligan [68] that the difficulties of cluster analysis lie in the following three aspects: (1) Clustering is essentially an inexhaustible combinatorial problem; (2) There exist no widely accepted theories for clustering; (3) The definition of a cluster seems to be a bit "arbitrary", which is determined by the data characteristics and the understandings of users. These three points well illustrate why there are so many clustering algorithms proposed in the literature, and why it is valuable to formulate the clustering problems as optimization problems which can be solved by some heuristics.

In what follows, we categorize the clustering algorithms into various types, and introduce some examples to illustrate the distinct properties of algorithms in different categories. This part has been most heavily influenced by the books written by Tan et al. [87] and Jain and Dubes [48]. Note that we have no intention of making this part as a comprehensive overview of clustering algorithms. Readers with this interest can refer to the review papers written by Jain et al. [49], Berkhin [11], and Xu and Wunsch [96]. Some books that may also be of interest include those written by Anderberg [3], Kaufman and Rousseeuw [53], Mirkin [69], etc. A paper by Kleinberg [55] provides some in-depth discussions on the clustering theories.

Prototype-Based Algorithms. This kind of algorithms learns a prototype for each cluster, and forms clusters by data objects around the prototypes. For some algorithms such as the well-known K-means [63] and Fuzzy *c*-Means (FCM) [14], the prototype of a cluster is a centroid, and the clusters tend to be globular. Self-Organizing Map (SOM) [56], a variant of artificial neural networks, is another representative prototype-based algorithm. It uses a neighborhood function to preserve the topological properties of data objects, and the weights of the whole network will then be trained via a competitive process. Being different from the above algorithms, Mixture Model (MM) [65] uses a probability distribution function to characterize the prototype, the unknown parameters of which are usually estimated by the Maximum Likelihood Estimation (MLE) method [15].

Density-Based Algorithms. This kind of algorithms takes a cluster as a dense region of data objects that is surrounded by regions of low densities. They are often employed when the clusters are irregular or intertwined, or when noise and outliers are present. DBSCAN [34] and DENCLUE [46] are two representative density-based algorithms. DBSCAN divides data objects into core points, border points and noise, respectively, based on the Euclidean density [87], and then finds the clusters naturally. DENCLUE defines a probability density function based on the kernel function of each data object, and then finds the clusters by detecting the variance of densities. When it comes to data in high dimensionality, the density notion is valid only in subspaces of features, which motivates the subspace clustering. For instance, CLIQUE [1], a grid-based algorithm, separates the feature space into grid units, and finds dense regions in subspaces. A good review of subspace clustering can be found in [77].

Graph-Based Algorithms. If we regard data objects as nodes, and the distance between two objects as the weight of the edge connecting the two nodes, the data can be represented as a graph, and a cluster can be defined as a connected subgraph. The well-known agglomerative hierarchical clustering algorithms (AHC) [87], which merge the nearest two nodes/groups in one round until all nodes are connected, can be regarded as a graph-based algorithm to some extent. The Jarvis-Patrick algorithm (JP) [50] is a typical graph-based algorithm that defines the shared nearest-neighbors for each data object, and then sparsifies the graph to obtain the clusters. In recent years, spectral clustering becomes an important topic in this area, in which data can be represented by various types of graphs, and linear algebra is then used to solve the optimization problems defined on the graphs. Many spectral clustering algorithms have been proposed in the literature, such as Normalized Cuts [82] and MinMaxCut [31]. Readers with interests can refer to [74] and [62] for more details.

Hybrid Algorithms. Hybrid algorithms, which use two or more clustering algorithms in combination, are proposed in order to overcome the shortcomings of single clustering algorithms. Chameleon [51] is a typical hybrid algorithm, which firstly uses a graph-based algorithm to separate data into many small components, and then employs a special AHC to get the final clusters. In this way, bizarre clusters can be discovered. FPHGP [16, 43] is another interesting hybrid algorithm, which uses association analysis to find frequent patterns [2] and builds a data graph upon the patterns, and then applies a hypergraph partitioning algorithm [52] to partition the graph into clusters. Experimental results show that FPHGP performs excellently for web document data.

Algorithm-Independent Methods. Consensus clustering [72, 84], also called clustering aggregation or cluster ensemble, runs on the clustering results of basic clustering algorithms rather than the original data. Given a set of basic partitionings of data, consensus clustering aims to find a single partitioning that matches every basic partitioning as closely as possible. It has been recognized that consensus clustering has merits in generating better clusterings, finding bizarre clusters, handling noise and outliers, and integrating partitionings of distributed or even inconsistent data [75]. Typical consensus clustering algorithms include the graph-based algorithms such as CPSA, HGPA and MCLA[84], the co-association matrix-based methods [36], and the prototype-based clustering methods [89, 90]. Some methods that employ meta-heuristics also show competitive results but at much higher computational costs [60].

1.2.2 Cluster Validity

Cluster validity, or clustering evaluation, is a necessary but challenging task in cluster analysis. It is formally defined as giving *objective* evaluations to clustering results in a *quantitative* way [48]. A key motivation of cluster validity is that almost every clustering algorithm will find clusters in a data set that even has no natural cluster structure. In this situation, a validation measure is in great need to tell us how well the clustering is. Indeed, cluster validity has become the core task of cluster analysis, for which a great number of validation measures have been proposed and carefully studied in the literature.

These validation measures are traditionally classified into the following two types: external indices and internal indices (including the relative indices) [41]. External indices measure the extent to which the clustering structure discovered by a clustering algorithm matches some given external structure, e.g. the structure defined by the class labels. In contrast, internal indices measure the goodness of a clustering structure without respect to external information. As internal measures often make latent assumptions on the formation of cluster structures, and usually have much higher computational complexity, more research in recent years prefers to use external measures for cluster validity, when the purpose is only to assess clustering algorithms and the class labels are available.

Considering that we have no intention of making this part as an extensive review of all validation measures, and only some external measures have been employed for cluster validity in the following chapters, we will focus on introducing some popular external measures here. Readers with a broader interest can refer to the review papers written by Halkidi et al. [41, 42], although discussions on how to properly use the measures are not presented adequately. The classic book written by Jain and Dubes [48] covers fewer measures, but some discussions are very interesting.

According to the different sources, we can further divide the external measures into three categories as follows:

Statistics-Based Measures. This type of measures, such as Rand index (R), Jaccard coefficient (J), Folks and Mallows index (FM), and Γ statistic (Γ) [48], originated from the statistical area quite a long time ago. They focus on examining the group membership of each object pair, which can be quantified by comparing two matrices: the Ideal Cluster Similarity Matrix (ICuSM) and the Ideal Class Similarity Matrix (ICaSM) [87]. ICuSM has a 1 in the ij-th entry if two objects i and j are clustered into a same cluster and a 0, otherwise. ICaSM is defined with respect to class labels, which has a 1 in the ij-th entry if objects i and j belong to a same class, and a 0 otherwise. Consider the entries in the upper triangular matrices (UTM) of ICuSM and ICaSM. Let f_{00} (f_{11}) denote the number of entry pairs that have 0 (1) in the corresponding positions of the two UTMs, and let f_{01} and f_{10} denote the numbers of entry pairs that have different values in the corresponding positions of the two UTMs. R, J, and FM can then be defined as: $R = \frac{f_{00}+f_{11}}{f_{00}+f_{10}+f_{01}+f_{11}}$, $J = \frac{f_{11}}{f_{10}+f_{01}+f_{11}}$, and $FM = \frac{f_{11}}{\sqrt{(f_{11}+f_{10})(f_{11}+f_{01})}}$. The definition of Γ is more straightforward by computing the correlation coefficient of the two UTMs. More details about these measures can be found in [48].

Information-Theoretic Measures. This type of measures is typically designed based on the concepts of information theory. For instance, the widely used Entropy measure (E) [98] assumes that the clustering quality is higher if the entropy of data objects in each cluster is smaller. Let $E_j = \sum_i p_{ij} \log p_{ij}$, where p_{ij} is the proportion of objects in cluster j that are from class i, n_j is the number of objects in cluster j, and $n = \sum_j n_j$. We then have $E = \sum_j \frac{n_j}{n} E_j$. The Mutual Information measure (MI) [85] and the Variation of Information measure (VI) [66, 67] are another two representative measures that evaluate the clustering results by comparing the information contained in class labels and cluster labels, respectively. As these measures have special advantages including clear concepts and simple computations, they become very popular in recent studies, even more popular than the long-standing statistics-based measures.

Classification-Based Measures. This type of measures evaluates clustering results from a classification perspective. The F-measure (F) is such an example, which was originally designed for validating the results of hierarchical clustering [57], but also used for partitional clustering in recent studies [83, 95]. Let p_{ij} denote the proportion of data objects in cluster j that are from class i (namely the precision of cluster j for objects of class i), and q_{ij} the proportion of data objects from class i that are assigned to cluster j (namely the recall of class i in cluster j) [79]. We then have the F-measure of class i as: $F_i = \max_j \frac{2p_{ij}q_{ij}}{p_{ij}+q_{ij}}$, and the overall F-measure of clustering results as: $F = \sum_i \frac{n_i}{n} F_i$, where n_i is the number of objects of class i, and $n = \sum_i n_i$. Another representative measure is the Classification Error (ε), which tries to map each class to a different cluster so as to minimize the total misclassification rate. Details of ε can be found in [21].

Sometimes we may want to compare the clustering results of different data sets. In this case, we should normalize the validation measures into a value range of about [0,1] or [-1,+1] before using them. However, it is surprising that only a few research has addressed the issue of measure normalization in the literature, including [48] for Rand index, [66] for Variation of Information, and [30] for Mutual Information. Among these studies, two methods are often used for measure normalization, i.e. the expected-value method [48] and the extreme-value method [61], which are both based on the assumption of the multivariate hypergeometric distribution (MHD) [22] of clustering results. The difficulty lies in the computation of the expected values or the min/max values of the measures, subjecting to MHD. A thorough study of measure normalization has been provided in Chap. 5, and we therefore will not go into the details here.

1.3 K-means Clustering: An Ageless Algorithm

In this book, we focus on K-means clustering, one of the oldest and most widely used clustering algorithms. The research on K-means can be traced back to the middle of the last century, conducted by numerous researchers across different disciplines, most notably Lloyd (1957, 1982) [59], Forgey (1965) [35], Friedman and Rubin (1967) [37], and MacQueen (1967) [63]. Jain and Dubes (1988) provides a detailed history of K-means along with descriptions of several variations [48]. Gray and Neuhoff (1998) put K-means in the larger context of hill-climbing algorithms [40].

In a nutshell, K-means is a prototype-based, simple partitional clustering algorithm that attempts to find K non-overlapping clusters. These clusters are represented by their centroids (a cluster centroid is typically the mean of the points in that cluster). The clustering process of K-means is as follows. First, K initial centroids are selected, where K is specified by the user and indicates the desired number of clusters. Every point in the data is then assigned to the closest centroid, and each collection of points assigned to a centroid forms a cluster. The centroid of each cluster is then updated based on the points assigned to that cluster. This process is repeated until no point changes clusters.

It is beneficial to delve into the mathematics behind K-means. Suppose $\mathcal{D} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ is the data set to be clustered. K-means can be expressed by an objective function that depends on the proximities of the data points to the cluster centroids as follows:

$$\min_{\{m_k\},1\leq k\leq K} \sum_{k=1}^{K} \sum_{\boldsymbol{x}\in C_k} \pi_{\boldsymbol{x}} \mathrm{dist}(\boldsymbol{x}, \boldsymbol{m}_k), \tag{1.1}$$

where $\pi_{\boldsymbol{x}}$ is the weight of $\boldsymbol{x}$, n_k is the number of data objects assigned to cluster C_k, $\boldsymbol{m}_k = \sum_{\boldsymbol{x}\in C_k} \frac{\pi_{\boldsymbol{x}}\boldsymbol{x}}{n_k}$ is the centroid of cluster C_k, K is the number of clusters set by the user, and the function "dist" computes the distance between object $\boldsymbol{x}$ and centroid $\boldsymbol{m}_k$, $1 \leq k \leq K$. While the selection of the distance function is optional, the squared Euclidean distance, i.e. $\parallel \boldsymbol{x} - \boldsymbol{m} \parallel^2$, has been most widely used in both research and practice. The iteration process introduced in the previous paragraph is indeed a gradient-descent alternating optimization method that helps to solve Eq. (1.1), although often converges to a local minima or a saddle point.

Considering that there are numerous clustering algorithms proposed in the literature, it may be argued that why this book is focused on the "old" K-means clustering. Let us understand this from the following two perspectives. First, K-means has some distinct advantages compared with other clustering algorithms. That is, K-means is very simple and robust, highly efficient, and can be used for a wide variety of data types. Indeed, it has been ranked the second among the top-10 data mining algorithms in [93], and has become the defacto benchmark method for newly proposed methods. Moreover, K-means as an optimization problem still has some theoretical challenges, e.g. the distance generalization problem studied in Chap. 3. The emerging data with complicated properties, such as large-scale, high-dimensionality, and class imbalance, also require to adapt the classic K-means to different challenging scenarios, which in turn rejuvenates K-means. Some disadvantages of K-means, such as performing poorly for non-globular clusters, and being sensitive to outliers, are often dominated by the advantages, and partially corrected by the proposed new variants.

In what follows, we review some recent research on K-means from both the theoretical perspective and the data-driven perspective. Note that we here do not expect to coverage all the works of K-means, but would rather introduce some works that relate to the main themes of this book.

1.3.1 Theoretical Research on K-means

In general, the theoretical progress on K-means clustering lies in the following three aspects:

Model Generalization. The Expectation-Maximization (EM) [26] algorithm-based Mixture Model (MM) has long been regarded as the generalized form of K-means for taking the similar alternating optimization heuristic [65]. Mitchell (1997) gave the details of how to derive squared Euclidean distance-based K-means from the Gaussian distribution-based MM, which unveil the relationship between K-means and MM [70]. Banerjee et al. (2005) studied the von Mises-Fisher distribution-based MM, and demonstrated that under some assumptions this model could reduce to K-means with cosine similarity, i.e. the spherical K-means [4]. Zhong and Ghosh (2004) proposed the Model-Based Clustering (MBC) algorithm [99], which unifies

MM and K-means via the introduction of the deterministic annealing technique. That is, MBC reduces to MM when the temperature $T = 1$, and to K-means when $T = 0$; As T decreases from 1 to 0, MBC gradually changes from allowing soft assignment to only allowing hard assignment of data objects.

Search Optimization. One weakness of K-means is that the iteration process may probably converge to a local minimum or even a saddle point. The traditional search strategies, i.e. the batch mode and the local mode, cannot avoid this problem, although some research has pointed out that using the local search immediately after the batch search may improve the clustering quality of K-means. The "kmeans" function included in MATLAB v7.1 [64] implemented this hybrid strategy. Dhillon et al. (2002) proposed a "first variation" search strategy for spherical K-means, which shares some common grounds with the hybrid strategy [27]. Steinbach et al. (2000) proposed a simple bisecting scheme for K-means clustering, which selects and divides a cluster into two sub-clusters in each iteration [83]. Empirical results demonstrate the effectiveness of bisecting K-means in improving the clustering quality of spherical K-means, and solving the random initialization problem. Some metaheuristics, such as deterministic annealing [80, 81] and variable neighborhood search [45, 71], can also help to find better local minima for K-means.

Distance Design. The distance function is one of the key factors that influence the performance of K-means. Dhillon el al. (2003) proposed an information-theoretic co-clustering algorithm based on the distance of Kullback-Leibler divergence (or KL-divergence for short) [30] originated from the information theory [23]. Empirical results demonstrate that the co-clustering algorithm improves the clustering efficiency of K-means using KL-divergence (or Info-Kmeans for short), and has higher clustering quality than traditional Info-Kmeans on some text data. Banerjee et al. (2005) studied the generalization issue of K-means clustering by using the Bregman divergence [19], which is actually a family of distances including the well-known squared Euclidean distance, KL-divergence, Itakura-Saito distance [5], and so on. To find clearer boundaries between different clusters, kernel methods have also been introduced to K-means clustering [28], and the concept of distance has therefore been greatly expanded by the kernel functions.

1.3.2 Data-Driven Research on K-means

As the emergence of big data in various research and industrial domains in recent years, the traditional K-means algorithm faces great challenges stemming from the diverse and complicated data factors, such as the high dimensionality, the data streaming, the existence of noise and outliers, and so on. In what follows, we focus on some data-driven advances in K-means clustering.

K-means Clustering for High-Dimensional Data. With the prosperity of information retrieval and bioinformatics, high-dimensional text data and micro-array data have become the challenges to clustering. Numerous studies have pointed out that K-means with the squared Euclidean distance is not suitable for high-dimensional data clustering because of the "curse of dimensionality" [8].

One way to solve this problem is to use alternative distance functions. Steinbach et al. (2000) used the cosine similarity as the distance function to compare the performance of K-means, bisecting K-means, and UPGMA on high-dimensional text data [83]. Experimental results evaluated by the entropy measure demonstrate that while K-means is superior to UPGMA, bisecting K-means has the best performance. Zhao and Karypis (2004) compared the performance of K-means using different types of objective functions on text data, where the cosine similarity was again employed for the distance computation [98]. Zhong and Ghosh (2005) compared the performance of the mixture model using different probability distributions [100]. Experimental results demonstrate the advantage of the von Mises-Fisher distribution. As this distribution corresponds to the cosine similarity in K-means [4], these results further justify the superiority of the cosine similarity for K-means clustering of high-dimensional data.

Another way to tackle this problem is to employ dimension reduction for high-dimensional data. Apart from the traditional methods such as the Principal Component Analysis, Multidimensional Scaling, and Singular Value Decomposition [54], some new methods particularly suitable for text data have been proposed in the literature, e.g. Term-Frequency-Inverse-Document-Frequency (TFIDF), Latent Semantic Indexing (LSI), Random Projection (RP), and Independent Component Analysis (ICA). A comparative study of these methods was given in [88], which revealed the following ranking: ICA $\succ$ LSI $\succ$ TFIDF $\succ$ RP. In particular, ICA and LSI show significant advantages on improving the performance of K-means clustering. Dhillon et al. (2003) used Info-Kmeans to cluster term features for dimension reduction [29]. Experimental results show that their method can improve the classification accuracy of the Naïve Bayes (NB) classifier [44] and the Support Vector Machines (SVMs) [24, 91].

K-means Clustering on Data Stream. Data stream clustering is a very challenging task because of the distinct properties of stream data: rapid and continuous arrival online, need for rapid response, potential boundless volume, etc. [38]. Being very simple and highly efficient, K-means naturally becomes the first choice for data stream clustering. We here highlight some representative works. Domingos and Hulten (2001) employed the Hoeffding inequality [9] for the modification of K-means clustering, and obtained approximate cluster centroids in data streams, with a probability-guaranteed error bound [32]. Ordonez (2003) proposed three algorithms: online K-means, scalable K-means, and incremental K-means, for binary data stream clustering [76]. These algorithms use several sufficient statistics and carefully manipulate the computation of the sparse matrix to improve the clustering quality. Experimental results indicate that the incremental K-means algorithm performs the best. Beringer and Hullermeier (2005) studied the clustering of multiple data streams [10]. The sliding-window technique and the discrete Fourier transformation technique were employed to extract the signals in data streams, which were then clustered by K-means algorithm using the squared Euclidean distance.

Semi-Supervised K-means Clustering. In recent years, more and more researchers recognize that clustering quality can be effectively improved by using *partially available* external information, e.g. the class labels or the pair-wise constraints of data. Semi-supervised K-means clustering has therefore become the focus of a great deal of research. For instance, Wagstaff et al. (2001) proposed the COP-KMeans algorithm for semi-supervised clustering of data with two types of pair-wise constraints: must-link and cannot-link [92]. The problem of COP-KMeans is that it cannot handle inconsistent constraints. Basu et al. (2002) proposed two algorithms, i.e. SEEDED-KMeans and CONSTRAINED-KMeans, for semi-supervised clustering using partial label information [6]. Both the two algorithms employ seed clustering for initial centroids, but only CONSTRAINED-KMeans reassigns the data objects outside the seed set during the iteration process. Experimental results demonstrate the superiority of the two methods to COP-KMeans, and SEEDED-KMeans shows good robustness to noise. Basu et al. (2004) further proposed the HMRF-KMeans algorithm that based on the hidden Markov random fields for pair-wise constraints [7], and the experimental results show that HMRF-Kmeans is significantly better than K-means. Davidson and Ravi (2005) proved that to satisfy all the pair-wise constraints in K-means is NP-complete, and thus only satisfied partial constraints to speed up the constrained K-means clustering [25]. They also proposed δ- and ε-constraints in the cluster level to improve the clustering quality.

K-means Clustering on Data with Other Characteristics. Other data factors that may impact the performance of K-means including the scale of data, the existence of noise and outliers, and so on. For instance, Bradley et al. (1998) considered how to adapt K-means to the situation that the data could not be entirely loaded into the memory [17]. They also studied how to improve the scalability of the EM-based mixture model [18]. Some good reviews about the scalability of clustering algorithms can be found in [73] and [39]. Noise removal, often conducted before clustering, is very important for the success of K-means. Some new methods for noise removal include the well-known LOF [20] and the pattern-based HCleaner [94], and a good review of the traditional methods can be found in [47].

1.3.3 Discussions

In general, K-means has been widely studied in a great deal of research from both the optimization and the data perspectives. However, there still some important problems remain unsolve as follows.

First, few research has realized the impact of skewed data distribution (i.e. the imbalance of true cluster sizes) on K-means clustering. This is considered dangerous, because data imbalance is a universal situation in practice, and cluster validation measures may not have the ability to capture its impact to K-means. So we have the following problems:

Problem 1.1 How can skewed data distributions make impact on the performance of K-means clustering? What are the cluster validation measures that can identify this impact?

The answer to the above questions can provide a guidance for the proper use of K-means. This indeed motivates our studies on the uniform effect of K-means in Chap. 2, and the selection of validation measures for K-means in Chap. 5.

Second, although there have been some distance functions widely used for K-means clustering, their common grounds remain unclear. Therefore, it will be a theoretical contribution to provide a general framework for distance functions that are suitable for K-means clustering. So we have the following problems:

Problem 1.2 Is there a unified expression for all the distance functions that fit K-means clustering? What are the common grounds of these distance functions?

The answer to the above questions can establish a general framework for K-means clustering, and help to understand the essence of K-means. Indeed, these questions motivate our study on the generalization of distance functions in Chap. 3, and the answers help to derive a new variant of K-means in Chap. 4.

Finally, it is interesting to know the potential of K-means as a utility to improve the performance of other learning schemes. Recall that K-means has some distinct merits such as simplicity and high efficiency, which make it a good booster for this task. So we have the following problem:

Problem 1.3 Can we use K-means clustering to improve other learning tasks such as the supervised classification and the unsupervised ensemble clustering?

The answer to the above question can help to extend the applicability of K-means, and drive this ageless algorithm to new research frontiers. This indeed motivates our studies on rare class analysis in Chap. 6 and consensus clustering in Chap. 7.

1.4 Concluding Remarks

In this chapter, we present the motivations of this book. Specifically, we first highlight the exciting development of data mining and knowledge discovery in both academia and industry in recent years. We then focus on introducing the basic preliminaries and some interesting applications of cluster analysis, a core topic in data mining. Recent advances in K-means clustering, a most widely used clustering algorithm, are also introduced from a theoretical and a data-driven perspectives, respectively. Finally, we put forward three important problems remained unsolved in the research of K-means clustering, which indeed motivate the main themes of this book.

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 94–105 (1998)

2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
3. Anderberg, M.: Cluster Analysis for Applications. Academic Press, New York (1973)
4. Banerjee, A., Dhillon, I., Ghosh, J., Sra, S.: Clustering on the unit hypersphere using von mises-fisher distributions. J. Mach. Learn. Res. **6**, 1345–1382 (2005)
5. Banerjee, A., Merugu, S., Dhillon, I., Ghosh, J.: Clustering with bregman divergences. J. Mach. Learn. Res. **6**, 1705–1749 (2005)
6. Basu, S., Banerjee, A., Mooney, R.: Semi-supervised clustering by seeding. In: Proceedings of the Nineteenth International Conference on, Machine Learning, pp. 19–26 (2002)
7. Basu, S., Bilenko, M., Mooney, R.: A probabilistic framework for semi-supervised clustering. In: Proceedings of 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 59–68 (2004)
8. Bellman, R.E., Corporation, R.: Dynamic Programming. Princeton University Press, New Jersey (1957)
9. Bentkus, V.: On hoeffding's inequalities. Ann. Probab. **32**(2), 1650–1673 (2004)
10. Beringer, J., Hullermeier, E.: Online clustering of parallel data streams. Data Knowl. Eng. **58**(2), 180–204 (2005)
11. Berkhin, P.: Survey of clustering data mining techniques. Technical Report, Accrue Software, San Jose (2002)
12. Berry, M., Linoff, G.: Data Mining Techniques: For Marketing, Sales, and Customer Support. Wiley, New York (1997)
13. Berry, M., Linoff, G.: Matering Data Mining: The Art and Science of Customer Relationship Management. Wiley, New York (1999)
14. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algoritms. Plenum Press, New York (1981)
15. Bilmes, J.: A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report, ICSITR-97-021, International Computer Science Institute and U.C. Berkeley (1997)
16. Boley, D., Gini, M., Gross, R., Han, E., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Partitioning-based clustering for web document categorization. Decis. Support Syst. **27**(3), 329–341 (1999)
17. Bradley, P., Fayyad, U., Reina, C.: Scaling clustering algorithms to large databases. In: Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 9–15 (1998)
18. Bradley, P., Fayyad, U., Reina, C.: Scaling em (expectation maximization) clustering to large databases. Technical Report, MSR-TR-98-35, Microsoft Research (1999)
19. Bregman, L.: The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. USSR Comput. Math. Math. Phys. **7**, 200–217 (1967)
20. Breunig, M., Kriegel, H., Ng, R., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of 2000 ACM SIGMOD International Conference on Management of Data, pp. 93–104 (2000)
21. Brun, M., Sima, C., Hua, J., Lowey, J., Carroll, B., Suh, E., Dougherty, E.: Model-based evaluation of clustering validation measures. Pattern Recognit. **40**, 807–824 (2007)
22. Childs, A., Balakrishnan, N.: Some approximations to the multivariate hypergeometric distribution with applications to hypothesis testing. Comput. Stat. Data Anal. **35**(2), 137–154 (2000)
23. Cover, T., Thomas, J.: Elements of Information Theory, 2nd edn. Wiley-Interscience, Hoboken (2006)
24. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)
25. Davidson, I., Ravi, S.: Clustering under constraints: feasibility results and the k-means algorithm. In: Proceedings of the 2005 SIAM International Conference on Data Mining (2005)

26. Dempster, A., Laird, N., Rubin, D.: Maximum-likelihood from incomplete data via the em algorithm. J. Royal Stat. Soc. Ser. B **39**(1), 1–38 (1977)
27. Dhillon, I., Guan, Y., Kogan, J.: Iterative clustering of high dimensional text data augmented by local search. In: Proceedings of the 2002 IEEE International Conference on Data Mining, pp. 131–138 (2002)
28. Dhillon, I., Guan, Y., Kulis, B.: Kernel k-means: Spectral clustering and normalized cuts. In: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 551–556. New York (2004)
29. Dhillon, I., Mallela, S., Kumar, R.: A divisive information-theoretic feature clustering algorithm for text classification. J. Mach. Learn. Res. **3**, 1265–1287 (2003)
30. Dhillon, I., Mallela, S., Modha, D.: Information-theoretic co-clustering. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 89–98 (2003)
31. Ding, C., He, X., Zha, H., Gu, M., Simon, H.: A min-max cut for graph partitioning and data clustering. In: Proceedings of the 1st IEEE International Conference on Data Mining, pp. 107–114 (2001)
32. Domingos, P., Hulten, G.: A general method for scaling up machine learning algorithms and its application to clustering. In: Proceedings of the 18th International Conference on, Machine Learning, pp. 106–113 (2001)
33. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley-Interscience, New York (2000)
34. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
35. Forgy, E.: Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics **21**(3), 768–769 (1965)
36. Fred, A., Jain, A.: Combining multiple clusterings using evidence accumulation. IEEE Trans. Pattern Anal. Mach. Intell. **27**(6), 835–850 (2005)
37. Friedman, H., Rubin, J.: On some invariant criteria for grouping data. J. Am. Stat. Assoc. **62**, 1159–1178 (1967)
38. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. SIGMOD Rec. **34**(2), 18–26 (2005)
39. Ghosh, J.: Scalable clustering methods for data mining. In: Ye, N. (ed.) Handbook of Data Mining, pp. 247–277. Lawrence Ealbaum (2003)
40. Gray, R., Neuhoff, D.: Quantization. IEEE Trans. Info. Theory **44**(6), 2325–2384 (1998)
41. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Cluster validity methods: Part I. SIGMOD Rec. **31**(2), 40–45 (2002)
42. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Clustering validity checking methods: Part II. SIGMOD Rec. **31**(3), 19–27 (2002)
43. Han, E.H., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Webace: a web agent for document categorization and exploration. In: Proceedings of the 2nd International Conference on Autonomous Agents, pp. 408–415 (1998)
44. Hand, D., Yu, K.: Idiot's bayes—not so stupid after all? Int. Stat. Rev. **69**(3), 385–399 (2001)
45. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. Euro. J. Oper. Res. **130**, 449–467 (2001)
46. Hinneburg, A., Keim, D.: An efficient approach to clustering in large multimedia databases with noise. In: Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 58–65. AAAI Press, New York (1998)
47. Hodge, V., Austin, J.: A survey of outlier detection methodologies. Artif. Intell. Rev. **22**, 85–126 (2004)
48. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
49. Jain, A., Murty, M., Flynn, P.: Data clustering: A review. ACM Comput. Surv. **31**(3), 264–323 (1999)

50. Jarvis, R., Patrick, E.: Clusering using a similarity measure based on shared nearest neighbors. IEEE Trans. Comput. **C-22(11)**, 1025–1034 (1973)
51. Karypis, G., Han, E.H., Kumar, V.: Chameleon: a hierarchical clustering algorithm using dynamic modeling. IEEE Comput. **32**(8), 68–75 (1999)
52. Karypis, G., Kumar, V.: A fast and highly quality multilevel scheme for partitioning irregular graphs. SIAM J. Sc. Comput. **20**(1), 359–392 (1998)
53. Kaufman, L., Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley Series in Probability and Statistics. Wiley, New York (1990)
54. Kent, J., Bibby, J., Mardia, K.: Multivariate Analysis (Probability and Mathematical Statistics). Elsevier Limited, New York (2006)
55. Kleinberg, J.: An impossibility theorem for clustering. In: Proceedings of the 16th Annual Conference on Neural Information Processing Systems, pp. 9–14 (2002)
56. Kohonen, T., Huang, T., Schroeder, M.: Self-Organizing Maps. Springer,Heidelberg (2000)
57. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 16–22 (1999)
58. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th International Conference on, World Wide Web, pp. 631–640 (2010)
59. Lloyd, S.: Least squares quantization in pcm. IEEE Trans. Info. Theory **28**(2), 129–137 (1982)
60. Lu, Z., Peng, Y., Xiao, J.: From comparing clusterings to combining clusterings. In: Fox, D., Gomes, C. (eds.) Proceedings of the 23rd AAAI Conference on Artificial Intelligence, pp. 361–370. AAAI Press, Chicago (2008)
61. Luo, P., Xiong, H., Zhan, G., Wu, J., Shi, Z.: Information-theoretic distance measures for clustering validation: Generalization and normalization. IEEE Trans. Knowl. Data Eng. **21**(9), 1249–1262 (2009)
62. Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
63. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
64. MathWorks: K-means clustering in statistics toolbox. http://www.mathworks.com
65. McLachlan, G., Basford, K.: Mixture Models. Marcel Dekker, New York (2000)
66. Meila, M.: Comparing clusterings by the variation of information. In: Proceedings of the 16th Annual Conference on Computational Learning Theory, pp. 173–187 (2003)
67. Meila, M.: Comparing clusterings—an axiomatic view. In: Proceedings of the 22nd International Conference on, Machine learning, pp. 577–584 (2005)
68. Milligan, G.: Clustering validation: Results and implications for applied analyses. In: Arabie, P., Hubert, L., Soete, G. (eds.) Clustering and Classification, pp. 345–375. World Scientific, Singapore (1996)
69. Mirkin, B.: Mathematical Classification and Clustering. Kluwer Academic Press, Dordrecht (1996)
70. Mitchell, T.: Machine Learning. McGraw-Hill, Boston (1997)
71. Mladenovic, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
72. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Mach. Learn. **52**(1–2), 91–118 (2003)
73. Murtagh, F.: Clustering massive data sets. In: Abello, J., Pardalos, P.M., Resende, M.G. (eds.) Handbook of Massive Data Sets, pp. 501–543. Kluwer Academic Publishers, Norwell (2002)
74. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: analysis and an algorithm. In: Advances in Neural Information Processing Systems, pp. 849–856. MIT Press (2001)
75. Nguyen, N., Caruana, R.: Consensus clusterings. In: Proceedings of the 7th IEEE International Conference on Data Mining, pp. 607–612. Washington (2007)

76. Ordonez, C.: Clustering binary data streams with k-means. In: Proceedings of the SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2003)
77. Parsons, L., Haque, E., Liu, H.: Subspace clustering for high dimensional data: a review. SIGKDD Explor. **6**(1), 90–105 (2004)
78. Pearson, K.: Contributions to the mathematical theory of evolution. Philos. Trans. Royal Soc. Lond. **185**, 71–110 (1894)
79. Rijsbergen, C.: Information Retrieval, 2nd edn. Butterworths, London (1979)
80. Rose, K.: Deterministic annealing for clustering, compression, classification, regression and related optimization problems. Proc. IEEE **86**, 2210–2239 (1998)
81. Rose, K., Gurewitz, E., Fox, G.: A deterministic annealing approach to clustering. Pattern Recognit. Lett. **11**, 589–594 (1990)
82. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
83. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of the KDD Workshop on Text Mining (2000)
84. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining partitions. J. Mach. Learn. Res. **3**, 583–617 (2002)
85. Strehl, A., Ghosh, J., Mooney, R.: Impact of similarity measures on web-page clustering. In: Proceedings of the AAAI Workshop on AI for Web Search (2000)
86. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence 2009, Article ID 421,425, 19 pp (2009)
87. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2005)
88. Tang, B., Shepherd, M., Heywood, M., Luo, X.: Comparing dimension reduction techniques for document clustering. In: Proceedings of the Canadian Conference on, Artificial Intelligence, pp. 292–296 (2005)
89. Topchy, A., Jain, A., Punch, W.: Combining multiple weak clusterings. In: Proceedings of the 3rd IEEE International Conference on Data Mining, pp. 331–338. Melbourne (2003)
90. Topchy, A., Jain, A., Punch, W.: A mixture model for clustering ensembles. In: Proceedings of the 4th SIAM International Conference on Data Mining. Florida (2004)
91. Vapnik, V.: The Nature of Statistical Learning. Springer, New York (1995)
92. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proceedings of the Eighteenth International Conference on, Machine Learning, pp. 577–584 (2001)
93. Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2008)
94. Xiong, H., Pandey, G., Steinbach, M., Kumar, V.: Enhancing data analysis with noise removal. IEEE Trans. Knowl. Data Eng. **18**(3), 304–319 (2006)
95. Xiong, H., Wu, J., Chen, J.: K-means clustering versus validation measures: a data-distribution perspective. IEEE Trans. Syst. Man Cybern. Part B Cybern. **39**(2), 318–331 (2009)
96. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)
97. Yang, J., Yuz, K., Gongz, Y., Huang, T.: Linear spatial pyramid matching using sparse coding. In: Proceedings of the 2009 IEEE Conference on Computer Vision and, Pattern Recognition, pp. 1794–1801 (2009)
98. Zhao, Y., Karypis, G.: Criterion functions for document clustering: experiments and analysis. Mach. Learn. **55**(3), 311–331 (2004)
99. Zhong, S., Ghosh, J.: A unified framework for model-based clustering. J. Mach. Learn. Res. **4**(6), 1001–1037 (2004)
100. Zhong, S., Ghosh, J.: Generative model-based document clustering: a comparative study. Knowl. Inf. Syst. **8**(3), 374–384 (2005)

Chapter 2
The Uniform Effect of K-means Clustering

2.1 Introduction

This chapter studies the *uniform effect* of K-means clustering. As a well-known and widely used partitional clustering method, K-means has attracted great research interests for a very long time. Researchers have identified some data characteristics that may strongly impact the performance of K-means clustering, including the types and scales of data and attributes, the sparseness of data, and the noise and outliers in data [23]. However, further investigation is needed to unveil how data distributions can make impact on the performance of K-means clustering. Along this line, we provide an organized study of the effect of skewed data distributions on K-means clustering. The results can guide us for the better use of K-means. This is considered valuable, since K-means has been shown to perform as well as or better than a variety of other clustering techniques in text clustering, and has an appealing computational efficiency [17, 22, 29].

In this chapter, we first formally illustrate that K-means tends to produce clusters in relatively uniform sizes, even if the input data have varying true cluster sizes. Also, we show that some clustering validation measures, such as the entropy measure, may not capture the uniform effect of K-means, and thus provide misleading evaluations on the clustering results. To deal with this, the Coefficient of Variation (CV) [5] statistic is employed as a complement for cluster validation. That is, if the CV value of the cluster sizes has a significant change before and after the clustering, the clustering performance is considered questionable. However, the reverse is not true; that is, a minor change of the CV value does not necessarily indicate a good clustering result.

In addition, we have conducted extensive experiments on a number of real-world data sets, including text data, gene expression data, and UCI data, obtained from different application domains. Experimental results demonstrate that, for data with highly varying true cluster sizes (e.g. $CV > 1.0$), K-means tends to generate clusters in relatively uniform sizes ($CV < 1.0$). In contrast, for data sets with uniform true cluster sizes (e.g. $CV < 0.3$), K-means tends to generate clusters in varying sizes

($CV > 0.3$). In other words, for these two cases, the clustering performance of K-means is often poor.

The remainder of this chapter is organized as follows. Section 2.2 formally illustrates the uniform effect of K-means clustering. In Sect. 2.3, we illustrate the biased effect of the entropy measure. Section 2.4 shows experimental results. The related work is presented in Sect. 2.5, and we finally draw conclusions in Sect. 2.6.

2.2 The Uniform Effect of K-means Clustering

In this section, we mathematically formulate the fact that K-means clustering tends to produce clusters in uniform sizes, which is also called the *uniform effect* of K-means.

K-means is typically expressed by an objective function that depends on the proximities of the data points to the cluster centroids. Let $X = \{x_1, \ldots, x_n\}$ be the data, and $m_l = \sum_{x \in C_l} \frac{x}{n_l}$ be the centroid of cluster C_l, $1 \leq l \leq k$, where n_l is the number of data objects in cluster C_l, and k is the number of clusters. The objective function of K-means clustering is then formulated as the sum of squared errors as follows:

$$F_k = \sum_{l=1}^{k} \sum_{x \in C_l} \parallel x - m_l \parallel^2. \tag{2.1}$$

Let $d(C_p, C_q) = \sum_{x_i \in C_p} \sum_{x_j \in C_q} \parallel x_i - x_j \parallel^2$. We have the sum of all pair-wise distances of data objects within k clusters as follows:

$$D_k = \sum_{i=1}^{n} \sum_{j=1}^{n} \parallel x_i - x_j \parallel^2 = \sum_{l=1}^{k} d(C_l, C_l) + 2 \sum_{1 \leq i < j \leq k} d(C_i, C_j). \tag{2.2}$$

Note that D_k is a constant for a given data set regardless of k. We use the subscript k for the convenience of the mathematical induction. Also, $n = \sum_{l=1}^{k} n_l$ is the total number of objects in the data.

2.2.1 Case I: Two Clusters

Here, we first illustrate the uniform effect of K-means when the number of clusters is only two. We have,

$$D_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \parallel x_i - x_j \parallel^2 = d(C_1, C_1) + d(C_2, C_2) + 2d(C_1, C_2).$$

In this case, D_2 is also a constant, and $n = n_1 + n_2$ is the total number of data objects. If we substitute m_l in Eq. (2.1) by $\sum_{x \in C_l} \frac{x}{n_l}$, we have

$$F_2 = \frac{1}{2n_1} \sum_{x_i, x_j \in C_1} \| x_i - x_j \|^2 + \frac{1}{2n_2} \sum_{x_i, x_j \in C_2} \| x_i - x_j \|^2 = \frac{1}{2} \sum_{l=1}^{2} \frac{d(C_l, C_l)}{n_l}. \tag{2.3}$$

If we let

$$F_D^{(2)} = -n_1 n_2 \left[\frac{d(C_1, C_1)}{n_1^2} + \frac{d(C_2, C_2)}{n_2^2} - 2\frac{d(C_1, C_2)}{n_1 n_2} \right],$$

we thus have

$$F_2 = -\frac{F_D^{(2)}}{2n} + \frac{D_2}{2n}. \tag{2.4}$$

Furthermore, we can show that

$$\frac{2d(C_1, C_2)}{n_1 n_2} = \frac{d(C_1, C_1)}{n_1^2} + \frac{d(C_2, C_2)}{n_2^2} + 2\| m_1 - m_2 \|^2.$$

Therefore, we finally have

$$F_D^{(2)} = 2n_1 n_2 \| m_1 - m_2 \|^2.$$

Equation (2.4) indicates that the minimization of the K-means objective function F_2 is equivalent to the maximization of the distance function $F_D^{(2)}$. As $F_D^{(2)} > 0$ when m_1 is not equal to m_2, if we isolate the effect of $\| m_1 - m_2 \|^2$, the maximization of $F_D^{(2)}$ implies the maximization of $n_1 n_2$, which leads to $n_1 = n_2 = n/2$.

Discussion. In the above analysis, we have isolated the effect of two components: $\| m_1 - m_2 \|^2$ and $n_1 n_2$. For real-world data sets, the values of these two components are related to each other. Indeed, under certain circumstances, the goal of maximizing $n_1 n_2$ may contradict the goal of maximizing $\| m_1 - m_2 \|^2$. Figure 2.1 illustrates such a scenario when $n_1 n_2$ is dominated by $\| m_1 - m_2 \|^2$. In this example, we generate two true clusters, i.e. one `stick` cluster and one `circle` cluster, each of which contains 500 objects. If we apply K-means on these two data sets, we can have the clustering results in which 106 objects of the `stick` cluster are assigned to the `circle` cluster, as indicated by the green dots in the `stick` cluster. In this way, while the value of $n_1 n_2$ decreases a little bit, the value of $\| m_1 - m_2 \|^2$ increases more significantly, which finally leads to the decrease of the overall objective function value. This implies that, K-means will increase the variation of true cluster sizes slightly in this scenario. However, it is hard to further clarify the relationship between these two components in theory, as this relationship is affected by many factors, such

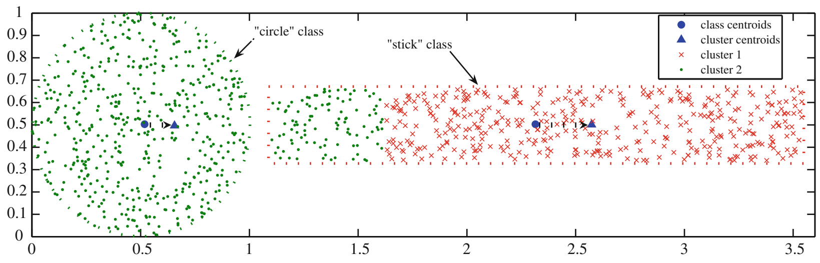


Fig. 2.1 Illustration of the violation of the uniform effect. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

as the shapes of clusters and the densities of data. As a complement, we present an extensive experimental study in Sect. 2.4 to provide a better understanding to this.

2.2.2 *Case II: Multiple Clusters*

Here, we consider the case that the number of clusters is greater than two. If we substitute m_l, the centroid of cluster C_l, in Eq. (2.1) by $\sum_{x \in C_l} \frac{x}{n_l}$, we have

$$F_k = \sum_{l=1}^{k} \left(\frac{1}{2n_l} \sum_{x_i, x_j \in C_l} \| x_i - x_j \|^2 \right) = \frac{1}{2} \sum_{l=1}^{k} \frac{d(C_l, C_l)}{n_l}. \tag{2.5}$$

We then decompose F_k by using the two lemmas as follows:

Lemma 2.1

$$D_k = \sum_{l=1}^{k} \frac{n}{n_l} d(C_l, C_l) + 2 \sum_{1 \le i < j \le k} n_i n_j \| m_i - m_j \|^2. \tag{2.6}$$

Proof We use the mathematical induction.

When $k = 1$, by Eq. (2.2), the left hand side of Eq. (2.6) is $d(C_1, C_1)$. The right hand side of Eq. (2.6) is also equal to $d(C_1, C_1)$, as there is no cross-cluster item. As a result, Lemma 2.1 holds.

When $k = 2$, by Eq. (2.2), to prove Eq. (2.6) is equivalent to prove the following equation:

$$2d(C_1, C_2) = \frac{n_2}{n_1} d(C_1, C_1) + \frac{n_1}{n_2} d(C_2, C_2) + 2n_1 n_2 \| m_1 - m_2 \|^2. \tag{2.7}$$

If we substitute $m_1 = \frac{\sum_{i=1}^{n_1} x_i}{n_1}$, $m_2 = \frac{\sum_{i=1}^{n_2} y_i}{n_2}$, and

$$d(C_1, C_1) = 2 \sum_{1 \le i < j \le n_1} \| x_i - x_j \|^2 = 2(n_1 - 1) \sum_{i=1}^{n_1} \| x_i \|^2 - 4 \sum_{1 \le i < j \le n_1} x_i x_j,$$

$$d(C_2, C_2) = 2 \sum_{1 \le i < j \le n_2} \| y_i - y_j \|^2 = 2(n_2 - 1) \sum_{i=1}^{n_2} \| y_i \|^2 - 4 \sum_{1 \le i < j \le n_2} y_i y_j,$$

$$d(C_1, C_2) = \sum_{1 \le i \le n_1} \sum_{1 \le j \le n_2} \| x_i - y_j \|^2 = 2n_2 \sum_{i=1}^{n_1} \| x_i \|^2 + 2n_1 \sum_{i=1}^{n_2} \| y_i \|^2 - 4 \sum_{1 \le i \le n_1} \sum_{1 \le j \le n_2} x_i y_j$$

into Eq. (2.7), we can show that the left hand side will be equal to the right hand side. Therefore, Lemma 2.1 also holds for $k = 2$.

Now we assume that Lemma 2.1 also holds when the cluster number is $k - 1$. Then for the case that the cluster number is k, we first define $D_{k-1}^{(i)}$ as the sum of squared pair-wise distances between data objects within $k - 1$ clusters selected from the total k clusters excluding cluster i. It is trivial to note that $D_{k-1}^{(i)} < D_k$, and they have relationship as follows:

$$D_k = D_{k-1}^{(p)} + d(C_p, C_p) + 2 \sum_{1 \le j \le k, j \ne p} d(C_p, C_j). \tag{2.8}$$

Note that Eq. (2.8) holds for any $p = 1, 2, \ldots, k$. So actually we have k equations. We sum up these k equations and get

$$kD_k = \sum_{p=1}^{k} D_{k-1}^{(p)} + \sum_{p=1}^{k} d(C_p, C_p) + 4 \sum_{1 \le i < j \le k} d(C_i, C_j). \tag{2.9}$$

As Eq. (2.6) holds for the case that the cluster number is $k - 1$, we have

$$D_{k-1}^{(p)} = \sum_{1 \le l \le k, l \ne p} \left[\frac{n - n_p}{n_l} d(C_l, C_l) \right] + 2 \sum_{1 \le i < j \le k}^{i,j \ne p} [n_i n_j \| m_i - m_j \|^2].$$

So the first part of the right hand side of Eq. (2.9) is

$$\sum_{p=1}^{k} D_{k-1}^{(p)} = (k-2) \left(\sum_{l=1}^{k} \frac{n}{n_l} d(C_l, C_l) + 2 \sum_{1 \le i < j \le k} n_i n_j \| m_i - m_j \|^2 \right) + \sum_{l=1}^{k} d(C_l, C_l). \tag{2.10}$$

Accordingly, we can further transform Eq. (2.9) into

$$kD_k = (k-2)\left(\sum_{l=1}^{k}\left[\frac{n}{n_l}d(C_l, C_l)\right] + 2\sum_{1\le i<j\le k}[n_i n_j \parallel m_i - m_j \parallel^2]\right) + 2\left[\sum_{l=1}^{k} d(C_l, C_l) + 2\sum_{1\le i<j\le k} d(C_i, C_j)\right]. \tag{2.11}$$

According to Eq. (2.2), we know that the second part of the right hand side of Eq. (2.11) is exactly $2D_k$. So we can finally have

$$D_k = \sum_{l=1}^{k}\frac{n}{n_l}d(C_l, C_l) + 2\sum_{1\le i<j\le k} n_i n_j \parallel m_i - m_j \parallel^2,$$

which implies that Lemma 2.1 also holds for the case that the cluster number is k. We complete the proof. □

Lemma 2.2 *Let*

$$F_D^{(k)} = D_k - 2nF_k. \tag{2.12}$$

Then

$$F_D^{(k)} = 2\sum_{1\le i<j\le k}[n_i n_j \parallel m_i - m_j \parallel^2]. \tag{2.13}$$

Proof If we substitute F_k in Eq. (2.5) and D_k in Eq. (2.6) into Eq. (2.12), we can know that Eq. (2.13) is true. □

Discussion. By Eq. (2.12), we know that the minimization of the K-means objective function F_k is equivalent to the maximization of the distance function $F_D^{(k)}$, where both D_k and n are constants for a given data set. For $F_D^{(k)}$ in Eq. (2.13), if we assume for all $1 \le i < j \le k$, $\parallel m_i - m_j \parallel^2$ are the same, i.e. all the pair-wise distances between two centroids are the same, then it is easy to show that the maximization of $F_D^{(k)}$ is equivalent to the uniform distribution of n_i, i.e. $n_1 = n_2 = \cdots = n_k = n/k$. Note that we have isolated the effect of two components: $\parallel m_i - m_j \parallel^2$ and $n_i n_j$ here to simplify the discussion. For real-world data sets, however, these two components are interactive.

2.3 The Relationship between K-means Clustering and the Entropy Measure

In this section, we study the relationship between K-means clustering and a widely used clustering validation measure: Entropy (E).

2.3.1 The Entropy Measure

Generally speaking, there are two types of clustering validation techniques [10, 13], which are based on external and internal criteria, respectively. Entropy is an external validation measure using the class labels of data as external information. It has been widely used for a number of K-means clustering applications [22, 29].

Entropy measures the purity of the clusters with respect to the given class labels. Thus, if each cluster consists of objects with a single class label, the entropy value is 0. However, as the class labels of objects in a cluster become more diverse, the entropy value increases.

To compute the entropy of a set of clusters, we first calculate the class distribution of the objects in each cluster. That is, for each cluster j, we compute p_{ij}, the probability of assigning an object of class i to cluster j. Given this class distribution, the entropy of cluster j is calculated as

$$E_j = -\sum_i p_{ij} log(p_{ij}),$$

where the sum is taken over all classes. The total entropy for a set of clusters is computed as the weighted sum of the entropies of all clusters:

$$E = \sum_{j=1}^{m} \frac{n_j}{n} E_j,$$

where n_j is the size of cluster j, m is the number of clusters, and n is the total number of data objects.

2.3.2 The Coefficient of Variation Measure

Before we describe the relationship between the entropy measure and K-means clustering, we first introduce the Coefficient of Variation (CV) statistic [5], which is a measure of the data dispersion. CV is defined as the ratio of the standard deviation to the mean. Given a set of data objects $X = \{x_1, x_2, \ldots, x_n\}$, we have

$$CV = \frac{s}{\bar{x}}, \tag{2.14}$$

where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \text{ and } s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}.$$

CV is a dimensionless number that allows comparing the variations of populations that have significantly different mean values. In general, the larger the CV value, the greater the variation in the data.

Recall that K-means clustering has a uniform effect (Sect. 2.2). CV can serve as a good indicator for the detection of the uniform effect. That is, if the CV value of the cluster sizes has a significant change after K-means clustering, we know that the uniform effect exists, and the clustering quality tends to be poor. However, it does not necessarily indicate a good clustering performance if the CV value of the cluster sizes only has a minor change after the clustering.

2.3.3 The Limitation of the Entropy Measure

In practice, we have observed that the entropy measure tends to favor clustering algorithms, such as K-means, which produce clusters with relatively uniform sizes. We call this the *biased effect* of the entropy measure. To illustrate this, we create a sample data set shown in Table 2.1. This data set consists of 42 documents belonging to five classes, i.e. five true clusters, whose CV value is 1.119.

For this data set, assume we have two clustering results generated by different clustering algorithms, as shown in Table 2.2. In the table, we can observe that the first clustering result has five clusters with relatively uniform sizes. This is also indicated by the CV value of 0.421. In contrast, for the second clustering result, the CV value of the cluster sizes is 1.201, which indicates a severe imbalance. According to the entropy measure, clustering result I is better than clustering result II. This is due to the fact that the entropy measure penalizes a large impure cluster more just as the first cluster in clustering I. However, if we look at the five true clusters carefully, we can find that the second clustering result is much closer to the true cluster distribution, and the first clustering result is actually far away from the true cluster distribution. This is also reflected by the CV values; that is, the CV value (1.201) of five cluster sizes in the second clustering result is much closer to the CV value (1.119) of five true cluster sizes.

In summary, this example illustrates that the entropy measure tends to favor K-means which produces clusters in relatively uniform sizes. This effect becomes even more significant in the situation that the data have highly imbalanced true clusters. In other words, if the entropy measure is used for validating K-means clustering, the validation result can be misleading.

Table 2.1 A document data set

1: Sports, Sports	24 objects
2: Entertainment, Entertainment	2 objects
3: Foreign, Foreign, Foreign, Foreign, Foreign	5 objects
4: Metro, Metro, Metro, Metro, Metro, Metro, Metro, Metro, Metro, Metro	10 objects
5: Politics	1 object
$CV = 1.119$	

Table 2.2 Two clustering results

Clustering I	1: Sports Sports Sports Sports Sports Sports Sports Sports	$CV = 0.421$
	2: Sports Sports Sports Sports Sports Sports Sports Sports	$E = 0.247$
	3: Sports Sports Sports Sports Sports Sports Sports Sports	
	4: Metro Metro Metro Metro Metro Metro Metro Metro Metro Metro	
	5: Entertainment Entertainment Foreign Foreign Foreign Foreign Foreign Politics	
Clustering II	1: Sports Foreign	$CV = 1.201$ $E = 0.259$
	2: Entertainment Entertainment	
	3: Foreign Foreign Foreign	
	4: Metro Metro Metro Metro Metro Metro Metro Metro Metro Metro Foreign	
	5: Politics	

2.4 Experimental Results

In this section, we conduct experiments on a number of real-world data sets to show the uniform effect of K-means clustering and the bias effect of the entropy measure.

2.4.1 Experimental Setup

We first introduce the experimental setup, including the clustering tools and the data information.

Clustering Tools. In our experiments, we used the CLUTO implementation of K-means.[1] As the Euclidean notion of proximity is not very effective for K-means

[1] http://glaros.dtc.umn.edu/gkhome/views/cluto

Table 2.3 Some notations used in experiments

CV_0:	The CV value of the true cluster sizes
CV_1:	The CV value of the resulting cluster sizes
DCV:	$CV_0 - CV_1$
$\bar{S}$:	The average cluster sizes
STD_O:	The standard deviation of the true cluster sizes
STD_1:	The standard deviation of the resulting cluster sizes
E:	The entropy measure

Table 2.4 Some characteristics of experimental data sets

Data	Source	#object	#feature	#class	MinClassSize	MaxClassSize	CV_0
			Document data				
fbis	TREC	2463	2000	17	38	506	0.961
hitech	TREC	2301	126373	6	116	603	0.495
sports	TREC	8580	126373	7	122	3412	1.022
tr23	TREC	204	5832	6	6	91	0.935
tr45	TREC	690	8261	10	14	160	0.669
la2	TREC	3075	31472	6	248	905	0.516
ohscal	OHSUMED-233445	11162	11465	10	709	1621	0.266
re0	Reuters-21578	1504	2886	13	11	608	1.502
re1	Reuters-21578	1657	3758	25	10	371	1.385
k1a	WebACE	2340	21839	20	9	494	1.004
k1b	WebACE	2340	21839	6	60	1389	1.316
wap	WebACE	1560	8460	20	5	341	1.040
			Biomedical data				
LungCancer	KRBDSR	203	12600	5	6	139	1.363
Leukemia	KRBDSR	325	12558	7	15	79	0.584
			UCI data				
ecoli	UCI	336	7	8	2	143	1.160
page-blocks	UCI	5473	10	5	28	4913	1.953
pendigits	UCI	10992	16	10	1055	1144	0.042
letter	UCI	20000	16	26	734	813	0.030

clustering on high-dimensional data, the cosine similarity is used in the objective function of K-means. Some notations used in the experiments are shown in Table 2.3.

Experimental Data. We used a number of real-world data sets that were obtained from different application domains. Some characteristics of these data sets are shown in Table 2.4.

Document Data. The `fbis` data set was obtained from the Foreign Broadcast Information Service data of the TREC-5 collection.[2] The `hitech` and `sports` data sets were derived from the San Jose Mercury newspaper articles that were distributed as part of the TREC collection (TIPSTER Vol. 3). The `hitech` data

[2] http://trec.nist.gov

set contains documents about computers, electronics, health, medical, research, and technology, and the `sports` data set contains documents about baseball, basketball, bicycling, boxing, football, golfing, and hockey. Data sets `tr23` and `tr45` were derived from the TREC-5, TREC-6, and TREC-7 collections. The `la2` data set is part of the TREC-5 collection and contains news articles from the Los Angeles Times. The `ohscal` data set was obtained from the OHSUMED collection [12], which contains documents from the antibodies, carcinoma, DNA, in-vitro, molecular sequence data, pregnancy, prognosis, receptors, risk factors, and tomography categories. The data sets `re0` and `re1` were from Reuters-21578 text categorization test collection Distribution 1.0.[3] The data sets `k1a` and `k1b` contain exactly the same set of documents but differ in how the documents were assigned to different classes. In particular, `k1a` contains a finer-grain categorization than that contained by `k1b`. The data set `wap` was obtained from the WebACE project (WAP) [11]; each document corresponds to a web page listed in the subject hierarchy of Yahoo!. For all these data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [20].

Biomedical Data. `LungCancer` [1] and `Leukemia` [26] data sets were obtained from Kent Ridge Biomedical Data Set Repository (KRBDSR), which is an online repository of high-dimensional biomedical data.[4] The `LungCancer` data set consists of samples of lung adenocarcinomas, squamous cell lung carcinomas, pulmonary carcinoid, small-cell lung carcinomas, and normal lung described by 12600 genes. The `Leukemia` data set contains six subtypes of pediatric acute lymphoblastic leukemia samples and one group samples that do not fit in any of the above six subtypes, and each sample is described by 12558 genes.

UCI Data. In addition to the high-dimensional data above, we also used some UCI data sets in lower dimensionality.[5] The `ecoli` data set is about the information of cellular localization sites of proteins. The `page-blocks` data set contains the information of five type blocks of the page layout of a document that is detected by a segmentation process. The `pendigits` and `letter` data sets contain the information of handwritings. The `pendigits` data set includes the number information of 0–9, while the `letter` data set contains the letter information of A–Z.

Note that for each data set in Table 2.4, the experiment was conducted ten times to void the randomness, and the average value is presented.

2.4.2 The Evidence of the Uniform Effect of K-means

Here, we illustrate the uniform effect of K-means clustering. In the experiment, we first used CLUTO with default settings to cluster the input data sets, and then

[3] http://www.research.att.com/~lewis.

[4] http://sdmc.i2r.a-star.edu.sg/rp/

[5] http://www.ics.uci.edu/~mlearn/MLRepository.html

Table 2.5 Experimental results on real-world data sets

Data	$\bar{S}$	STD_0	STD_1	CV_0	CV_1	DCV	E
fbis	145	139	80	0.96	0.55	0.41	0.345
hitech	384	190	140	0.50	0.37	0.13	0.630
k1a	117	117	57	1.00	0.49	0.51	0.342
k1b	390	513	254	1.32	0.65	0.66	0.153
la2	513	264	193	0.52	0.38	0.14	0.401
ohscal	1116	297	489	0.27	0.44	−0.17	0.558
re0	116	174	45	1.50	0.39	1.11	0.374
re1	66	92	22	1.39	0.32	1.06	0.302
sports	1226	1253	516	1.02	0.42	0.60	0.190
tr23	34	32	14	0.93	0.42	0.51	0.418
tr45	69	46	30	0.67	0.44	0.23	0.329
wap	78	81	39	1.04	0.49	0.55	0.313
LungCancer	41	55	26	1.36	0.63	0.73	0.332
Leukemia	46	27	17	0.58	0.37	0.21	0.511
ecoli	42	49	21	1.16	0.50	0.66	0.326
page-blocks	1095	2138	1029	1.95	0.94	1.01	0.146
letter	769	23	440	0.03	0.57	−0.54	0.683
pendigits	1099	46	628	0.04	0.57	−0.53	0.394
Min	34	23	14	0.03	0.33	−0.54	0.146
Max	1226	2138	1029	1.95	0.94	1.11	0.683

computed the CV values of the cluster sizes. The number of clusters K was set to the true cluster number for the purpose of comparison.

Table 2.5 shows the experimental results on real-world data sets. As can be seen, for 15 data sets with relatively large CV_0 values, K-means tends to reduce the variation of the cluster sizes in the clustering results, as indicated by the smaller CV_1 values. This means that the uniform effect of K-means exists for data sets with highly imbalanced true clusters. Indeed, if we look at Eq. (2.13) in Sect. 2.2, this result implies that the factor $\| m_i - m_j \|^2$ is dominated by the factor $n_i n_j$.

For data sets `ohscal`, `letter`, and `pendigits` with very small CV_0 values, however, K-means increases the variation of the cluster sizes slightly, as indicated by the corresponding CV_1 values. This implies that the uniform effect of K-means is not significant for data sets with true clusters in relatively uniform sizes. Indeed, according to Eq. (2.13) in Sect. 2.2, this result indicates that the factor $n_i n_j$ is dominated by the factor $\| m_i - m_j \|^2$.

2.4.3 The Quantitative Analysis of the Uniform Effect

In this experiment, we attempt to get a quantitative understanding about the uniform effect of K-means on the clustering results.

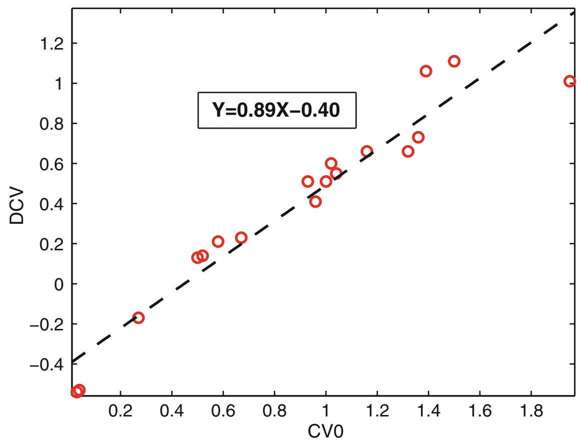


Fig. 2.2 The linear relationship between DCV and CV_0. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

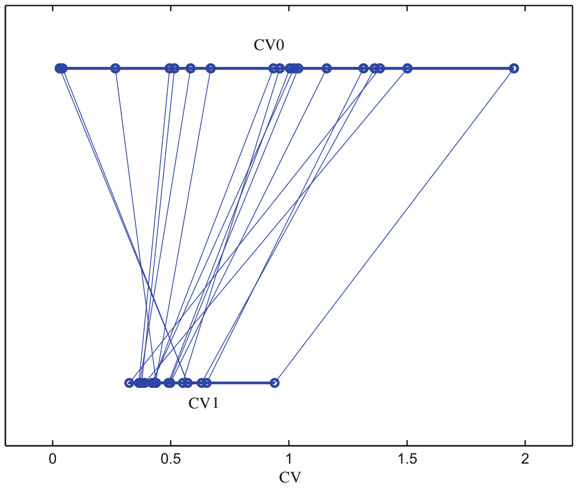


Fig. 2.3 The relationship between CV_1 and CV_0. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

Figure 2.2 shows the relationship between DCV and CV_0, in which all the points (CV_0, DCV) are fitted into a linear line: $y = 0.89x - 0.40$. Apparently, the DCV value increases with the increase of the CV_0 value, and $y = 0$ when $x = 0.45$ in the linear fitting line. This indicates that if $CV_0 > 0.45$, K-means clustering tends to have $CV_1 < CV_0$. Otherwise, $CV_1 > CV_0$. In other words, 0.45 is the empirical threshold to invoke the uniform effect of K-means.

Figure 2.3 shows the relationship between CV_0 and CV_1 for all the experimental data sets listed in Table 2.4. Note that there is a link between CV_0 and CV_1 for every data set. An interesting observation is that, while the range of CV_0 is between 0.03 and 1.95, the range of CV_1 is restricted into a much narrower range from 0.33 to 0.94. So we have the empirical value range of CV_1: [0.3, 1].

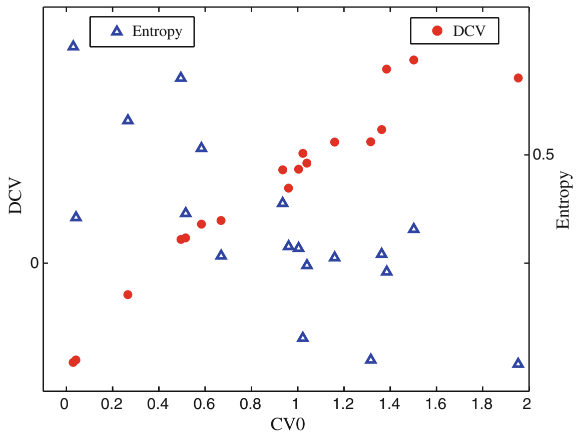


Fig. 2.4 Illustration of the biased effect of the entropy measure. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

2.4.4 The Evidence of the Biased Effect of the Entropy Measure

In this subsection, we present the biased effect of the entropy measure on the clustering results of K-means. Figure 2.4 shows the entropy values of the clustering results of all 18 data sets. A general trend is that while the difference of the cluster size distributions before and after clustering increases with the increase of CV_0, the entropy value tends to decrease. In other words, there is a disagreement between DCV and the entropy measure on evaluating the clustering quality. Entropy indicates a higher quality as CV_0 increases, but DCV denies this by showing that the distribution of the clustering result is getting farther away from the true distribution. This observation well agrees with our analysis in Sect. 2.3 that entropy has a biased effect on K-means.

To further illustrate the biased effect of entropy, we also generated two groups of synthetic data sets. These data sets have wide ranges of distributions of true cluster sizes. The first group of synthetic data sets was derived from the `pendigits` data set. We applied the following sampling strategy: (1) The original data set was first sampled to get a sample of 10 classes, each of which contains 1000, 100, 100, ..., 100 objects, respectively; (2) To get data sets with decreasing CV_0 values, the size of the largest class was gradually reduced from 1000 to 100; (3) To get data sets with increasing CV_0 values, the sizes of the remaining nine classes were gradually reduced from 100 to 30. A similar sampling strategy was also applied to the `letter` data set for generating the second group of synthetic data sets. Note that we repeated sampling a data set ten times, and output the average evaluation of the clustering results.

Figures 2.5 and 2.6 show the clustering results evaluated by the entropy measure and DCV, respectively. A similar trend can be observed; that is, the entropy value decreases as the CV_0 value increases. This further justifies the existence of the biased effect of the entropy measure on the clustering result of K-means.

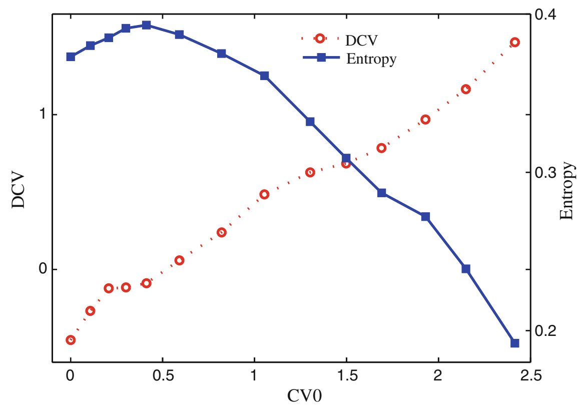


Fig. 2.5 The biased effect of entropy: synthetic data from `pendigits`. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

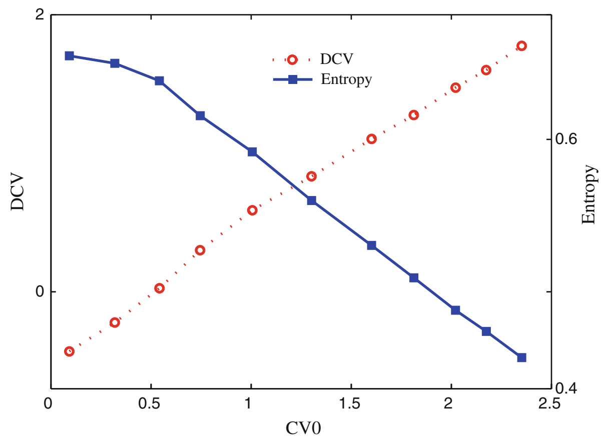


Fig. 2.6 The biased effect of entropy: synthetic data from `letter`. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

2.4.5 The Hazard of the Biased Effect

Having the biased effect, it is very dangerous to use the entropy measure for the validation of K-means. To illustrate this, we selected five data sets with high CV_0 values, i.e. `re0`, `re1`, `wap`, `ecoli`, and `k1a`, for experiments. We did K-means clustering on these data sets, and labeled each cluster by the label of the members in majority. We found that many true clusters were disappeared in the clustering results. Figure 2.7 shows the percentage of the disappeared true clusters in the clustering results. As can be seen, every data set has a significant number of true clusters disappeared. For the `re0` data set ($CV_0 = 1.502$), even more than 60 % true clusters

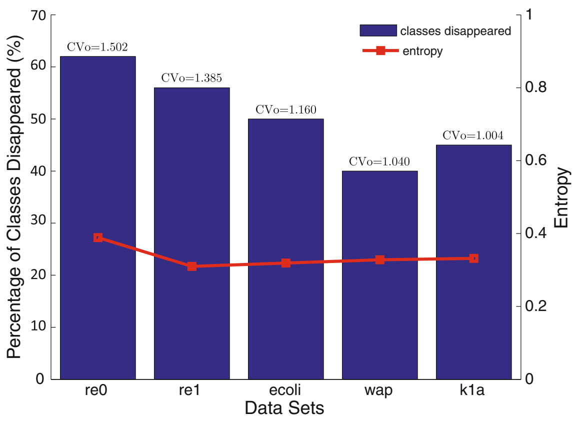


Fig. 2.7 The percentage of the disappeared true clusters in highly imbalanced data. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

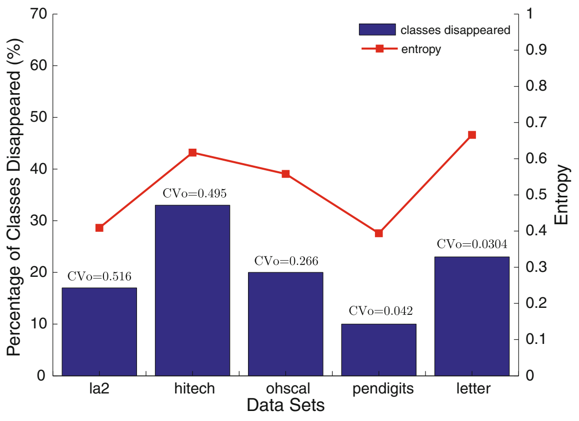


Fig. 2.8 The percentage of the disappeared true clusters in relatively balanced data. © 2009 IEEE. Reprinted, with permission, from Ref. [25]

disappear after K-means clustering! In sharp contrast, as shown in Fig. 2.7, very low entropy values were achieved for these five data sets, which imply that the performance of K-means clustering is "excellent". This experiment clearly illustrates the hazard of the biased effect of entropy.

For the purpose of comparison, we also conducted a similar experiment on five data sets with low CV_0 values. Figure 2.8 shows the percentage of the disappeared true clusters. An interesting observation is that, compared to the results of data sets with high CV_0 values, the percentages of the disappeared true clusters become much smaller, and the entropy values increase. In other words, the entropy measure is more reliable for data sets with relatively uniform true cluster sizes.

2.5 Related Work

People have investigated K-means clustering from various perspectives. Many data factors, which may strongly affect the performance of K-means clustering, have been identified and addressed. In the following, we highlight some research results which are most related to the main theme of this chapter.

First, people have studied the impact of high dimensionality on the performance of K-means clustering, and found that the traditional Euclidean notion of proximity is not very effective for K-means clustering on real-world high-dimensional data, such as gene expression data and document data. To meet this challenge, one research direction is to make use of dimension reduction techniques, such as Multidimensional Scaling (MDS) [2], Principal Components Analysis (PCA) [15], and Singular Value Decomposition (SVD) [6]. Also, several feature transformation techniques have been proposed for high-dimensional document data, such as Latent Semantic Indexing (LSI), Random Projection (RP), and Independent Component Analysis (ICA). In addition, feature selection techniques have been widely used, and a detailed discussion and comparison of these techniques have been provided by Tang et al. [24]. Another direction for this problem is to redefine the notions of proximity, e.g. by the Shared Nearest Neighbors (SNN) similarity introduced by Jarvis and Patrick [14]. Finally, some other similarity measures, e.g. the cosine measure, have also shown appealing effects on clustering document data [29].

Second, it has been recognized that K-means has difficulty in detecting the "natural" clusters with non-globular shapes [13, 23]. To address this, one research direction is to modify the K-means clustering algorithm. For instance, Guha et al. [9] proposed the CURE method which makes use of multiple representative points to get the shape information of the "natural" clusters. Another research direction is to use some non-prototype-based clustering methods which usually perform better than K-means on data in non-globular or irregular shapes [23].

Third, outliers and noise in the data can also degrade the performance of clustering algorithms [16, 27, 30], especially for prototype-based algorithms such as K-means. To deal with this, one research direction is to incorporate some outlier removal techniques before conducting K-means clustering. For instance, a simple method of detecting outliers is based on the distance measure [16]. Breunig et al. [4] proposed a density based method using the Local Outlier Factor (LOF) for the purpose of identifying outliers in data with varying densities. There are also some other clustering based methods to detect outliers as small and remote clusters [21], or objects that are farthest from their corresponding cluster centroids [18]. Another research direction is to handle outliers during the clustering process. There have been serval techniques designed for such purpose. For example, DBSCAN automatically classifies low-density points as noise points and removes them from the clustering process [8]. Also, SNN density-based clustering [7] and CURE [9] explicitly deal with noise and outliers during the clustering process.

Fourth, many clustering algorithms that work well for small or medium-size data are unable to handle large-scale data. Along this line, a discussion of scaling

K-means clustering to large-scale data was provided by Bradley et al. [3]. A broader discussion of specific clustering techniques can be found in [19]. Some representative techniques include CURE [9], BIRCH [28], and so on.

Finally, some researchers have identified some other factors, such as the types of attributes and data sets, that may impact the performance of K-means clustering. However, in this chapter, we focused on understanding the uniform effect of K-means and the biased effect of the entropy measure, which have not been systematically studied in the literature.

2.6 Concluding Remarks

In this chapter, we present an organized study on K-means clustering and cluster validation measures from a data distribution perspective. We first theoretically illustrate that K-means clustering tends to produce clusters with uniform sizes. We then point out that the widely adopted validation measure entropy has a biased effect and therefore cannot detect the uniform effect of K-means. Extensive experiments on a number of real-world data sets clearly illustrate the uniform effect of K-means and the biased effect of the entropy measure, via the help of the Coefficient of Variation statistic. Most importantly, we unveil the danger induced by the combined use of K-means and the entropy measure. That is, many true clusters will become unidentifiable when applying K-means for highly imbalanced data, but this situation is often disguised by the low values of the entropy measure.

References

1. Bhattacharjee, A., Richards, W., Staunton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M., Loda, M., Weber, G., Mark, E., Lander, E., Wong, W., Johnson, B., Golub, T., Sugarbaker, D., Meyerson, M.: Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. In. Proceedings of the National Academy of Sciences of the United States of America, **98**, 13790–13795 (2001)
2. Borg, I., Groenen, P.: Modern Multidimensional Scaling–Theory and Applications. Springer Verlag, New York (1997)
3. Bradley, P., Fayyad, U., Reina, C.: Scaling clustering algorithms to large databases. In: Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 9–15 (1998)
4. Breunig, M., Kriegel, H.P., Ng, R., Sander, J.: Lof: identifing density based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 427–438 (2000)
5. DeGroot, M., Schervish, M.: Probability and Statistics, 3rd edn. Addison Wesley, Upper Saddle River (2001)
6. Demmel, J.: Applied numerical linear algebra. Soc. Ind. App. Math. **32**, 206–216 (1997)
7. Ertoz, L., Steinbach, M., Kumar, V.: A new shared nearest neighbor clustering algorithm and its applications. In: Proceedings of the Workshop on Clustering High Dimensional Data and its Applications at the 2nd SIAM International Conference on Data Mining (2002)

8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
9. Guha, S., Rastogi, R., Shim, K.: Cure: an efficient clustering algorithm for large databases. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 73–84 (1998)
10. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Cluster validity methods: Part i. SIGMOD Record **31**(2), 40–45 (2002)
11. Han, E.H., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Webace: a web agent for document categorization and exploration. In: Proceedings of the 2nd International Conference on Autonomous Agents (1998)
12. Hersh, W., Buckley, C., Leone, T.J., Hickam, D.: Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in, Information Retrieval, pp. 192–201 (1994)
13. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice Hall, Englewood cliff (1998)
14. Jarvis, R., Patrick, E.: Clusering using a similarity measure based on shared nearest neighbors. IEEE Trans. Comput. **C-22**(11), 1025–1034 (1973)
15. Jolliffe, I.: Principal Component Analysis, 2nd edn. Springer Verlag, New York (2002)
16. Knorr, E., Ng, R., Tucakov, V.: Distance-based outliers: algorithms and applications. VLDB J. **8**, 237–253 (2000)
17. Krishna, K., Narasimha Murty, M.: Genetic k-means algorithm. IEEE Trans. Syst. Man Cybern. Part B **29**(3), 433–439 (1999)
18. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 16–22 (1999)
19. Murtagh, F.: Clustering massive data sets. In: Abello, J., Pardalos, P.M., Resende, M.G. (eds.) Handbook of Massive Data Sets, pp. 501–543. Kluwer Academic Publishers, Norwell (2002)
20. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
21. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proceedings of the 2001 ACM CSS Workshop on Data Mining Applied to, Security (DMSA-2001) (2001)
22. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of the KDD Workshop on Text Mining (2000)
23. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Upper Saddle River (2005)
24. Tang, B., Shepherd, M., Heywood, M., Luo, X.: Comparing dimension reduction techniques for document clustering. In: Proceedings of the Canadian Conference on, Artificial Intelligence, pp. 292–296 (2005)
25. Xiong, H., Wu, J., Chen, J.: K-means clustering versus validation measures: a data-distribution perspective. IEEE Trans. Syst. Man Cybern. Part B Cybern. **39**(2), 318–331 (2009)
26. Yeoh, E.J.: Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell **1**, 133–143 (2002)
27. Zhang, J.S., Leung, Y.W.: Robust clustering by pruning outliers. IEEE Trans. Syst. Man Cybern. Part B **33**(6), 983–998 (2003)
28. Zhang, T., Ramakrishnan, R., M.Livny: Birch: an efficient data clustering mehtod for very large databases. In: Proceedings of 1996 ACM SIGMOD International Conference on Management of Data, pp. 103–114 (1996)
29. Zhao, Y., Karypis, G.: Criterion functions for document clustering: experiments and analysis. Mach. Learn. **55**(3), 311–331 (2004)
30. Zhou, A., Cao, F., Yan, Y., Sha, C., He, X.: Distributed data stream clustering: a fast em-based approach. In: Proceedings of the 23rd International Conference on Data, Engineering, pp. 736–745 (2007)

Chapter 3
Generalizing Distance Functions for Fuzzy c-Means Clustering

3.1 Introduction

Fuzzy c-means (FCM) is a well-known partitional clustering method, which allows an object to belong to two or more clusters with a membership grade between zero and one [3, 10, 18, 38]. Recently, due to the rich information conveyed by the membership grade matrix, FCM has been widely used in many real-world application domains where well-separated clusters are typically not available. In addition, people also recognize that the simple centroid-based iterative procedure of FCM is very appealing when dealing with large volumes of data.

In the literature, considerable research efforts have been dedicated to fuzzy c-means clustering, e.g., adapting FCM to specific domains by modifying the objective function or the constraints, exploiting new distance functions other than the squared Euclidean distance for FCM, proving the convergence of the centroid-based alternating optimization (AO) method of FCM, and improving the computational efficiency of FCM. Nonetheless, the common characteristics of distance functions suitable for FCM remain unclear. Further study is still needed to establish a general understanding that can characterize the interrelationships between the distance functions and the effectiveness of the AO method for FCM. Specifically, in this chapter, we aim to answer the following two questions:

- Are there more distance functions that can be used for FCM while preserving the applicability of the centroid-based AO method for FCM?
- What are the necessary and sufficient conditions for such distance functions?

The answers to the above questions are valuable since (1) diversified distance functions can provide extra flexibility when clustering data with substantially distinct features, and (2) the centroid-based AO method is very simple and often leads to a quick convergence. Along this line, in this chapter, we provide an organized study of the generalization issues of distance functions for FCM. The major contributions of this research are summarized as follows:

First, we propose a strict definition for distance functions that *fit* fuzzy c-means *directly*. That is, a distance function fits FCM directly, if and only if the iteration sequence generated by the *centroid-based* AO method can lead to the continuous decrease of the objective function value, and result in a global convergence. This definition distinguishes this study from the existing ones, which introduce distance functions to FCM by using computationally more expensive optimization methods. Finally, by definition, the directly-fit distance functions can preserve the simplicity and high efficiency of FCM via using centroids of arithmetic means.

Second, we show that any distance function that fits FCM directly can be derived from a continuously differentiable convex function. As they share a same mathematical expression, we call them the *point-to-centroid distance* (P2C-D). In other words, a distance function fits FCM directly only if it is a P2C-D (i.e., an instance of P2C-D). We further divide P2C-D into two categories according to the convexity of the derivation function. We show that Type-I P2C-D is the well-known Bregman divergence derived by strictly convex functions. However, the widely used cosine similarity belongs to Type-II P2C-D, which is derived by convex-but-not-strictly-convex functions. To the best of our knowledge, this study is the first one to explore how to use Type-II P2C-D for FCM.

Third, we strictly prove that if the membership grade matrix is nondegenerate, the point-to-centroid distance indeed fits FCM directly. This is done by proving the global convergence of the iteration sequence generated by the centroid-based AO method. When there is degeneration for Type-II P2C-D, we also provide a feasible solution which still guarantees the convergence of the FCM iterations. As a result, we eventually establish the general theory that a distance function fits FCM directly if and only if it is a point-to-centroid distance. As a "side product", we also point out that the point-to-centroid distance is also the generalized distance function of K-means clustering, which is actually a special case of FCM when the fuzzy factor $m \to 1$.

Finally, we conduct experiments on both synthetic and real-world data sets. The results demonstrate the global convergence of FCM using various point-to-centroid distances. Also, there are strong evidences that adapting the right distance functions for data sets in different application domains can improve the clustering quality substantially.

The remainder of this chapter is organized as follows. Section 3.2 describes the FCM method and defines the problem. In Sect. 3.3, we introduce the point-to-centroid distance. Section 3.4 shows the global convergence of FCM using the point-to-centroid distance. In Sect. 3.5, we give some examples of the point-to-centroid distance. Section 3.6 presents experimental results. Finally, we describe the related work in Sect. 3.7 and draw conclusions in Sect. 3.8.

3.2 Preliminaries and Problem Definition

In this section, we first introduce the Zangwill's global convergence theorem and the fuzzy c-means method. Then, we define the problem to be studied in this chapter.

3.2.1 Math Notations

Vectors are presented by bold-faced and lower-case alphabets, e.g., $\boldsymbol{x}$ and $\boldsymbol{v}$. Matrices are represented by bold-faced and upper-case alphabets, e.g., $\boldsymbol{U}$ and $\boldsymbol{V}$. Sets are represented by calligraphic upper-case alphabets, e.g., $\mathscr{S}$ and $\mathscr{I}$. Let $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{R}_{++}$, $\mathbb{R}^d$, and $\mathbb{R}^{cd}$ denote the sets of reals, non-negative reals, positive reals, d-dimensional real vectors, and $c \times d$ real matrix, respectively. Let $\mathbb{Z}$ denote the sets of integers, and $\mathbb{Z}_+$, $\mathbb{Z}_{++}$, $\mathbb{Z}^d$, and $\mathbb{Z}^{cd}$ are defined analogously. For real vector $\boldsymbol{x}$, $\|\boldsymbol{x}\|$ denotes the l_2 norm, and $\boldsymbol{x}^T$ denotes the transposition. For a multivariate function f, $\mathrm{dom}_i(f)$ denotes the domain of the ith variable. We use f_x to denote the first-order partial derivative of f w.r.t. variable x, and f_{xy} to denote the second-order partial derivative accordingly. The gradient of f w.r.t. $\boldsymbol{x}$ is represented by $\nabla_{\boldsymbol{x}} f$. ln and log are used to represent the natural logarithm and the logarithm of base 2, respectively.

3.2.2 Zangwill's Global Convergence Theorem

Many clustering algorithms can be formulated as the optimization of some objective functions, e.g., the fuzzy c-means algorithm, the mixture model, and the K-means algorithm. To solve the optimization problems, an Alternating Optimization (AO) method is often employed, which iteratively produces a sequence of solutions with the hope of approaching the optimal solution(s). However, not all the sequences generated by AO can converge to the solution set. To clarify this, the global convergence theorem proposed by Zangwill [29, 52] is often used, which establishes some technical conditions for which convergence is guaranteed. In what follows, we state without proof the specific results of the global convergence theorem.

Theorem 3.1 ([52]) *Let A be an algorithm on $\mathscr{X}$, and suppose that, given $\boldsymbol{x}^{(0)}$ the sequence $(\boldsymbol{x}^{(l)})_{l=0}^{\infty}$ is generated satisfying $\boldsymbol{x}^{(l+1)} \in A(\boldsymbol{x}^{(l)})$. Let a solution set $\Omega \subset \mathscr{X}$ be given, and suppose*

1. *all points $\boldsymbol{x}^{(l)}$, $l = 0, 1, \ldots$ are contained in a compact set $\mathscr{S} \subset \mathscr{X}$;*
2. *there is a continuous function f on $\mathscr{X}$ such that*
   - *if $\boldsymbol{x} \notin \Omega$, then $f(\boldsymbol{y}) < f(\boldsymbol{x})$ for any $\boldsymbol{y} \in A(\boldsymbol{x})$,*
   - *if $\boldsymbol{x} \in \Omega$, then $f(\boldsymbol{y}) \le f(\boldsymbol{x})$ for any $\boldsymbol{y} \in A(\boldsymbol{x})$;*
3. *the mapping A is closed at points outside Ω.*

Then the limit of any convergent subsequence of $(\boldsymbol{x}^{(l)})_{l=0}^{\infty}$ is a solution.

Remark The construction of a suitable solution set is often the key point when using this theorem. Also note that "global" here means the starting point of the sequence can be arbitrary, but is not the guarantee that the algorithm converges to the global optimum.

To establish the closeness of a point-to-set mapping A for Theorem 3.1, we also need the following corollary in [52]:

Corollary 3.1 ([52]) *Let $C : \mathscr{M} \mapsto \mathscr{S}$ be a function and $B : \mathscr{S} \mapsto P(\mathscr{S})$ is a point-to-set mapping. Assume that C is continuous at $\boldsymbol{x}$ and B is closed at $C(\boldsymbol{x})$. Then $A = B \circ C$ is closed at $\boldsymbol{x}$.*

3.2.3 Fuzzy c-Means

Fuzzy c-Means (FCM) is a method of clustering which allows one data point to belong to two or more clusters. Given a data set $\mathscr{X} = \{\boldsymbol{x}_k\}_{k=1}^n$ containing $n > c$ distinct points in d dimensions, FCM aims to minimize the following objective function:

$$\min J_m(\boldsymbol{U}, \boldsymbol{V}) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2, \tag{3.1}$$

$$\text{s.t.} \sum_{i=1}^{c} u_{ik} = 1,\ 1 \le k \le n, \tag{3.2a}$$

$$\sum_{k=1}^{n} u_{ik} > 0,\ 1 \le i \le c, \tag{3.2b}$$

$$0 \le u_{ik} \le 1,\ 1 \le i \le c,\ 1 \le k \le n, \tag{3.2c}$$

where $\boldsymbol{v}_i \in \mathbb{R}^d$ is the centroid of cluster i, $1 \le i \le c$, u_{ik} is the membership grade of $\boldsymbol{x}_k$ in cluster i, and $m \in (1, +\infty)$ is the fuzzy factor. The set of all matrices in $\mathbb{R}^{cn}$ satisfying Eq. (3.2) is denoted as M_{fc}. Therefore, from an optimization perspective, the fuzzy c-means problem is to find good partitions $\boldsymbol{U} \in M_{fc}$ and centroids $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c)^T \in \mathbb{R}^{cd}$ such that J_m can be minimized. In [3], the author gave the necessary conditions for a minimizer $(\boldsymbol{U}^*, \boldsymbol{V}^*)$ as follows.

Theorem 3.2 ([3]) *Let $\mathscr{X} = \{\boldsymbol{x}_k\}_{k=1}^n$ contain $n > c$ distinct points and $m \in (1, +\infty)$. Let $d_{ik} = \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2$, $1 \le k \le n$, $1 \le i \le c$. $\forall$ k, define the sets $I_k = \{i | d_{ik} = 0,\ 1 \le i \le c\}$ and $\tilde{I}_k = \{1, \ldots, c\} \setminus I_k$. Then J_m may be globally minimized only if*

Fig. 3.1 The basic fuzzy c-means algorithm

Algorithm 1: **Fuzzy c-Means (FCM)**

```
1.  $l \leftarrow 0$;
2.  Initialize $\boldsymbol{U}^{(l)}$ and $\boldsymbol{V}^{(l)}$;
3.  repeat
4.    $l \leftarrow l+1$;
5.    Calculate $\boldsymbol{V}^{(l)}$ using Eq. (3.3) and $\boldsymbol{U}^{(l-1)}$;
6.    Calculate $\boldsymbol{U}^{(l)}$ using Eq. (3.4) and $\boldsymbol{V}^{(l)}$;
7.  until Any stopping criterion is met;
```

$$\boldsymbol{v}_i^* = \sum_{k=1}^{n} (u_{ik}^*)^m \boldsymbol{x}_k \Big/ \sum_{k=1}^{n} (u_{ik}^*)^m, \quad 1 \le i \le c, \tag{3.3}$$

and

$$u_{ik}^* = \frac{(d_{ik}^*)^{-1/(m-1)}}{\sum_{l=1}^{c} (d_{lk}^*)^{-1/(m-1)}}, \quad \textit{if } I_k = \emptyset, \textit{ and} \tag{3.4a}$$

$$u_{ik}^* = 0 \;\forall\, i \in \tilde{I}_k \textit{ and } \sum_{i \in I_k} u_{ik}^* = 1, \quad \textit{if } I_k \neq \emptyset, \tag{3.4b}$$

where $d_{ik}^* = \|\boldsymbol{x}_k - \boldsymbol{v}_i^*\|^2$, $1 \le k \le n$, $1 \le i \le c$.

The proof of Theorem 3.2 can be found in pages 67–69 of [3]. Note that the computations of $\boldsymbol{U}^*$ and $\boldsymbol{V}^*$ in Eqs. (3.3) and (3.4) are interdependent. Therefore, a natural way to find the solution(s) is to employ an AO method, as shown in Fig. 3.1.

The FCM algorithm in Fig. 3.1 iterates through Steps 5 and 6. Note that in Step 6, Eq. (3.4b) may not uniquely specify $\boldsymbol{U}$ in case that multiple singularities occur, i.e., $\exists\, i \neq j,\ d_{ik}^* = d_{jk}^* = 0$ for some $\boldsymbol{x}_k$. However, a particular choice for $\boldsymbol{U}$ still must be made when implementing FCM. We detail this in the experimental section below. To further describe the iteration, some notation is given as follows. Let $G : M_{fc} \mapsto \mathbb{R}^{cd}$ be the function defined by $G(\boldsymbol{U}) = \boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c)^T$, where $\boldsymbol{v}_i$ is calculated by Eq. (3.3), $\forall\, 1 \le i \le c$. Let $F : \mathbb{R}^{cd} \mapsto P(M_{fc})$ denote the point-to-set mapping defined by $F(\boldsymbol{V}) = \{\boldsymbol{U} \in M_{fc} | \boldsymbol{U}$ satisfies Eq. (3.4)$\}$. Then the FCM iteration can be described by the point-to-set mapping $T_m : M_{fc} \times \mathbb{R}^{cd} \mapsto P(M_{fc} \times \mathbb{R}^{cd})$ as follows:

$$T_m(\boldsymbol{U}, \boldsymbol{V}) = \{(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) | \hat{\boldsymbol{V}} = G(\boldsymbol{U}),\ \hat{\boldsymbol{U}} \in F(\hat{\boldsymbol{V}})\}. \tag{3.5}$$

We say that $((\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}))_{l=0}^{\infty}$ is an FCM iteration sequence if $\boldsymbol{U}^{(0)} \in M_{fc}$, $\boldsymbol{V}^{(0)} = G(\boldsymbol{U}^{(0)})$, and $(\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}) \in T_m(\boldsymbol{U}^{(l-1)}, \boldsymbol{V}^{(l-1)})$ for $l = 1, 2, \ldots$. Based on the above notation, [16] gave an improved proof for the global convergence of the FCM algorithm, which was originally established in [2].

Let the solution set $\Omega = \{(\boldsymbol{U}^*, \boldsymbol{V}^*) \in M_{fc} \times \mathbb{R}^{cd} | J_m(\boldsymbol{U}^*, \boldsymbol{V}^*) \leq J_m(\boldsymbol{U}, \boldsymbol{V}^*) \, \forall \boldsymbol{U} \in M_{fc}$, and $J_m(\boldsymbol{U}^*, \boldsymbol{V}^*) < J_m(\boldsymbol{U}^*, \boldsymbol{V}) \, \forall \, \boldsymbol{V} \neq \boldsymbol{V}^*\}$. Then we have the global convergence theorem as follows:

Theorem 3.3 ([16]) *Given the fuzzy c-means problem defined by Eqs. (3.1)–(3.2), let* $(\boldsymbol{U}^{(0)}, G(\boldsymbol{U}^{(0)}))$ *be the starting point of iteration with* T_m*, where* $\boldsymbol{U}^{(0)} \in M_{fc}$*. Then the iteration sequence* $((\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}))_{l=0}^{\infty}$ *either terminates at a point in* Ω*, or there is a subsequence converging to a point in* Ω*.*

The proof can be found in [16], which is based heavily on the Zangwill's convergence theorem [52] and the original proof of Bezdek [2]. Theorem 3.3 guarantees that arbitrary iteration sequences generated by FCM converge, at least along a subsequence, to either a local minimum or a saddle point of J_m.

3.2.4 Problem Definition

Theorems 3.2 and 3.3 show that the AO method represented by T_m can be used to find a solution for min J_m. This is considered valuable, since (1) computing $\boldsymbol{V}$ and $\boldsymbol{U}$ through Eqs. (3.3)–(3.4) are very simple, and (2) the global convergence of the generated sequence is guaranteed. However, this result is based on an important premise—the distance function used for FCM is the squared Euclidean distance (i.e., the l_2 norm) or its immediate extension to inner product (e.g., the Mahalanobis distance) [3]. As a natural extension, one may raise questions as follows:

- Are there any more distance functions that can be used for FCM while keeping the effectiveness of T_m in finding a solution?
- What are the necessary and sufficient conditions for such distance functions?

To answer these questions, we should first adapt the FCM problem to a more general circumstance. We have the objective function of the Generalized-Distance-Based Fuzzy *c*-Means (GD-FCM) as follows:

$$\begin{aligned} \min J_{mf}(\boldsymbol{U}, \boldsymbol{V}) &= \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m f(\boldsymbol{x}_k, \boldsymbol{v}_i), \qquad (3.6) \\ \text{s.t. } \boldsymbol{U} &\in M_{fc}, \ \boldsymbol{v}_i \in \text{dom}_2(f), \ \forall 1 \leq i \leq c, \end{aligned}$$

where f is some distance function, $m \in (1, +\infty)$, $1 < c < n$. Then we give the definition of the *direct fitness* of f as follows.

Definition 3.1 (*f's Direct Fitness*) A distance function f is said to fit GD-FCM directly, if and only if the iteration sequence generated by T_m in Eq. (3.5) decreases J_{mf} monotonically and has a global convergence.

Note that for T_m in Definition 3.1, d_{ik}^* has a more general concept in Eq. (3.4), which indeed represents $f(\boldsymbol{x}_k, \boldsymbol{v}_i^*)$, $1 \leq k \leq n$, $1 \leq i \leq c$. Now, we formulate the problem to be studied as follows:

Find all the distance functions that fit GD-FCM directly.

Remark Note that, in this study, the introduction of the distance functions that fit GD-FCM directly does not require to modify the classic fuzzy c-means algorithm as shown in Fig. 3.1. In other words, Algorithm 1 is also the GD-FCM clustering algorithm if the directly-fit distances are used. In the literature, some researchers have used some distance functions other than the squared Euclidean distance for the fuzzy c-means clustering. However, these distance functions, such as l_1 and l_∞ norms [5], l_p norm [17], and the modified distances [26], require to modify the optimization scheme of the classic fuzzy c-means algorithm, and thus are beyond the scope of our study.

3.3 The Point-to-Centroid Distance

Here, we first derive the general point-to-centroid distance for GD-FCM. Then, we show the categories of the point-to-centroid distance and some of its mathematical properties.

3.3.1 Deriving the Point-to-Centroid Distance

Lemma 3.1 *Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set, and let $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ be a differentiable function. If f fits GD-FCM directly, then for any subset $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathscr{S}$, there exist $\lambda_1, \ldots, \lambda_n \in [0, 1]$ with $\sum_{k=1}^n \lambda_k > 0$ satisfying*

$$\sum_{k=1}^{n} \lambda_k \nabla_{\boldsymbol{y}} f(\boldsymbol{x}_k, \boldsymbol{y}^*) = 0, \tag{3.7}$$

where

$$\boldsymbol{y}^* = \sum_{k=1}^{n} \lambda_k \boldsymbol{x}_k / \sum_{k=1}^{n} \lambda_k. \tag{3.8}$$

Proof As f fits GD-FCM directly, according to Definition 3.1, $\boldsymbol{v}_i^* = \frac{\sum_{k=1}^n (u_{ik})^m \boldsymbol{x}_k}{\sum_{k=1}^n (u_{ik})^m}$ is the minimizer of $\sum_{k=1}^n (u_{ik})^m f(\boldsymbol{x}_k, \boldsymbol{v}_i)$, given fixed u_{ik}, $1 \le k \le n$, $1 \le i \le c$. This indicates that $\sum_{k=1}^n (u_{ik})^m \nabla_{\boldsymbol{v}_i} f(\boldsymbol{x}_k, \boldsymbol{v}_i^*) = 0$. Let $\lambda_k \equiv (u_{ik})^m$, $\boldsymbol{y} \equiv \boldsymbol{v}_i$, and the lemma thus follows. □

Lemma 3.2 *Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set, and let $\phi : \mathscr{S} \mapsto \mathbb{R}$ be a continuously differentiable function. Then ϕ is convex if and only if $\forall\, \boldsymbol{x},\ \boldsymbol{y} \in \mathscr{S}$, $\phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla\phi(\boldsymbol{y}) \geq 0$. Further, ϕ is strictly convex if and only if $\forall\, \boldsymbol{x} \neq \boldsymbol{y}$, $\phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla\phi(\boldsymbol{y}) > 0$.*

Lemma 3.2 is well-known as the first-order convexity condition. The proof can be found in pages 69–70 in [6], which we omit here. Now, based on the above two lemmas, we can derive a necessary condition for f being a distance function that fits GD-FCM directly. We have the following theorem.

Theorem 3.4 *Let $\mathscr{S} \subseteq \mathbb{R}$ be a nonempty open convex set. Assume $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ is a continuously differentiable function satisfying: (1) $f(x, x) = 0$, $\forall\, x \in \mathscr{S}$; (2) $f_y(x, y)$ is continuously differentiable on x. If f fits GD-FCM directly, there exists some continuously differentiable convex function $\phi : \mathscr{S} \mapsto \mathbb{R}$ such that*

$$f(x, y) = \phi(x) - \phi(y) - (x - y)\phi'(y). \tag{3.9}$$

Proof According to Lemma 3.1, since f fits GD-FCM directly, $\exists\, \lambda_k \in [0, 1]$, $\sum_{k=1}^n \lambda_k > 0$, we have

$$\sum_{k=1}^{n} \lambda_k f_y(x_k, y^*) = 0, \tag{3.10}$$

where $y^* = \sum_{k=1}^n \lambda_k x_k / \sum_{k=1}^n \lambda_k$. Note that it is trivial to have only one $\lambda_k > 0$ for Eq. (3.10), which will lead to $\nabla_{\boldsymbol{y}} f(\boldsymbol{x}_k, \boldsymbol{y}^*) = 0$, where $\boldsymbol{y}^* = \boldsymbol{x}_k$. Therefore, without loss of generality, we assume $\lambda_1, \lambda_2 > 0$. Let $x_1' = x_1 + \frac{\delta}{\lambda_1}$, $x_2' = x_2 - \frac{\delta}{\lambda_2}$, $\delta > 0$. Then we have

$$y^* = \sum_{k=1}^{n} \lambda_k x_k / \sum_{k=1}^{n} \lambda_k = (\lambda_1 x_1' + \lambda_2 x_2' + \sum_{k=3}^{n} \lambda_k x_k) / \sum_{k=1}^{n} \lambda_k .$$

According to Eq. (3.10), we have

$$\lambda_1 f_y(x_1', y^*) + \lambda_2 f_y(x_2', y^*) + \sum_{k=3}^{n} \lambda_k f_y(x_k, y^*) = 0. \tag{3.11}$$

Subtracting Eq. (3.11) by (3.10), and dividing both sides by δ, we have

$$\begin{aligned} \Delta^+ &\doteq \frac{f_y(x_1 + \frac{\delta}{\lambda_1}, y^*) - f_y(x_1, y^*)}{\frac{\delta}{\lambda_1}} \\ &= \frac{f_y(x_2 - \frac{\delta}{\lambda_2}, y^*) - f_y(x_2, y^*)}{-\frac{\delta}{\lambda_2}} \doteq \Delta^- . \end{aligned} \tag{3.12}$$

Let $\delta \to 0$, we have $f_{yx}^{+}(x_1, y^*) = \lim_{\delta \to 0} \Delta^{+} = \lim_{\delta \to 0} \Delta^{-} = f_{yx}^{-}(x_2, y^*)$. Since f_y is continuously differentiable on x, we have $f_{yx}(x_1, y^*) = f_{yx}(x_2, y^*)$. Similarly, we have

$$f_{yx}(x_k, y^*) = f_{yx}(x_{k'}, y^*),\ \forall\, k \neq k'. \tag{3.13}$$

Note that Eq. (3.13) holds for any subset $\{x_1, \ldots, x_n\} \subset \mathscr{S}$ with $y^* = \sum_{k=1}^{n} \lambda_k x_k / \sum_{k=1}^{n} \lambda_k$, which implies that x is not the variable of f_{yx}. Let $f_{yx}(x, y) = -h(y)$. Integrating f_{yx} w.r.t. x, we have

$$f_y(x, y) = -h(y)x + I(y). \tag{3.14}$$

Recall Eq. (3.10). If we substitute "$f_y(x_k, y^*)$" in Eq. (3.10) by "$-h(y^*)x_k + I(y^*)$" according to Eq. (3.14), we have

$$-h(y^*) \sum_{k=1}^{n} \lambda_k x_k + I(y^*) \sum_{k=1}^{n} \lambda_k = 0.$$

Since y^* is the λ-weighted arithmetic mean of x_k, $1 \le k \le n$, i.e., $\sum_{k=1}^{n} \lambda_k x_k = y^* \sum_{k=1}^{n} \lambda_k$, we have

$$(-h(y^*)y^* + I(y^*)) \sum_{k=1}^{n} \lambda_k = 0.$$

Since $\sum_{k=1}^{n} \lambda_k > 0$, we have $I(y^*) = h(y^*)y^*$. Due to the arbitrariness of y^*, we finally have $I(y) = h(y)y$. Accordingly, we have

$$f_y(x, y) = (y - x)h(y). \tag{3.15}$$

Let ϕ be a continuously differentiable function such that $\phi'(y) = \int h(y)dy$. Since $f(x, x) = 0$, we have

$$\begin{aligned} f(x, y) &= \int_x^y (y - x)h(y)dy = \int_x^y (y - x)d\phi'(y) \\ &= \phi(x) - \phi(y) - (x - y)\phi'(y). \end{aligned} \tag{3.16}$$

Furthermore, since $f(x, y) \ge 0$, according to Lemma 3.2, ϕ is convex. We complete the proof. □

Theorem 3.4 is only for data points of one dimension. It is valuable to extend it to the multi-dimensional data case as follows.

Theorem 3.5 *Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set. Assume $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ is a continuously differentiable function satisfying: (1) $f(\boldsymbol{x}, \boldsymbol{x}) = 0$, $\forall\, \boldsymbol{x} \in \mathscr{S}$; (2) $f_{y_j}(\boldsymbol{x}, \boldsymbol{y})$ is continuously differentiable on x_l, $1 \le j,\ l \le d$. If f fits GD-FCM directly, there exists a continuously differentiable convex function $\phi : \mathscr{S} \mapsto \mathbb{R}$ such that*

$$f(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla\phi(\boldsymbol{y}). \tag{3.17}$$

Proof According to Lemma 3.1, since f fits GD-FCM directly, $\exists\ \lambda_k \in [0, 1]$, $\sum_{k=1}^n \lambda_k > 0$, we have

$$\sum_{k=1}^{n} \lambda_k f_{y_j}(\boldsymbol{x}_k, \boldsymbol{y}^*) = 0,\ 1 \le j \le d, \tag{3.18}$$

where $\boldsymbol{y}^* = \sum_{k=1}^n \lambda_k \boldsymbol{x}_k / \sum_{k=1}^n \lambda_k$. Similar to the limit analysis in the proof of Theorem 3.4, we have

$$f_{y_j}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{y} - \boldsymbol{x})^T \boldsymbol{h}_j(\boldsymbol{y}) = \sum_{p=1}^{d} (y_p - x_p) h_{pj}(\boldsymbol{y}),\ 1 \le j \le d, \tag{3.19}$$

where $\boldsymbol{h}_j$ is the vector of some unknown functions for f_{y_j}, and h_{pj} is the pth dimension of $\boldsymbol{h}_j$. Accordingly, for all $1 \le j, l \le d$, we have

$$f_{y_j y_l}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{p=1}^{d} (y_p - x_p)(h_{pj})_{y_l}(\boldsymbol{y}) + h_{lj}(\boldsymbol{y}), \tag{3.20}$$

$$f_{y_l y_j}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{p=1}^{d} (y_p - x_p)(h_{pl})_{y_j}(\boldsymbol{y}) + h_{jl}(\boldsymbol{y}). \tag{3.21}$$

Since $f_{y_j y_l} = f_{y_l y_j}$, and $\boldsymbol{x}$ and $\boldsymbol{y}$ can take arbitrary values in $\mathscr{S}$, we have

$$h_{jl}(\boldsymbol{y}) = h_{lj}(\boldsymbol{y}),\ 1 \le j, l \le d, \tag{3.22}$$

and

$$(h_{jp})_{y_l} = (h_{pj})_{y_l} = (h_{pl})_{y_j} = (h_{lp})_{y_j},\quad 1 \le j, l, p \le d. \tag{3.23}$$

Equation (3.23) implies that $(h_{1p}, h_{2p}, \ldots, h_{dp})^T$ is an exact differential form [11, 23], $p = 1, \ldots, d$. Therefore, there exists a function ϕ_p such that

$$\nabla\phi_p = \boldsymbol{h}_p,\ p = 1, \ldots, d. \tag{3.24}$$

Furthermore, according to Eq. (3.22),

$$\frac{\partial \phi_j}{\partial y_l} = h_{lj} = h_{jl} = \frac{\partial \phi_l}{\partial y_j}. \tag{3.25}$$

Therefore, $(\phi_1, \phi_2, \dots, \phi_d)^T$ is also an exact differential form. Accordingly, there also exists a function ϕ such that

$$\nabla \phi = (\phi_1, \phi_2, \dots, \phi_d)^T. \tag{3.26}$$

As a result, we have $\phi_{y_j y_l} = h_{lj}$, $1 \le j, l \le d$. Then Eq. (3.19) can be rewritten as

$$f_{y_j}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{y}-\boldsymbol{x})^T \nabla \phi_{y_j}(\boldsymbol{y}) = \frac{\partial}{\partial y_j}(-\phi(\boldsymbol{y})-(\boldsymbol{x}-\boldsymbol{y})^T \nabla \phi(\boldsymbol{y}) + I(\boldsymbol{x})),\ \ 1 \le j \le d.$$

Since $f(\boldsymbol{x}, \boldsymbol{x}) = 0$, we finally have

$$f(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^t \nabla \phi(\boldsymbol{y}).$$

Moreover, since $f \ge 0$, ϕ is convex according to Lemma 3.2. We complete the proof. □

Definition 3.2 (*Point-to-Centroid Distance*) Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set. A twice continuously differentiable function $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ is called a point-to-centroid distance, if there exists some higher-order continuously differentiable convex function $\phi : \mathscr{S} \mapsto \mathbb{R}$ such that

$$f(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla \phi(\boldsymbol{y}). \tag{3.27}$$

Remark Actually, the "higher-order" in Definition 3.2 is not necessary, since we only need to guarantee that $f_{y_j}(\boldsymbol{x}, \boldsymbol{y})$ is continuously differentiable on x_l, $1 \le j, l \le d$. However, considering that it is not likely that we will use a "weird" ϕ for f, the "higher-order" is indeed a comfortable request.

Corollary 3.2 *Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set. Assume $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ is a twice continuously differentiable function. Then f fits GD-FCM directly only if f is a point-to-centroid distance.*

As the widely used K-means clustering [30] is a special case of FCM when the fuzzy factor $m \to 1$, Theorem 3.5 also holds for K-means clustering, which leads to the following corollary:

Corollary 3.3 *Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set. Assume $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ is a twice continuously differentiable function. Then f fits K-means directly only if f is a point-to-centroid distance.*

3.3.2 Categorizing the Point-to-Centroid Distance

According to Definition 3.2, the point-to-centroid distance is indeed a family of distance functions which are derived from different ϕ functions. Hereinafter, we use f^{ϕ} to denote the point-to-centroid distance for simplicity. We then categorize f^{ϕ} in terms of the convexity of ϕ as follows.

Definition 3.3 (*f^{ϕ}'s Categories*) The point-to-centroid distances can be divided into two categories:

- f_I^{ϕ}: Type-I point-to-centroid distances which are derived from strictly convex ϕ functions.
- f_{II}^{ϕ}: Type-II point-to-centroid distances which are derived from convex-but-not-strictly-convex ϕ functions.

It is interesting to note that f_I^{ϕ} has the same expression as the well-known *Bregman divergence* (although we have an extra "higher-order" request for ϕ in deriving f^{ϕ}), which was first introduced by Bregman [7], and recently well studied by Banerjee et al. [1]. To further understand the differences between the two types of f^{ϕ}, we have a lemma as follows.

Lemma 3.3 *Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set, and let $f^{\phi} : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ be a point-to-centroid distance. Then ϕ is strictly convex, if and only if for any subset $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathscr{S}$, there exist $\lambda_1, \ldots, \lambda_n \in [0, 1]$ with $\sum_{k=1}^n \lambda_k > 0$ such that $\boldsymbol{y}^* = \sum_{k=1}^n \lambda_k \boldsymbol{x}_k / \sum_{k=1}^n \lambda_k$ is the unique minimizer of $\sum_{k=1}^n \lambda_k f(\boldsymbol{x}_k, \boldsymbol{y})$.*

Proof $\forall\, \boldsymbol{y} \in \mathscr{S}$, it is easy to note that

$$\Delta = \sum_{k=1}^{n} \lambda_k f(\boldsymbol{x}_k, \boldsymbol{y}) - \sum_{k=1}^{n} \lambda_k f(\boldsymbol{x}_k, \boldsymbol{y}^*) = f(\boldsymbol{y}, \boldsymbol{y}^*) \sum_{k=1}^{n} \lambda_k \geq 0. \tag{3.28}$$

By Lemma 3.3, ϕ is strictly convex $\Longleftrightarrow \forall\, \boldsymbol{y} \in \mathscr{S}$, $\boldsymbol{y} \neq \boldsymbol{y}^*$, $f(\boldsymbol{y}, \boldsymbol{y}^*) > 0 \Longleftrightarrow$ $\boldsymbol{y}^*$ is the unique minimizer of $\sum_{k=1}^n \lambda_k f(\boldsymbol{x}_k, \boldsymbol{y})$. We complete the proof. □

Remark Lemma 3.3 reveals the interrelationship between the convexity of ϕ and the uniqueness of centroids as the minimizer of J_{mf}. That is, given $\boldsymbol{U}$ fixed, the centroids computed by Eq. (3.3) constitute the unique minimizer of J_{mf}, if ϕ is strictly convex. However, if ϕ is convex-but-not-strictly-convex, J_{mf} will have multiple minimizers other than the centroids.

3.3.3 Properties of the Point-to-Centroid Distance

Here we study some important properties of the point-to-centroid distance. In particular, we pay special attention to the differences between f_I^{ϕ} and f_{II}^{ϕ}.

1. **Positive Definiteness**. f_I^ϕ is positively definite, since $f_I^\phi(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$. However, f_{II}^ϕ is merely non-negative rather than positively definite, for $\boldsymbol{x} = \boldsymbol{y}$ is only the sufficient condition for $f_{II}^\phi(\boldsymbol{x}, \boldsymbol{y}) = 0$.
2. **Symmetry**. f^ϕ is asymmetric in nature for measuring the distance from data points to centroids. However, some members of f^ϕ family do have the symmetric property due to their special ϕ functions, e.g., the squared Euclidean distance.
3. **Triangle Equality**. f^ϕ does not hold the triangle equality. Specifically, we have

$$f^\phi(\boldsymbol{x}_1, \boldsymbol{x}_2) + f^\phi(\boldsymbol{x}_2, \boldsymbol{x}_3) - f^\phi(\boldsymbol{x}_1, \boldsymbol{x}_3)$$
$$= \underbrace{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^t(\nabla\phi(\boldsymbol{x}_3) - \nabla\phi(\boldsymbol{x}_2))}_{(a)}.$$

   It is easy to show, $\forall\, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3 \in \text{dom}(\phi)$, $(a) \geq 0 \Leftrightarrow \nabla\phi(\boldsymbol{x}) = \boldsymbol{c} \Leftrightarrow \phi(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{c}' \Leftrightarrow f^\phi = 0$, where $\boldsymbol{c}$ and $\boldsymbol{c}'$ are constant vectors. Therefore, any *non-trivial* member of f^ϕ does not hold the triangle inequality, and thus is not a *metric*.
4. **Linearity**. f^ϕ is a linear operator. That is,

$$f^{\phi_1+\phi_2} = f^{\phi_1} + f^{\phi_2}, \text{ and } f^{c\phi} = cf^\phi.$$

5. **Convexity**. Given a point-to-distance function f^ϕ, we have

$$f^\phi_{x_i x_j}(\boldsymbol{x}, \boldsymbol{y}) = \phi_{x_i x_j}(\boldsymbol{x}),\ \forall\, i, j.$$

   Since ϕ is convex, f^ϕ is convex in $\boldsymbol{x}$. Furthermore, we have

$$f^\phi_{y_i y_j}(\boldsymbol{x}, \boldsymbol{y}) = \phi_{y_i y_j}(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla\phi_{y_i y_j}(\boldsymbol{y}).$$

   Therefore, f^ϕ is convex in $\boldsymbol{y} \Leftrightarrow \nabla\phi_{y_i y_j}(\boldsymbol{y}) = \boldsymbol{0} \Leftrightarrow \phi_{y_i y_j} = \boldsymbol{c}$, where $\boldsymbol{c}$ is a constant. This implies that f^ϕ is convex in $\boldsymbol{y}$ if and only if ϕ is a quadratic, linear or constant function, e.g., $\phi(\boldsymbol{y}) = \|\boldsymbol{y}\|^2$.

3.4 The Global Convergence of GD-FCM

In Sect. 3.3.1, we derive f^ϕ and show that if a function fits GD-FCM directly, it must be a point-to-centroid distance. In this section, we continue to prove the sufficient condition. That is, all the point-to-centroid distances indeed fit GD-FCM directly. To that end, recall Definition 3.1, we need to prove that for GD-FCM using f^ϕ, the iteration sequence (or one of its subsequences) generated by T_m has a global convergence. Note that our proof below is influenced by [2, 16], which established the convergence theorem for FCM using the squared Euclidean distance. We begin by the f_I^ϕ case.

Let $\mathscr{S} \subseteq \mathbb{R}^d$ be a nonempty open convex set, and $f_I^\phi : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ be a Type-I point-to-centroid distance used in GD-FCM. Let $\mathscr{X} = \{\boldsymbol{x}_k\}_{k=1}^n \subset \mathscr{S}$ contain $n > c$ distinct points. Fix the fuzzy factor $m > 1$. For $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c)^T$, we say $\boldsymbol{V} \in \mathscr{S}^c$ if and only if $\boldsymbol{v}_i \in \mathscr{S}$ for all $1 \le i \le c$. We first have the following lemmas.

Lemma 3.4 *Fix $\boldsymbol{V} \in \mathscr{S}^c$. Then the set of global minimizers of $J_{mf}(\boldsymbol{U})$ is the subset of M_{fc} satisfying*

$$u_{ik}^* = \frac{(d_{ik})^{-1/(m-1)}}{\sum_{l=1}^c (d_{lk})^{-1/(m-1)}}, \text{ if } I_k = \emptyset, \quad \textit{and} \tag{3.29a}$$

$$u_{ik}^* = 0 \,\forall\, i \in \tilde{I}_k \textit{ and } \sum_{i \in I_k} u_{ik}^* = 1, \text{ if } I_k \neq \emptyset, \tag{3.29b}$$

where $d_{ik} = f_I^\phi(\boldsymbol{x}_k, \boldsymbol{v}_i)$, $\forall\, i,\ k$, and $I_k = \{i | d_{ik} = 0,\ 1 \le i \le c\}$ and $\tilde{I}_k = \{1, \ldots, c\} \setminus I_k$, $\forall\, k$.

Proof Since the proof has nothing to do with the specific form of d_{ik}, we only need to prove that the lemma holds for the square Euclidean distance, which indeed has been given by Proposition 1 of [16]. □

Lemma 3.5 *Fix $\boldsymbol{U} \in M_{fc}$, then $\boldsymbol{V}^* = (\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_c^*)^T$ satisfying*

$$\boldsymbol{v}_i^* = \sum_{k=1}^n (u_{ik})^m \boldsymbol{x}_k / \sum_{k=1}^n (u_{ik})^m, \quad 1 \le i \le c, \tag{3.30}$$

is the unique global minimizer of $J_{mf}(\boldsymbol{V})$.

Proof Let $\Delta = J_{mf}(\boldsymbol{V}) - J_{mf}(\boldsymbol{V}^*)$. Given $f_I^\phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - (\boldsymbol{x} - \boldsymbol{y})^T \nabla_{\boldsymbol{y}} \phi(\boldsymbol{y})$, according to Eq. (3.6), we have

$$\Delta = \sum_{i=1}^c \sum_{k=1}^n (\mu_{ik})^m (\phi(\boldsymbol{v}_i^*) - \phi(\boldsymbol{v}_i)) - A + B, \tag{3.31}$$

where $A = \sum_{k=1}^n (\mu_{ik})^m (\boldsymbol{x}_k - \boldsymbol{v}_i)^T \nabla\phi(\boldsymbol{v}_i)$ and $B = \sum_{k=1}^n (\mu_{ik})^m (\boldsymbol{x}_k - \boldsymbol{v}_i^*)^T \nabla\phi(\boldsymbol{v}_i^*))$. Since $\boldsymbol{v}_i^*$ is the $(\mu_{ik})^m$-weighted arithmetic mean of $\boldsymbol{x}_k$ $(1 \le k \le n)$ in Eq. (3.30), we have $A = \sum_{k=1}^n (\mu_{ik})^m (\boldsymbol{v}_i^* - \boldsymbol{v}_i)^T \nabla\phi(\boldsymbol{v}_i)$ and $B = 0$. As a result,

$$\Delta = \sum_{i=1}^{c}\sum_{k=1}^{n}(\mu_{ik})^m(\phi(\boldsymbol{v}_i^*) - \phi(\boldsymbol{v}_i) - (\boldsymbol{v}_i^* - \boldsymbol{v}_i)^T\nabla\phi(\boldsymbol{v}_i))$$
$$= \sum_{i=1}^{c} f_I^{\phi}(\boldsymbol{v}_i^*, \boldsymbol{v}_i)\sum_{k=1}^{n}(\mu_{ik})^m.$$

Since $\sum_{k=1}^{n}(u_{ik})^m > 0$ and $f_I^{\phi} \geq 0$, we have $\Delta \geq 0$, which implies that $\boldsymbol{V}^*$ is a global minimizer. Furthermore, since ϕ is strictly convex, we have $\forall\ \boldsymbol{V} \neq \boldsymbol{V}^*,\ f_I^{\phi}(\boldsymbol{v}_i, \boldsymbol{v}_i^*) > 0 \Rightarrow \Delta > 0$, which indicates that $\boldsymbol{V}^*$ is the only global minimizer. □

Based on the above lemmas, we can formally define the T_{mf} mapping for GD-FCM as follows:

Definition 3.4 (*T_{mf} for GD-FCM*) Let G be the function defined by $G(\boldsymbol{U}) = \boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c)^T$, where $\boldsymbol{v}_i$ satisfies Eq. (3.30), $1 \leq i \leq c$. Let F denote the point-to-set mapping defined by $F(\boldsymbol{V}) = \{\boldsymbol{U}|\boldsymbol{U}$ satisfies Eq. (3.29)$\}$. Then the GD-FCM iteration can be described by the point-to-set mapping T_{mf} as follows:

$$T_{mf}(\boldsymbol{U}, \boldsymbol{V}) = \{(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}})|\hat{\boldsymbol{V}} = G(\boldsymbol{U}),\ \hat{\boldsymbol{U}} \in F(\hat{\boldsymbol{V}})\}. \tag{3.32}$$

For the range of T_{mf}, we have the following lemma:

Lemma 3.6 *Let $\boldsymbol{U}^{(0)} \in M_{fc}$, $\boldsymbol{V}^{(0)} = G(\boldsymbol{U}^{(0)})$, and $(\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}) \in T_{mf}(\boldsymbol{U}^{(l-1)}, \boldsymbol{V}^{(l-1)})$, $l = 1, 2, \ldots$. Then $\forall\ l$, $(\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}) \in P(M_{fc} \times \mathscr{S}^c)$.*

Proof First assume $\boldsymbol{U}^{(l)} \in P(M_{fc})$, $l = 1, 2, \ldots$. According to Eq. (3.30),

$$\boldsymbol{v}_i^{(l)} = \sum_{k=1}^{n} w_{ik}\boldsymbol{x}_k,\ 1 \leq i \leq c, \tag{3.33}$$

where $w_{ik} = (u_{ik}^{(l)})^m / \sum_{k=1}^{n}(u_{ik}^{(l)})^m$. Therefore, we have $w_{ik} \in [0, 1]$ and $\sum_{k=1}^{n} w_{ik} = 1$, which implies that $\boldsymbol{v}_i^{(l)} \in \text{Conv}(\mathscr{X}) \subseteq \mathscr{S} \Rightarrow \boldsymbol{V}^{(l)} \in \mathscr{S}^c$. So it remains to show that $\boldsymbol{U}^{(l)} \in P(M_{fc})$.

It is obvious that $\boldsymbol{U}^{(l)}$ in Eq. (3.29) satisfies Constraints 3.2a, c of M_{fc}. To meet Constraint 3.2b, we should guarantee that $\boldsymbol{U}^{(l)}$ will not degenerate. Suppose $\exists\ 1 \leq r \leq c$ such that $(u_{rk})^{(l)} = 0\ \forall\ k$. As we know, given a specific k, if $(d_{ik})^{(l)} > 0\ \forall\ i$, then $(u_{rk})^{(l)} > 0$, which contradicts the degeneration assumption. So $\forall\ k$, $\exists\ i$ such that $(d_{ik})^{(l)} = f_I^{\phi}(\boldsymbol{x}_k, (\boldsymbol{v}_i)^{(l)}) = 0$, which implies that $(\boldsymbol{v}_i)^{(l)} = \boldsymbol{x}_k$ for ϕ is strictly convex. Recall that $\mathscr{X}$ contains $n > c$ *distinct* points, we should therefore have n distinct centroids, which leads to the contradiction. We complete the proof. □

Remark Lemma 3.6 shows that for f_I^{ϕ}, $T_{mf} : M_{fc} \times \mathscr{S}^c \mapsto P(M_{fc} \times \mathscr{S}^c)$, with $G : M_{fc} \mapsto \mathscr{S}^c$ and $F : \mathscr{S}^c \mapsto P(M_{fc})$. This is crucial for the proof of the closeness of T_{mf} below.

Then we continue to prove that T_{mf} decreases J_{mf} monotonically. More importantly, the decrease is strict for $(\boldsymbol{U}, \boldsymbol{V})$ not in the solution set. We have the following descent theorem.

Theorem 3.6 *Let* $\Omega = \{(\boldsymbol{U}^*, \boldsymbol{V}^*) \in M_{fc} \times \mathscr{S}^c |$

$$J_{mf}(\boldsymbol{U}^*, \boldsymbol{V}^*) \le J_{mf}(\boldsymbol{U}, \boldsymbol{V}^*),\ \forall\, \boldsymbol{U} \in M_{fc},\ \text{and} \tag{3.34}$$

$$J_{mf}(\boldsymbol{U}^*, \boldsymbol{V}^*) < J_{mf}(\boldsymbol{U}^*, \boldsymbol{V}),\ \forall\, \boldsymbol{V} \in \mathscr{S}^c,\ \boldsymbol{V} \ne \boldsymbol{V}^*\} \tag{3.35}$$

be the solution set, and let $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}) \in M_{fc} \times \mathscr{S}^c$. *Then* $\forall\, (\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) \in T_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$, $J_{mf}(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) \le J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$, *with the strictness in the inequality if* $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}) \notin \Omega$.

Proof According to Lemmas 3.4 and 3.5, $\forall\, (\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) \in T_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$ we have

$$J_{mf}(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) \le J_{mf}(\bar{\boldsymbol{U}}, \hat{\boldsymbol{V}}) \le J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}). \tag{3.36}$$

Suppose $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}) \notin \Omega$, which means that either Eq. (3.34) or Eq. (3.35) is violated. Without loss of generality, assume Eq. (3.35) does not hold. It follows that $\exists\, \boldsymbol{V}' \in \mathscr{S}^c$ and $\boldsymbol{V}' \ne \bar{\boldsymbol{V}}$, $J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}) \ge J_{mf}(\bar{\boldsymbol{U}}, \boldsymbol{V}')$. According to Lemma 3.5, given $\boldsymbol{U} = \bar{\boldsymbol{U}}$ fixed, $\hat{\boldsymbol{V}}$ is the unique global minimizer, which implies that $J_{mf}(\bar{\boldsymbol{U}}, \hat{\boldsymbol{V}}) < J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$. We then have $J_{mf}(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) < J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$ accordingly.

Now assume Eq. (3.35) holds but Eq. (3.34) does not hold. We have $\bar{\boldsymbol{V}} = \hat{\boldsymbol{V}}$. By the similar reasoning, we have $\exists \boldsymbol{U}' \in M_{fc},\ J_{mf}(\boldsymbol{U}', \hat{\boldsymbol{V}}) < J_{mf}(\bar{\boldsymbol{U}}, \hat{\boldsymbol{V}})$. By Lemma 3.4, we further have $J_{mf}(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) \le J_{mf}(\boldsymbol{U}', \hat{\boldsymbol{V}})$, which implies that $J_{mf}(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) < J_{mf}(\bar{\boldsymbol{U}}, \hat{\boldsymbol{V}}) = J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$. □

Theorem 3.7 *The point-to-set mapping* $T_{mf} : M_{fc} \times \mathscr{S}^c \mapsto P(M_{fc} \times \mathscr{S}^c)$ *is closed at every point in* $M_{fc} \times \mathscr{S}^c$.

Proof If there is no singularity, we have $T_{mf} : M_{fc} \times \mathscr{S}^c \mapsto M_{fc} \times \mathscr{S}^c$, and F is continuous on $\mathscr{S}^c$. Furthermore, since G is continuous on M_{fc}, we have T_{mf} is continuous on $M_{fc} \times \mathscr{S}^c$, which indicates that T_{mf} is closed on $M_{fc} \times \mathscr{S}^c$. For the case that the singularity happens, the proof is more complicated. According to Zangwill's Corollary in Sect. 3.2, we should prove that F is closed on $\mathscr{S}^c$. In Theorem 2 of [16], Hathaway et al. provided a proof for the squared Euclidean distance case. The proof can be adapted to our general case using f_I^{ϕ}. So we omit it here. □

Theorem 3.8 *Let* $(\boldsymbol{U}^{(0)}, \boldsymbol{V}^{(0)})$, $\boldsymbol{U}^{(0)} \in M_{fc}$ *and* $\boldsymbol{V}^{(0)} = G(\boldsymbol{U}^{(0)})$, *be the starting point of iteration with* T_{mf}. *Then the iteration sequence* $\{(\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)})\}$, $l = 1, 2, \ldots$ *is contained in a compact subset of* $M_{fc} \times \mathscr{S}^c$.

Proof By Lemma 3.6, $\boldsymbol{V}^{(l)} \in \text{Conv}(\mathscr{X})^c \subseteq \mathscr{S}^c$. So $(\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}) \in M_{fc} \times \text{Conv}(\mathscr{X})^c$, $l = 1, 2, \ldots$. Since $\mathscr{X}$ has finite points, $\text{Conv}(\mathscr{X})$ is bounded and closed, and thus $\text{Conv}(\mathscr{X})^c$ is compact. Moreover, since $u_{ik} \in [0, 1]\ \forall\, i, k$, M_{fc} is bounded. So it remains to show M_{fc} is closed. In T6.2 of [3], the author proved that $M_{fco} = \text{Conv}(M_{co})$, where M_{fco} is the superset of M_{fc} obtained by relaxing Constraint (3.2b), and M_{co} is the hard version of M_{fco} obtained by letting $u_{ik} \in \{0, 1\}$ instead. As pointed out in Lemma 3.6, $\boldsymbol{U}$ is nondegenerate, so $M_{fc} = M_{fco}$, we therefore have $M_{fc} = \text{Conv}(M_{co})$. Then by an argument similar to $\boldsymbol{V}^{(l)}$ above, we can establish the compactness of M_{fc}. □

Now, assembling the descent property and the closeness of T_{mf}, and the compactness of the domain, we can establish the global convergence theorem for GD-FCM. Let Ω be the solution set defined in Theorem 3.6, we have:

Theorem 3.9 *Let $(\boldsymbol{U}^{(0)}, \boldsymbol{V}^{(0)})$, $\boldsymbol{U}^{(0)} \in M_{fc}$ and $\boldsymbol{V}^{(0)} = G(\boldsymbol{U}^{(0)})$, be the starting point of iteration with T_{mf}. Then the iteration sequence $((\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}))_{l=0}^{\infty}$, or one of its subsequences converges to a point in Ω.*

Proof According to Theorems 3.6, 3.7 and 3.8, and the Zangwill's global convergence theorem, we have the global convergence of GD-FCM. □

Now we extend the above convergence results to the f_{II}^{ϕ} case. That is, for GD-FCM using f_{II}^{ϕ}, *if the degeneration of* $\boldsymbol{U}$ *does not happen*, we state without proof the following facts:

1. Lemma 3.4 holds no matter what the convexity of ϕ is, i.e., strictly convex or convex but not strictly convex.
2. In Lemma 3.5, $\boldsymbol{V}^*$ is no longer "the unique minimizer" but only one of the global minimizers. Accordingly, the solution set in the descent theorem (Theorem 3.6) should be modified to accommodate this change. Details can be found in 4) below.
3. Lemma 3.6 still holds for the nondegeneration assumption.
4. Since $\boldsymbol{V}^*$ is no longer the unique minimizer in (2), to keep the validity of the descent theorem (Theorem 3.6), we should modify the solution set Ω to

$$\begin{aligned} \Omega' = \{(\boldsymbol{U}^*, \boldsymbol{V}^*) \in M_{fc} \times \mathscr{S}^c |\\ J_{mf}(\boldsymbol{U}^*, \boldsymbol{V}^*) \le J_{mf}(\boldsymbol{U}, \boldsymbol{V}^*),\ \forall\, \boldsymbol{U} \in M_{fc},\ \text{and}\\ J_{mf}(\boldsymbol{U}^*, \boldsymbol{V}^*) \le J_{mf}(\boldsymbol{U}^*, \boldsymbol{V}),\ \forall\, \boldsymbol{V} \in \mathscr{S}^c\}. \end{aligned} \tag{3.37}$$

   The proof is similar to the proof of Theorem 3.6, so we omit it here.
5. The closeness of T_{mf} in Theorem 3.7 and the compactness of $M_{fc} \times \text{Conv}(\mathscr{X})$ in Theorem 3.8 still hold.
6. By (4) and (5), using the Zangwill's convergence theorem, we again have the global convergence theorem for GD-FCM using f_{II}^{ϕ}.

Note that the above reasoning is based on the assumption that there is no degeneration. In fact, in Lemma 3.6, we cannot guarantee the nondegeneration of $\boldsymbol{U}$ in theory due to the non-strict convexity of ϕ. Nevertheless, the occurrence probability of degeneration is yet very low for real-world data sets, since given $n \gg c$ it is very easy to find an $\boldsymbol{x}_k \in \mathscr{X}$ such that $f(\boldsymbol{x}_k, \boldsymbol{v}_i^{(l)}) > 0 \; \forall \, i$. As a result, in most cases, we can safely have $T_{mf} : M_{fc} \times \mathscr{S}^c \mapsto P(M_{fc} \times \mathscr{S}^c)$ regardless of the convexity of ϕ. At the very least, if the degeneration does happen, we still have a *remedy* for it, which can also guarantee the descent theorem, the closeness of T_{mf} and the compactness of the domain. As a result, the global convergence still holds. We detail this in Appendix.

Finally, we have the necessary and sufficient conditions for a distance function possessing the direct fitness to GD-FCM as follows:

Theorem 3.10 *Let $\mathscr{S}$ be a nonempty open convex set. Assume any data set to be clustered is a subset of $\mathscr{S}$, i.e., $\mathscr{X} \subset \mathscr{S}$. Then a distance function $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ fits GD-FCM directly if and only if f is a point-to-centroid distance.*

Theorem 3.10 also holds for the K-means clustering. Actually, the global convergence of K-means clustering using the point-to-centroid distance is much more straightforward. Let us revisit the two-phase iteration process of K-means. In the assignment phase, instances are assigned to the closest centroids, so the objective function value of K-means will decrease. Then in the update phase, centroids are updated using the assigned instances, which will continue to decrease the objective function value according to Lemma 3.5. To sum up, the iteration process of K-means will decrease the objective function value consistently. Considering that there are limited combinations of the assignments of instances, K-means using the point-to-centroid distance will finally converge to a local minimum or a saddle point in a finite number of iterations. In case of the degeneration, the remedy method introduced in the appendix of this chapter is also suitable for K-means clustering, which still guarantees the global convergence of K-means. We therefore have the following corollary:

Corollary 3.4 *Let $\mathscr{S}$ be a nonempty open convex set. Assume any data set to be clustered is a subset of $\mathscr{S}$, i.e., $\mathscr{X} \subset \mathscr{S}$. Then a distance function $f : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ fits K-means clustering directly if and only if f is a point-to-centroid distance.*

3.5 Examples of the Point-to-Centroid Distance

Here, we show some examples of point-to-centroid distance using different ϕ functions.

Example 3.1 Let $\mathscr{S} = \mathbb{R}^d$. Let $\phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^2$ with $\mathrm{dom}(\phi) = \mathscr{S}$. We have the well-known *squared Euclidean distance* $d^2 : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ as follows:

$$d^2(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{3.38}$$

Example 3.2 Let $\mathscr{S} = \{\boldsymbol{x} | \boldsymbol{x} \in \mathbb{R}^d_{++}, \sum_{j=1}^d x_j = 1\}$. Let $\phi(\boldsymbol{x}) = \sum_{j=1}^d x_j \log x_j$ with $\mathrm{dom}(\phi) = \mathscr{S}$. We have the well-known *KL-divergence* or *relative entropy* $D : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ as follows:

$$D(\boldsymbol{x} \| \boldsymbol{y}) = \sum_{j=1}^{d} x_j \log \frac{x_j}{y_j}. \tag{3.39}$$

It is easy to show that, d^2 and D are the distance functions of f_I^{ϕ}, for their ϕ functions are strictly convex. In what follows, we introduce some examples of f_{II}^{ϕ}.

Example 3.3 Let $\mathscr{S} = \mathbb{R}^d_{++}$. Let $\phi(\boldsymbol{x}) = \|\boldsymbol{x}\|$ with $\mathrm{dom}(\phi) = \mathscr{S}$. We have a distance $f_{cos} : \mathscr{S} \times \mathscr{S} \mapsto \mathbb{R}_+$ as follows:

$$f_{cos}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\| - \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{y}\|} = \|\boldsymbol{x}\|(1 - \cos(\boldsymbol{x}, \boldsymbol{y})). \tag{3.40}$$

In many applications, data points are first initialized to have unit length, i.e., $\boldsymbol{x} \leftarrow \boldsymbol{x}/\|\boldsymbol{x}\|$, before clustering. In that case, f_{cos} reduces to $1 - \cos(\boldsymbol{x}, \boldsymbol{y})$, which is equivalent to the well-known *cosine similarity*. Note that cosine similarity has long been treated as an angle-based similarity rather than a distance function [42, 53]. In this sense, the point-to-centroid distance provides a general framework for unifying some well-known distance-based and similarity-based proximity functions.

Example 3.4 Let $\mathscr{S} = \mathbb{R}^d_{++}$. Let $\phi(\boldsymbol{x}) = (\sum_{l=1}^d w_l x_l^p)^{1/p}$ with $\mathrm{dom}(\phi) = \mathscr{S}$, $p > 1$ and $w_l > 0$, $1 \le l \le d$. We have a general distance f_{l_p} derived from the weighted l_p norm as follows:

$$f_{l_p}(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) - \frac{\sum_{l=1}^d w_l x_l (y_l)^{p-1}}{(\phi(\boldsymbol{y}))^{p-1}}. \tag{3.41}$$

Since l_p norm is a convex function on $\mathbb{R}^d$ for $1 \le p < +\infty$ [8, 49], f_{l_p} is a point-to-centroid distance. Further, it is easy to show that $f_{l_p}(\boldsymbol{x}, \boldsymbol{y}) = f_{l_p}(\boldsymbol{x}, a\boldsymbol{y})\ \forall\, a > 0$, which indicates that the centroids are not the unique minimizer of J_{mf} using f_{l_p}. So by Lemma 3.3, ϕ is a convex-but-not-strictly-convex function, and f_{l_p} is a distance function of f_{II}^{ϕ}. Also, it is interesting to note that, $f_{l_2} \equiv f_{cos}$ when $p = 2$ and $w_l = 1$, $1 \le l \le d$. In other words, f_{cos} is a special case of f_{l_p}. To understand this, assume $\boldsymbol{x}$ and $\boldsymbol{y}$ are linearly dependent, i.e., $\boldsymbol{x} = a\boldsymbol{y}$, $a > 0$. Then we have $f_{l_p}(\boldsymbol{x}, \boldsymbol{y}) = 0$. This implies that like f_{cos}, f_{l_p} also takes the angle into consideration when measuring the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Discussion. In theory, Type-I and Type-II point-to-centroid distances are derived from ϕ with different convex properties. Also, employing different distances for GD-FCM may result in substantially-different clustering results (see the experimental results below). Nonetheless, from a user perspective, it has no difference in using a

Table 3.1 Some characteristics of experimental data sets

Data	Source	#Objects	#Atrributes	#Classes	MinClassSize	MaxClassSize	CV
Breast-w	UCI	699	9	2	241	458	0.44
Ecoli	UCI	336	7	8	2	143	1.16
Glass	UCI	214	9	6	9	76	0.83
Housing	UCI	506	13	229	1	16	0.76
Iris	UCI	150	4	3	50	50	0.00
Pageblocks	UCI	5473	10	5	28	4913	1.95
Pendigits	UCI	10992	16	10	1055	1143	0.04
Wine	UCI	178	13	3	48	71	0.19
Dermatology	UCI	358	34	6	20	112	0.51
Libras	UCI	360	90	15	24	24	0.00
Satimage	UCI	4435	36	6	415	1072	0.43

Type-I or Type-II distance for fuzzy c-means clustering, since the clustering process is the same for the two types of distances, and the computations of centroids are exactly the same by Eq. (3.30).

3.6 Experimental Results

In this section, we present experimental results to show the clustering performance of GD-FCM. Specifically, we demonstrate: (1) The global convergence of GD-FCM; (2) The necessity of GD-FCM in providing diversified distance functions.

3.6.1 Experimental Setup

We first introduce the experimental setup, including the information of the data, the clustering tools, the distance functions, and the evaluation measures.

Experimental data. In the experiments, we use a number of real-world and synthetic data sets. The details of real-world data sets are listed in Table 3.1. Note that CV here denotes the coefficient of variation of the class sizes, which reveals the degree of class imbalance in data. The 1-dimensional synthetic data sets are generated by the mixture models of four distributions, namely the normal distribution, the binomial distribution, the Poisson distribution, and the Gamma distribution. Figure 3.2 shows the probability density functions and the parameters of the four distributions, respectively. For each distribution, we generate a mixture model of three classes, each of which contains 500 objects sampled from one of the three models. The 2-dimensional synthetic data sets are all generated by the mixture models of the Gaussian distribution but with different parameters, as shown in Fig. 3.3. Also, for each distribution,

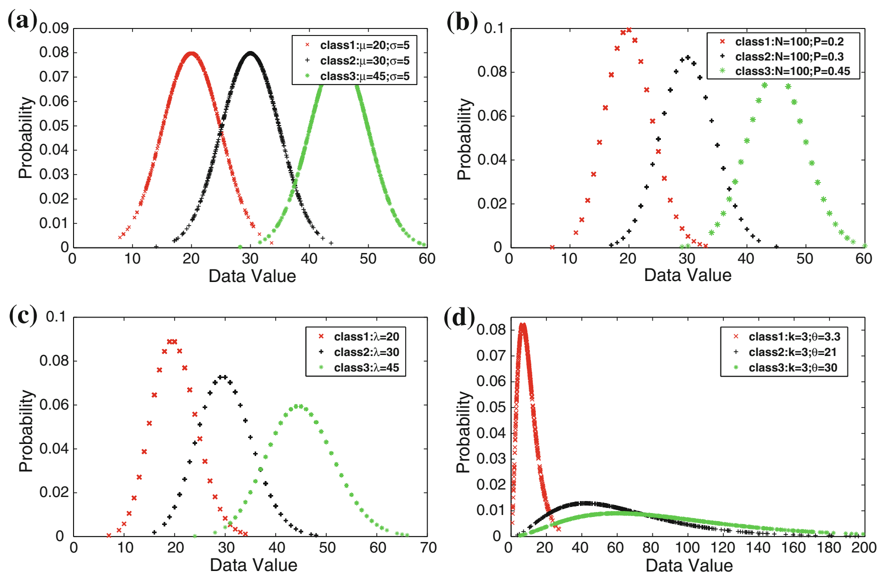


Fig. 3.2 The probability density distribution of 1-dimensional synthetic data. **a** Normal Distribution **b** Binomial Distribution **c** Poisson Distribution **d** Gamma Distribution. © 2012 IEEE. Reprinted, with permission, from Ref. [46]

we generate a mixture model of three classes, each of which contains 1000 objects sampled from one of the three models.

Clustering tools. We coded GD-FCM using MATLAB. Some notable details are as follows. First, in the implementation, we provided three initialization schemes, namely the random membership grade scheme [3], the random centroid (selected from data instances) scheme, and the user-specified centroid scheme. Unless otherwise specified, the first scheme is the default one in the experiments. Second, we adopted the stopping criterion suggested by Bezdek [3]. In other words, if $\|\boldsymbol{U}^{(l+1)} - \boldsymbol{U}^{(l)}\| \leq \epsilon$, the procedure suspends, where $\|\boldsymbol{U}\|$ is a Frobenius matrix norm [12], and ϵ is a user defined threshold with a default value 10^{-6}. Third, when multiple singularities occur in updating the membership grade matrix, we evenly assign the membership grades to all the data points coinciding at the centroid. That is, for the case that $I_k \neq \emptyset$ in Eq. (3.29b), we have $u_{ik} = 1/|I_k|$, $\forall i \in I_k$. Finally, the GD-FCM implementation was run in the environment of MATLAB R2010a, on a Microsoft Windows Server 2008R2 Standard platform with SP2 32bit edition. The experimental PC is with an Intel Core i7-930 2.8GHz×4 CPU, 4GB DDRIII 1600MHz RAM, and a 7200 RPM 32MB cache 1TB SATAII hard disk.

Distance functions. Undoubtedly we cannot test all the point-to-centroid distances in the experiments. Table 3.2 lists the distance functions we use for the experiments. Note that BD and PD are distance functions derived from the widely used 1-D binomial distribution and 1-D Poisson distribution, respectively. So they are for

Table 3.2 The distance functions used in the experiments

ID	Distance	Notation	$\phi(\boldsymbol{x})$	dom(ϕ)	$f^{\phi}(\boldsymbol{x}, \boldsymbol{y})$
1	Binomial distance	BD	$x \ln(\frac{x}{N}) +$ $(N - x) \ln(\frac{N-x}{N})$	$[0, N],\ N \in \mathbb{Z}_{++}$	$x \ln(\frac{x}{y}) + (N - x) \ln(\frac{N-x}{N-y})$
2	Poisson distance	PD	$x \ln x - x$	$\mathbb{R}_{++}$	$x \ln(\frac{x}{y}) - (x - y)$
3	Squared Euclidean distance	d^2	$\lVert\boldsymbol{x}\rVert^2$	$\mathbb{R}^d,\ d \in \mathbb{Z}_{++}$	$\lVert\boldsymbol{x} - \boldsymbol{y}\rVert^2$
4	KL-divergence	D	$\sum_{j=1}^{d} x_j \log x_j$	$\{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}_{++}^d, \sum_{j=1}^{d} x_j = 1\}$, $d - 1 \in \mathbb{Z}_{++}$	$\sum_{j=1}^{d} x_j \log \frac{x_j}{y_j}$
5	Cosine distance	f_{cos}	$\lVert\boldsymbol{x}\rVert$	$\{\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}_{++}^d, \lVert\boldsymbol{x}\rVert = 1\}$, $d - 1 \in \mathbb{Z}_{++}$	$1 - \cos(\boldsymbol{x}, \boldsymbol{y})$
6	l_p distance	f_{l_p}	$(\sum_{l=1}^{d} x_l^p)^{1/p}$	$\mathbb{R}_{++}^d,\ d - 1 \in \mathbb{Z}_{++},\ p > 1$	$\phi(\boldsymbol{x}) - \frac{\sum_{l=1}^{d} x_l (y_l)^{p-1}}{(\phi(\boldsymbol{y}))^{p-1}}$

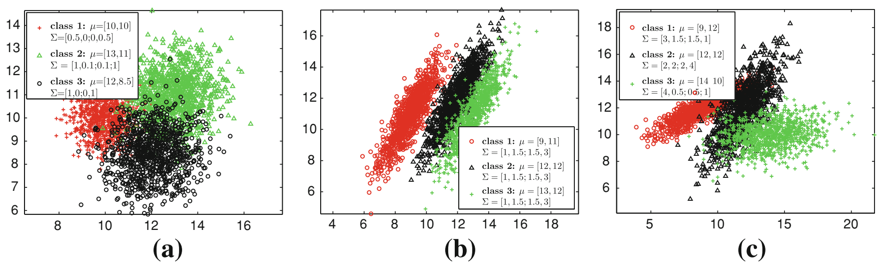


Fig. 3.3 The distribution of 2-dimensional synthetic data. **a** Gaussian Distribution I **b** Gaussian Distribution II **c** Gaussian Distribution III. © 2012 IEEE. Reprinted, with permission, from Ref. [46]

1-dimensional data ($d = 1$) only, and D (KL-divergence), $f_{\cos}$ (cosine distance) and f_{l_p} are distance functions for multi-dimensional data ($d \geq 2$) only. As a widely used distance function, d^2 (squared Euclidean distance) works for data of any dimensionality.

Validation measures. As we have class labels for both the synthetic and real-world data, we use the external measure for cluster validity [41]. In the literature, it has been proved that the normalized Variation of Information (VI_n) measure shows merits in evaluating the clustering performance of K-means [31, 45], which is also carefully examined by our study in Chap. 5. Here we adapt it to the fuzzy c-means case. Suppose we want to cluster the data set $\mathscr{X}$ of n objects and c' classes into c clusters using GD-FCM. For the membership grade matrix $\boldsymbol{U}^*$, we have the summarized matrix $\boldsymbol{Z}^*$ as follows:

$$z^*_{ij} = \sum_{x_k \in C'_j} u^*_{ik},\ 1 \leq i \leq c,\ \ 1 \leq j \leq c',\ \ 1 \leq k \leq n, \tag{3.42}$$

where C'_j denotes the class j. Let $z^*_{i\cdot} = \sum_{j=1}^{c'} z^*_{ij}$, $z^*_{\cdot j} = \sum_{i=1}^{c} z^*_{ij}$, $p_{i\cdot} = z^*_{i\cdot}/n$, $p_{\cdot j} = z^*_{\cdot j}/n$, and $p_{ij} = z^*_{ij}/n$. Then we have

$$VI_n = \frac{H(C|C') + H(C'|C)}{H(C) + H(C')} = 1 + \frac{2\sum_{i=1}^{c}\sum_{j=1}^{c'} p_{ij}\log(p_{ij}/p_{i\cdot}p_{\cdot j})}{\sum_{i=1}^{c} p_{i\cdot}\log p_{i\cdot} + \sum_{j=1}^{c'} p_{\cdot j}\log p_{\cdot j}}. \tag{3.43}$$

Obviously, $VI_n \in [0, 1]$, and a larger VI_n value indicates a poorer clustering performance. Note that VI_n differentiates itself from the traditional fuzzy measures such as Partition Coefficient (PC) and Classification Entropy (CE) by using the class information in Eq. (3.42).

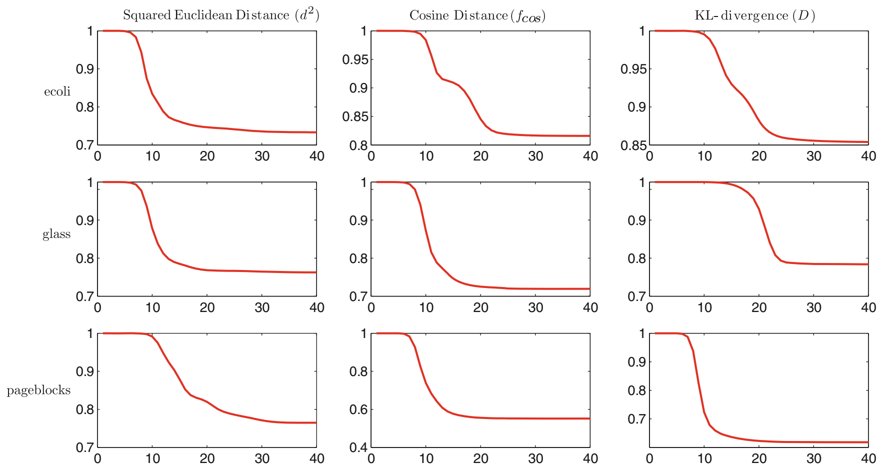


Fig. 3.4 An illustration of the convergence of GD-FCM using d^2, $f_{\cos}$, D. © 2012 IEEE. Reprinted, with permission, from Ref. [46]

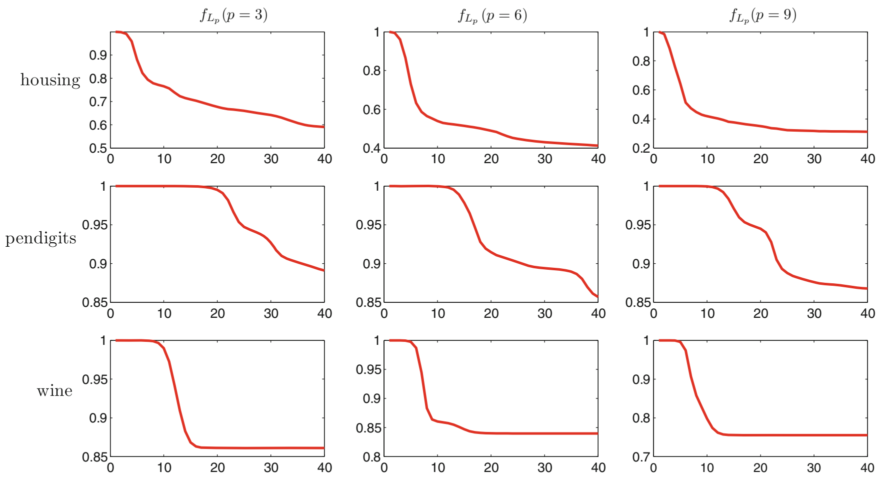


Fig. 3.5 An illustration of the convergence of GD-FCM using f_{l_p}. © 2012 IEEE. Reprinted, with permission, from Ref. [46]

3.6.2 The Global Convergence of GD-FCM

Here, we illustrate the global convergence of GD-FCM by observing the decreasing trend of the objective function values using various point-to-centroid distances.

Specifically, we first use GD-FCM to perform fuzzy clustering on 11 real-world data sets listed in Table 3.1. Since these data sets are all multi-dimensional, we use four point-to-centroid distance functions, i.e., d^2, D, f_{cos} and f_{l_p}, for this experiment, with the former two are Type-I P2C-D and the latter ones are Type-II P2C-D. To avoid the potential poor convergence introduced by randomizing the membership grade matrix in the initialization, we repeat clustering 10 times for each data set, and return the best one as the result. Other mentionable details are as follows: (1) Five data sets, namely `glass`, `housing`, `pageblocks`, `pendigits` and `wine`, are normalized to have feature values in [0,1] before clustering; (2) For each clustering, the maximum number of iterations is set to 40, the fuzzy factor is set to 2, and the number of clusters is set equally to the number of classes.

Figures 3.4 and 3.5 show the movements of the objective function values along the iteration process. Here the Y-axis represents the relative convergence rate of the objective function, which is calculated by having $J_{mf}^{(l)}/J_{mf}^{(1)}$ for the lth iteration. As can be seen in Fig. 3.4, all the red lines have a very similar trend; that is, they drop continuously regardless of the data set and the distance function used. Indeed, we find this trend is ubiquitous for all the $11 * 10 * 3 = 330$ runs of GD-FCM on 11 data sets. This indicates that GD-FCM has a global convergence using d^2, D and f_{cos}—the three most widely used distance functions for real-world data clustering.

Figure 3.5 further demonstrates the convergence property of GD-FCM using the f_{l_p} distance. A similar decreasing trend can be observed in the curves of Fig. 3.5. This indicates the continuous decreases of the objective function values. As mentioned in Sect. 3.4, f_{l_p} is derived by the convex-but-not-strictly-convex l_p norm and thus may lead GD-FCM to the dangerous degeneration case. However, this does not happen for our $11 \times 10 \times 3 = 330$ runs on 11 data sets using three f_{l_p} distances. This agrees with our analysis that GD-FCM with f_{l_p} usually leads to a global convergence for real-world data, although there is a very low degeneration risk in theory.

If we take a closer look at the convergence processes in Fig. 3.4, we can find that the convergence speed of GD-FCM is quite satisfying for the distance functions d^2, D and f_{cos}. That is, GD-FCM usually achieves the largest part of decrease of the objective function value within 30 iterations. The convergence speed of GD-FCM using f_{l_p} seems to be a bit lower in Fig. 3.5. For instance, when $p = 6$, the objective function value experiences a much longer decreasing process for the `pendigits` data set, which is finally stopped by the maximum iteration criterion. Nonetheless, it does not necessarily mean that GD-FCM with f_{l_p} tends to produce poorer clustering results. We will detail this in the following section.

3.6.3 The Merit of GD-FCM in Providing Diversified Distances

In this section, we demonstrate the importance of choosing the right distance functions when using GD-FCM for different data sets.

Two types of data sets are used for this experiments. The first type is the artificial data sets which we have introduced in Sect. 3.6.1. For the 1-dimensional data sets illustrated in Fig. 3.2, we have clustering results evaluated by VI_n in Table 3.3. As can be seen, although being most widely used, d^2 is not an all-purpose distance for

Table 3.3 Clustering results of synthetic data (measure: VI_n)

1-D distributions	d^2	BD	PD
Normal	**0.5522**	0.5561	0.5562
Binomial	0.5167	**0.5158**	0.5159
Poisson	0.6089	0.6060	**0.6059**
Gamma	0.7285	0.6745	**0.6734**
2-D distributions	d^2	D	f_{cos}
Gaussian I	**0.6844**	0.7577	0.7572
Gaussian II	0.8147	0.6317	**0.6306**
Gaussian III	0.7647	0.5764	**0.5731**

Table 3.4 Clustering results of real-world data (measure: VI_n)

Data	d^2	D	f_{cos}	$f_{l_p}(p=3)$	$f_{l_p}(p=6)$	$f_{l_p}(p=9)$
Breast-w	**0.5546**	0.9754	0.9729	0.9570	0.9411	0.9248
Ecoli	0.8017	0.8531	0.8430	0.8240	0.7912	**0.7521**
Glass	0.8726	0.8875	0.8705	0.8429	0.8008	**0.7752**
Housing	0.8852	0.9062	0.8958	0.8061	0.7078	**0.6818**
Iris	0.4495	0.3595	**0.3485**	0.3495	0.3591	0.3955
Pageblocks	0.9422	0.9940	0.9553	0.9151	**0.8861**	0.9056
Pendigits	1.0000	1.0000	0.9735	0.8993	**0.8828**	0.9016
Wine	0.8126	**0.8021**	0.8072	0.8203	0.8667	0.8843
Dermatology	0.9996	**0.8910**	0.8970	0.9524	1.0000	0.9919
Libras	1.0000	1.0000	0.9995	0.9824	0.9029	**0.8763**
Satimage	**0.7611**	0.7973	0.8071	0.8219	0.8635	0.8941

different data distributions. Indeed, d^2 only works best for the normal distribution data, and works even significantly worst for the Gamma distribution data. The 2-dimensional data sets illustrated in Fig. 3.3 further justify this observation. As can be seen in Table 3.3, for the three Gaussian distributions, d^2 only works best for the one with models of globular shapes. For the other two with models of elliptical shapes or even being rotated, d^2 shows much worse clustering performance than D and f_{cos}. Note that for each data distribution, we generate 10 sample data sets, and the results listed in Table 3.3 are the average VI_n values over the 10 samples.

We also evaluated GD-FCM on real-world data sets. Results are shown in Table 3.4. One observation is that, although being Type-II point-to-centroid distances, f_{l_p} and f_{cos} have the best performances for 7 out of 11 data sets. This observation is interesting, since (1) these distances have been proved to have flaws in theory, and (2) to the best of our knowledge, our study is the first one to study the effectiveness of f_{l_p} for clustering. In contrast, d^2 only works best for two data sets, which implies that d^2 may not be a good choice for most real-world applications.

In summary, GD-FCM has merits in providing diversified distance functions for clustering. Specifically, as a Type-II P2C-D, f_{l_p} shows some appealing advantages of clustering real-world data sets.

3.7 Related Work

Here, we present the work related to this research. First, people have recognized that outliers in data can affect the performance of FCM [9, 36]. To address this challenge, some researchers suggested to replace the squared Euclidean distance (l_2) with the city block distance (l_1) in FCM [24, 33]. Also, there are earlier work that uses l_1 and l_∞ norm in FCM [5, 22]. A more general study using l_p norm distances can be found in [17]. However, this work requires to change the way to compute the centroids (i.e., the weighted average of objects), and thus beyond the scope of our study. In a recent work, Banerjee et al. [1] proposed a general framework for K-means clustering by using the Bregman divergence [7] as the proximity function. In contrast, we focus on fuzzy c-means and its global convergence. Also, the point-to-centroid distance we propose is a more general concept than the Bregman divergence.

Second, the convergence is an important issue of FCM. Indeed, FCM has been known to produce results very quickly compared to some other approaches such as the maximum likelihood method [4]. In [2], based on the Zangwill's theorem [52], Bezdek first established the global convergence theorem for FCM, while this theorem was shown to be incorrect by Tucker [44]. A corrected version was provided in [16]. The recent work along this line can be found in [13, 20], which proved the convergence of the complete sequence. In addition, the first local convergence property for FCM can be found in [14]. This property can guarantee the linear convergence of FCM to the local minima. The computation of the rate of local convergence can be found in [16], and the testing of points for optimality is given by [19, 21, 25, 40]. A good summary for the convergence results can be found in [15]. The above work, however, mostly used the squared Euclidean distance or the inner product as the distance function. In contrast, our work provides the global convergence theorem for FCM using the general point-to-centroid distance as the distance function.

Finally, many FCM variants have been proposed in the literature. For instance, The Penalized FCM (PFCM) based on the fuzzy classification maximum likelihood was proposed in [48]. The Alternative FCM (AFCM) using a new metric was given by [47]. Also, Pedrycz proposed the Conditional FCM (CFCM) by considering the conditional effect imposed by some auxiliary variables [37], which was further generalized by [27]. Some Maximum Entropy-based Clustering (MEC) algorithms were proposed in [28, 35, 39]. Menard et al. [32] then proposed a Fuzzy Generalized c-Means (FGcM) algorithm that established a close relation between MEC and FCM. More recently, [50, 51] proposed the Generalized FCM (GFCM) and the Generalized Fuzzy Clustering Regularization (GFCR) to represent a wide variation of FCM algorithms. In addition, Teboulle established a unified continuous optimization framework for center-based clustering methods including FCM [43]. A comprehensive summarization of the FCM variants can be found in the excellent book by Miyamoto et al. [34]. Unlike these work, the focus of this chapter is on the distance functions that fit the classic FCM directly.

3.8 Concluding Remarks

In this chapter, we studied the generalization issues of distance functions for fuzzy c-means (FCM). Specifically, we show that any distance function that fits FCM directly can be derived by a continuously differentiable convex function. Such distance functions have the unified form which is defined as the point-to-centroid distance. Also, we prove that any point-to-centroid distance can fit FCM directly if the membership grade matrix will not degenerate. Finally, experimental results validate the effectiveness of the point-to-centroid distance for FCM. As a special case of FCM, the above results also apply for the widely used K-means clustering.

Appendix

As previously mentioned in Sect. 3.4, the degeneration of $\boldsymbol{U}$ may happen for GD-FCM using f_{II}^{ϕ}, although the probability of occurrence is extremely low for real-world data. Here, we provide a solution for this degeneration case and show how the global convergence of GD-FCM can still be guaranteed.

During a GD-FCM iteration, assume that we get $(\boldsymbol{U}^{de}, \boldsymbol{v}^{de}) = T_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$, where $\boldsymbol{U}^{de}$ is degenerate but $\bar{\boldsymbol{U}}$ is nondegenerate. Without loss of generality, suppose there is only one r such that $u_{rk}^{de} = 0 \; \forall \, k$. Then, we let $\hat{\boldsymbol{v}} = (\hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2, \ldots, \hat{\boldsymbol{v}}_c)^T$ be

$$\begin{cases} \hat{\boldsymbol{v}}_i = \boldsymbol{v}_i^{de}, \; \forall \, i \neq r, \text{ and} \\ \hat{\boldsymbol{v}}_r = \boldsymbol{x} \in \text{Conv}(\mathscr{X}), \; \boldsymbol{x} \neq \boldsymbol{v}_r^{de}. \end{cases} \tag{3.44}$$

Next, we try to resume the iteration by having $\hat{\boldsymbol{U}} = F(\hat{\boldsymbol{V}})$. If $\hat{\boldsymbol{U}}$ is nondegenerate, then we define $(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}) \doteq T_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$, and resume the iteration based on $(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}})$. Otherwise, we repeat choosing a new $\boldsymbol{x} \in \text{Conv}(\mathscr{X})$ for $\hat{\boldsymbol{v}}_r$ until we have a nondegenerate $\hat{\boldsymbol{U}} = F(\hat{\boldsymbol{V}})$. Typically, we choose $\boldsymbol{x}$ from $\mathscr{X}$ to ensure that $\exists \, k$ such that $u_{rk}^{de} \neq 0$.

Now, we establish the descent theorem when there is degeneration. Assume that $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$ is not in Ω' of Eq. (3.37). Since $u_{rk}^{de} = 0 \; \forall \, k$, we have $J_{mf}(\boldsymbol{U}^{de}, \boldsymbol{V}^{de}) = J_{mf}(\boldsymbol{U}^{de}, \hat{\boldsymbol{V}}) \geq J_{mf}(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}})$. Furthermore, by Theorem 3.6, we have $J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}) > J_{mf}(\boldsymbol{U}^{de}, \boldsymbol{V}^{de})$, which implies that $J_{mf}(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}}) > J_{mf}(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}})$. The descent theorem therefore holds. In addition, since we can "skip" the degenerate solution by jumping from $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$ to $(\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}})$, we still have $T_{mf} : M_{fc} \times \mathscr{S}^c \mapsto M_{fc} \times \mathscr{S}^c$, and the closeness of T_{mf} and the compactness of $M_{fc} \times \mathrm{Conv}(\mathscr{X})^c$ still hold. By assembling the above results, we again get the global convergence by Zangwill's convergence theorem.

Finally, in case that we cannot find any $\boldsymbol{x} \in \mathrm{Conv}(\mathscr{X})$ for $\hat{\boldsymbol{v}}_r$ such that $\hat{\boldsymbol{U}} = F(\hat{\boldsymbol{V}})$ is nondegenerate, we simply return $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{V}})$ as the solution.

References

1. Banerjee, A., Merugu, S., Dhillon, I., Ghosh, J.: Clusteringwith bregman divergences. J. Mach. Learn. Res. **6**, 1705–1749 (2005)
2. Bezdek, J.: A convergence theorem for the fuzzy isodata clustering algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-2**(1), 1–8 (1980)
3. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algoritms. Plenum Press, New York (1981)
4. Bezdek, J., Hathaway, R., Huggins, V.: Parametric estimation for normal mixtures. Pattern Recognit. Lett. **3**, 79–84 (1985)
5. Bobrowski, L., Bezdek, J.: C-means clustering with the l_1 and l_∞ norms. IEEE Trans. Syst. Man Cybern. **21**(3), 545–554 (1991)
6. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
7. Bregman, L.: The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. USSR Comput. Math. Math. Phys. **7**, 200–217 (1967)
8. Chen, L., Ng, R.: On the marriage of lp-norms and edit distance. In: Proceedings of the 13th International Conference on Very Large Data Bases, pp. 792–803. Toronto, Canada (2004)
9. Dav*é*, R.: Characterization and detection of noise in clustering. Pattern Recognit. Lett. **12**(11), 657–664 (1991)
10. Dunn, J.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybern. **3**, 32–57 (1973)
11. Flanders, H.: Differential Forms with Applications to the Physical Sciences. Dover Publications, New York (1989)
12. Golub, G., van Loan, C.: Matrix Computations. Johns Hopkins, Baltimore (1996)
13. Groll, L., Jakel, J.: A new convergence proof of fuzzy *c*-means. IEEE Trans. Fuzzy Syst. **13**(5), 717–720 (2005)
14. Hathaway, R., Bezdek, J.: Local convergence of the fuzzy *c*-means algorithms. Pattern Recognit. **19**, 477–480 (1986)
15. Hathaway, R., Bezdek, J.: Recent convergence results for the fuzzy c-means clustering algorithms. J. Classif. **5**, 237–247 (1988)
16. Hathaway, R., Bezdek, J., Tucker, W.: An improved convergence theory for the fuzzy c-means clustering algorithms. In: Bezdek, J. (ed.) Analysis of Fuzzy Information, vol. 3, pp. 123–131. CRC Press, Boca Raton (1987)
17. Hathaway, R.J., Bezdek, J.C., Hu, Y.: Generalized fuzzy c-means clustering strategies using l_p norm distances. IEEE Trans. Fuzzy Syst. **8**(5), 576–582 (2000)
18. Honda, K., Notsu, A., Ichihashi, H.: Fuzzy pca-guided robust k-means clustering. IEEE Trans. Fuzzy Syst. **18**(1), 67–79 (2010)
19. Hoppner, F., Klawonn, F.: A contribution to convergence theory of fuzzy c-means and derivatives. IEEE Trans. Fuzzy Syst. **11**(5), 682–694 (2003)
20. Hoppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. Wiley, New York (1999)

21. Ismail, M., Selim, S.: Fuzzy c-means: optimality of solutions and effective termination of the algorithm. Pattern Recognit. **19**, 481–485 (1984)
22. Jajuga, K.: l_1 norm-based fuzzy clustering. Fuzzy Sets Syst. **39**, 43–50 (1991)
23. Karoubi, M., Leruste, C.: Algebraic Topology via Differential Geometry. Cambridge University Press, Cambridge (1987)
24. Kersten, P.: Fuzzy order statistics and their application to fuzzy clustering. IEEE Trans. Fuzzy Syst. **7**, 708–712 (1999)
25. Kim, T., Bezdek, J., Hathaway, R.: Optimality tests for fixed points of the fcm algorithm. Pattern Recognit. **21**(6), 651–663 (1988)
26. Klawonn, F., Keller, A.: Fuzzy clustering based on modified distance measures. In: Proceedings of the 3rd International Symposium on Advances in Intelligent Data Analysis, pp. 291–302 (1999)
27. Leski, J.M.: Generalized weighted conditional fuzzy clustering. IEEE Trans. Fuzzy Syst. **11**(6), 709–715 (2003)
28. Li, R., Mukaidono, M.: A maximum entropy to fuzzy clustering. In: Proceedings of 4th IEEE Internation Conference on Fuzzy Systems, pp. 2227–2232. Yokohama, Japan (1995)
29. Luenberger, D., Ye, Y.: Linear and Nonlinear Programming, 3rd edn. Springer, New York (2008)
30. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
31. Meila, M.: Comparing clusterings-an axiomatic view. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 577–584 (2005)
32. Menard, M., Courboulay, V., Dardignac, P.: Possibistic and probabilistic fuzzy clustering: Unification within the framework of the nonextensive thermostatistics. Pattern Recognit. **36**(6), 1325–1342 (2003)
33. Miyamoto, S., Agusta, Y.: An efficient algorithm for l_1 fuzzy c-means and its termination. Control Cybern. **24**(4), 421–436 (1995)
34. Miyamoto, S., Ichihashi, H., Honda, K.: Algorithms for Fuzzy Clustering: Methods in c-Means Clustering with Applications. Springer, Berlin (2008)
35. Miyamoto, S., Umayahara, K.: Fuzzy clustering by quadratic regularization. In: Proceedings of the 7th IEEE Internation Conference on Fuzzy Systems, pp. 394–1399 (1998)
36. Ohashi, Y.: Fuzzy clustering and robust estimation. In: Proceedings of the 9th SAS Users Group International Meeting. Hollywood Beach, FL, USA (1984)
37. Pedrycz, W.: Conditional fuzzy c-means. Pattern Recognit. Lett. **17**, 625–632 (1996)
38. Pedrycz, W., Loia, V., Senatore, S.: Fuzzy clustering with viewpoints. IEEE Trans. Fuzzy Syst. **18**(2), 274–284 (2010)
39. Rose, K., Gurewitz, E., Fox, G.: A deterministic annealing approach to clustering. Pattern Recognit. Lett. **11**, 589–594 (1990)
40. Selim, S., Ismail, M.: On the local optimality of the fuzzy isodata clustering algorithm. IEEE Trans. Pattern Anal. Mach. Intell. **8**, 284–288 (1986)
41. Sledge, I., Bezdek, J., Havens, T., Keller, J.: Relational generalizations of cluster validity indices. IEEE Trans. Fuzzy Syst. **18**(4), 771–786 (2010)
42. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Boston (2005)
43. Teboulle, M.: A unified continuous optimization framework for center-based clustering methods. J. Mach. Learn. Res. **8**, 65–102 (2007)
44. Tucker, W.: Counterexamples to the convergence theorem for fuzzy isodata clustering algorithm. In: Bezdek, J. (ed.) Analysis of Fuzzy Information, vol. 3, pp. 109–122. CRC Press, Boca Raton (1987)
45. Wu, J., Xiong, H., Chen, J.: Adapting the right measures for k-means clustering. In: Proceedings of The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 877–886 (2009)

46. Wu, J., Xiong, H., Liu, C., Chen, J.: A generalization of distance functions for fuzzy *c*-means clustering with centroids of arithmetic means. IEEE Trans. Fuzzy Syst. (Forthcoming, 2012)
47. Wu, K., Yang, M.: Alternative *c*-means clustering algorithms. Pattern Recognit. **35**, 2267–2278 (2002)
48. Yang, M.: On a class of fuzzy classification maximum likelihood procedures. Fuzzy Sets Syst. **57**, 365–375 (1993)
49. Yi, B.K., Faloutsos, C.: Fast time sequence indexing for arbitrary lp norms. In: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 792–803. Cairo, Egypt (2000)
50. Yu, J., Yang, M.S.: Optimality test for generalized fcm and its application to parameter selection. IEEE Trans. Fuzzy Syst. **13**(1), 164–176 (2005)
51. Yu, J., Yang, M.S.: A generalized fuzzy clustering regularization model with optimality tests and model complexity analysis. IEEE Trans. Fuzzy Syst. **15**(5), 904–915 (2007)
52. Zangwill, W.: Nonlinear Programming: A Unified Approach. Prentice-Hall, Englewood Cliffs (1969)
53. Zhao, Y., Karypis, G.: Criterion functions for document clustering: experiments and analysis. Mach. Learn. **55**(3), 311–331 (2004)

Chapter 4
Information-Theoretic K-means for Text Clustering

4.1 Introduction

Cluster analysis is a fundamental task in various domains such as data mining [18], information retrieval [7], image processing, etc. Recent years have witnessed an increasing interest in information-theoretic clustering [4–6, 16, 19, 22], since information theory [3] can be naturally adapted as the guidance for the clustering process. For instance, the clustering analysis can be treated as the iteration process of finding a best partition on data in a way such that the loss of mutual information due to the partitioning is the least [4].

This chapter is focused on K-means clustering with KL-divergence [11] as the proximity function, which is called Info-Kmeans. To better understand the theoretic foundation of Info-Kmeans, we present an organized study of two different views on the objective functions of Info-Kmeans. First, we derive the objective function of Info-Kmeans from a probabilistic view. In this regard, we know that the probabilistic view takes several assumptions on data distributions, and the goal of Info-Kmeans is to maximize the likelihood function on multinomial distributions. In contrast, the information-theoretic view has no prior assumption on data distributions. In this case, the objective function of Info-Kmeans is to find a best partition on data so that the loss of mutual information is minimized. The above indicates that the information-theoretic view on Info-Kmeans is more appealing, since we do not need to make any assumption on data distributions. As a result, in this chapter, we take the information-theoretic view on Info-Kmeans.

While Info-Kmeans has the sound theoretic foundation, there are some challenging issues with it from a practical viewpoint. For example, people have shown that, for text clustering, the performance of Info-Kmeans is poorer than that of the spherical K-means, which has the cosine similarity as the proximity function [22] and is the benchmark of text clustering. Indeed, for high-dimensional sparse text vectors, Info-Kmeans often has some difficult scenarios. For example, the centroids in sparse data usually contain many zero-value features. This creates infinite KL-divergence values, which lead to a challenge in assigning objects to the centroids during the iteration process of Info-Kmeans. A traditional way to handle this zero-value dilemma is to smooth the sparse data by adding a very small value to every instance in the data [16]. In this way, there is no instance having zero feature values. While this smoothing method can avoid the zero-value dilemma for Info-Kmeans, it can also degrade the clustering performance, since the true values and the sparseness of data have been changed.

As an alternative to the smoothing technique, in this chapter, we propose a Summation-based Incremental Learning (SAIL) algorithm for Info-Kmeans clustering. Specifically, by using an equivalent objective function, SAIL replaces the computation of KL-divergence for the instance-centroid distances, by the incremental computation of Shannon entropy [3] for the centroids alone. This can avoid the zero-value dilemma caused by the use of KL-divergence. Moreover, by transforming the computation of KL-divergence, we can make use of the sparseness of text vectors and further lower the computational costs of SAIL. Two variants, i.e. V-SAIL using the Variable Neighborhood Search (VNS) meta-heuristic and PV-SAIL using the multithreaded parallel computing, have also been proposed to improve the clustering quality of SAIL while keeping the high computational efficiency.

Our experimental results on various real-world text corpora have shown that, with SAIL as a booster, the clustering performance of Info-Kmeans can be significantly improved. Indeed, for most of the text collections, SAIL produces clustering results competitive to or even slightly better than the results of the state-of-the-art spherical K-means algorithm: CLUTO. Some settings such as feature weighting, instance weighting and bisecting, have been shown to have varied effects on SAIL, but SAIL without these settings shows more robust results. V-SAIL further improves the clustering quality of SAIL by searching around the neighborhood of the solution using the VNS scheme. As a natural extension of V-SAIL, PV-SAIL effectively lowers the high computational cost of V-SAIL by using the multithreaded parallel computation.

The remainder of this chapter is organized as follows. Section 4.2 highlights the information-theoretic view of Info-Kmeans. Section 4.3 introduces the zero-value dilemma in Info-Kmeans clustering. Sections 4.4 and 4.5 introduce in details the SAIL algorithm and the two variants: V-SAIL and PV-SAIL. Section 4.6 shows the experimental results. Finally, we present related work in Sect. 4.7 and conclude this chapter in Sect. 4.8.

4.2 Theoretical Overviews of Info-Kmeans

To better understand the theoretical foundation of Info-Kmeans, in this section, we provide an organized study of two different views, i.e. the probabilistic view and the information-theoretic view, on the objective function of Info-Kmeans.

4.2.1 The Objective of Info-Kmeans

K-means [12] is a prototype-based, simple partitional clustering technique which attempts to find the user-specified K clusters. These clusters are represented by their centroids (a cluster centroid is typically the arithmetic mean of the data objects in that cluster). As pointed out in Chap. 1, different distance functions can lead to different types of K-means. Our focus in this chapter is on Info-Kmeans. Let $D(x\|y)$ denote the KL-divergence [11] between two discrete distributions x and y, we have

$$D(x\|y) = \sum_i x_i \log \frac{x_i}{y_i}. \tag{4.1}$$

It is easy to observe that, in most cases $D(x\|y) \neq D(y\|x)$, and that $D(x\|y) + D(y\|z) \geq D(x\|z)$ cannot be guaranteed. So D is not a *metric*. If we let "dist" be D in Eq. (1.1), we have the objective function of Info-Kmeans as follows:

$$obj: \quad \min \sum_k \sum_{x \in c_k} \pi_x D(x\|m_k), \tag{4.2}$$

where each instance x has been normalized to a discrete distribution before clustering. To further understand Info-Kmeans, we take two different views on Eq. (4.2) as follows.

4.2.2 A Probabilistic View of Info-Kmeans

In this section, we first derive the objective function of Info-Kmeans from a probabilistic view. Specifically, the objective function can be derived by maximizing the "partitioned" likelihood function of the EM algorithm, i.e. the crisp version of EM [22].

Assume that we have a text collection $\mathbb{D}$, which consists of K crisp partitions in multinomial distributions with different parameters, i.e. $\theta_1, \ldots, \theta_K$, respectively. Let random variables X and Y denote the text and the term, respectively. Let $n(x, y)$ denote the number of occurrences of term y in document x, and $n(x) = \sum_y n(x, y)$. Then we have

Theorem 4.1 *Let* $L = P(\mathbb{D}|\Theta) = \Pi_x p(x|\Theta)$ *be the likelihood function. Let* $B = \sum_x n(x)$, *and* $A = -\sum_x n(x) H(p(Y|x))$, *where* H *is the Shannon entropy. Then, we have*

$$\frac{A - \log L}{B} = \sum_k \sum_{x \in c_k} p(x) D(p(Y|x) \| p(Y|\theta_k)), \tag{4.3}$$

where $p(x) = n(x)/\sum_x n(x)$ *and* $p(y|x) = n(x,y)/n(x)$.

Proof By definition,

$$\begin{aligned}
\log L &= \sum_x \log p(x|\Theta) \\
&\overset{a}{=} \sum_k \sum_{x\in c_k} \log p(x|\theta_k) \\
&\overset{b}{=} \sum_k \sum_{x\in c_k} \sum_y n(x,y)\log(p(y|\theta_k)) \\
&= \sum_k \sum_{x\in c_k} n(x) \sum_y p(y|x)\log p(y|\theta_k),
\end{aligned}$$

where "a" reflects the "crisp" property of the modified EM model, and "b" follows the multinomial distribution, i.e. $p(x|\theta) = \Pi_y p(y|\theta)^{n(x,y)}$.

Meanwhile, A can be transformed into

$$A = \sum_k \sum_{x\in c_k} n(x) \sum_y p(y|x)\log p(y|x).$$

If we substitute the transformed A and $\log L$ into the left-hand-side of Eq. (4.3), we can easily get the right-hand-side. So we complete the proof. □

Remark Let us compare Eq. (4.3) with Eq. (4.2). If we let $\pi_x \equiv p(x)$, $x \equiv p(Y|x)$, and $m_k \equiv p(Y|\theta_k)$, we have $obj \Leftrightarrow \min(A - \log L)/B \Leftrightarrow \max \log L$. This implies that, if we take the probabilistic view of the objective function, Info-Kmeans aims to maximize the likelihood function based on multinomial distributions. This probabilistic view of Info-Kmeans requires two assumptions: $p(x) = n(x)/\sum_x n(x)$, and the multinomial distribution of $p(x|\theta_k)$. However, in the experimental section, we will show these assumptions may degrade the performance of Info-Kmeans.

4.2.3 An Information-Theoretic View of Info-Kmeans

Here, we derive the objective function in Eq. (4.2) from an information-theoretic point of view. We begin our analysis by introducing an important lemma as follows.

Given a set of discrete probabilistic distributions $\{p_1, p_2, \ldots, p_n\}$ and the corresponding weights $\{\pi_1, \pi_2, \ldots, \pi_n\}$, we have

Lemma 4.1 [3]

$$\sum_{i=1}^{n} \pi_i D\left(p_i || \sum_{i=1}^{n} \pi_i p_i\right) = H\left(\sum_{i=1}^{n} \pi_i p_i\right) - \sum_{i=1}^{n} \pi_i H(p_i). \tag{4.4}$$

Now, given a text collection $\mathbb{D}$, we want to partition $\mathbb{D}$ into K clusters without overlapping. Let random variables X, Y and C denote the text, the term and the cluster, respectively. Let x, y and c be the corresponding instances with $p(x)$, $p(y)$ and $p(c)$ being the probabilities of occurrences. Furthermore, we assume that $p(c) = \sum_{x \in c} p(x)$. Then we have the following theorem:

Theorem 4.2 *Let $I(X, Y)$ be the mutual information between two random variables X and Y, then*

$$I(X, Y) - I(C, Y) = \sum_k \sum_{x \in c_k} p(x) D(p(Y|x) \| p(Y|c_k)). \tag{4.5}$$

Proof By definition,

$$\begin{aligned}
&I(X, Y) - I(C, Y) \\
&= \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)} - \sum_k \sum_y p(c_k, y) \log \frac{p(c_k, y)}{p(c_k)p(y)} \\
&= \underbrace{\sum_x \sum_y p(y|x)p(x) \log p(y|x)}_{(a)} - \underbrace{\sum_k \sum_y p(y|c_k)p(c_k) \log p(y|c_k)}_{(b)} \\
&\quad - \underbrace{\sum_x \sum_y p(y|x)p(x) \log p(y)}_{(c)} + \underbrace{\sum_k \sum_y p(y|c_k)p(c_k) \log p(y)}_{(d)}.
\end{aligned}$$

If we substitute $p(y|c_k) = \sum_{x \in c_k} \frac{p(x)}{p(c_k)} p(y|x)$ into (d), we have $(c) = (d)$. Furthermore, it is easy to show

$$\begin{aligned}
(a) &= -\sum_k p(c_k) \left(\sum_{x \in c_k} \frac{p(x)}{p(c_k)} H(p(Y|x)) \right), \\
(b) &= -\sum_k p(c_k) H \left(\sum_{x \in c_k} \frac{p(x)}{p(c_k)} p(Y|x) \right).
\end{aligned}$$

By Lemma 4.1, we finally have Eq. (4.5). □

Remark Theorem 4.2 formulates the information-theoretic view of Info-Kmeans; that is, Info-Kmeans tries to find a best partition on data so that the loss of mutual information due to the partitioning is minimized.

4.2.4 Discussions

In summary, we have two different views on Info-Kmeans, as described by Eqs. (4.3) and (4.5), respectively. Both views lay the theoretic foundations of Info-Kmeans. Nonetheless, the information-theoretic view seems to be more appealing, since there is no prior assumption for $p(x)$ and $p(x|c)$, which is however crucial for the probabilistic view. In fact, we can regard the probabilistic framework as a special case of the information-theoretic framework of Info-Kmeans. And we will show in experimental results that in most cases the assumption of $p(x)$ and $p(x|c)$ in the probabilistic view is harmful to the clustering accuracy.

4.3 The Dilemma of Info-Kmeans

Though having sound theoretical foundations, Info-Kmeans has long been criticized for having performances inferior to the spherical K-means [21] on text clustering [22]. However, in this section, we highlight an implementation challenge of Info-Kmeans. We believe this challenge is one of the key factors that degrade the clustering performance of Info-Kmeans.

Assume that we use Info-Kmeans to cluster a text corpus. To optimize the objective in Eq. (4.2), we launch the two-phase iteration process of Info-Kmeans. To this end, we must compute the KL-divergence between each text vector $p(Y|x)$ and each centroid $p(Y|c_k)$. In practice, we usually let

$$p(Y|x) = \frac{x}{n(x)}, \text{ and } p(Y|c_k) = \frac{\sum_{x \in c_k} p(x)p(Y|x)}{\sum_{x \in c_k} p(x)},$$

where $n(x)$ is the sum of all the term frequencies of x, $p(x)$ is the weight of x, as in Eqs. (4.3) and (4.5). Therefore, by Eq. (4.1), to compute $D(p(Y|x)\|p(Y|c_k))$, we should expect that all the feature values of x are positive real numbers. Unfortunately, however, this is not the case for high-dimensional text vectors, which are famous for the sparseness in their high dimensionality.

To illustrate this, we observe the computation of KL-divergence in each dimension y. As we know, $D(p(Y|x)\|p(Y|c_k)) = \sum_y p(y|x) \log \frac{p(y|x)}{p(y|c_k)}$. To simplify the notations, hereinafter we denote $p(y|x)\log(p(y|x)/p(y|c_k))$ by D_y. Then, the different combinations of $p(y|x)$ and $p(y|c_k)$ values can result in four scenarios as follows:

1. *Case 1*: $p(y|x) > 0$ *and* $p(y|c_k) > 0$. In this case, the computation of D_y is straightforward, and the result can be any real number.
2. *Case 2*: $p(y|x) = 0$ *and* $p(y|c_k) = 0$. In this case, we can simply omit this feature, or equivalently let $D_y = 0$.
3. *Case 3*: $p(y|x) = 0$ *and* $p(y|c_k) > 0$. In this case, $\log(p(y|x)/p(y|c_k)) = \log 0 = -\infty$, which implies that the direct computation is infeasible. However,

Table 4.1 Four cases in KL-divergence computations

Case	i	ii	iii	iv
$p(y\|x)$	>0	$=0$	$=0$	>0
$p(y\|c_k)$	>0	$=0$	>0	$=0$
D_y	$\in \mathbb{R}$	$=0$	$=0$	$+\infty$

by the L' Hospital's rule [2], $\lim_{x \to 0^+} x \log(x/a) = 0$ ($a > 0$). So we can let $x \doteq p(y|x)$ and $a \doteq p(y|c_k)$, and thus have $D_y = 0$.

4. *Case 4*: $p(y|x) > 0$ *and* $p(y|c_k) = 0$. In this case, $D_y = +\infty$, which is hard to handle in practice.

We summarize the above four cases in Table 4.1. As can be seen, for Cases 1 and 2, the computation of D_y is logically reasonable. However, the computation of D_y in Case 3 is actually questionable; that is, it cannot reveal any difference between $p(Y|x)$ and $p(Y|c_k)$ in dimension y, although $p(y|c_k)$ may deviate heavily from zero. Also, it implies that the differences of various centroids in dimension y will be omitted.

Nevertheless, the most difficult case to handle is Case 4. On one hand, it is hard to do computations with $+\infty$ in practice. On the other hand, it is obvious that if there is some dimension y of Case 4, the total KL-divergence of $p(Y|x)$ and $p(Y|c_k)$ is infinite. This does not work for high-dimensional sparse text vectors, because the centroids of such data typically contain many zero-value features. Therefore, we will have big challenges in assigning instances to the centroids. We call this problem the "zero-value dilemma".

One way to solve the above dilemma is to smooth the sparse data. For instance, we can add a very small positive value to the entire data set so as to avoid having any zero feature value. While this smoothing technique facilitates the computations of KL-divergence, it indeed changes the sparseness property of the data. We will demonstrate in the experimental section that this method actually degrades the clustering performance of Info-Kmeans.

In summary, there is a need to develop a new implementation scheme for Info-Kmeans which should be able to avoid the zero-value dilemma.

4.4 The SAIL Algorithm

In this section, we propose a new variant, named SAIL, for Info-Kmeans. We first simplify the objective function of Info-Kmeans using the point-to-centroid distance introduced in Chap. 3. Then we refine the computations in SAIL to further improve the efficiency. Finally, we present the algorithmic details.

4.4.1 SAIL: Theoretical Foundation

We begin by briefly reviewing the notion of *Point-to-Centroid Distance* proposed in Chap. 3. As formulated in Definition 3.2, a point-to-centroid distance is derived by a continuously-differentiable convex function ϕ. As different ϕ can lead to different instances, the point-to-centroid distance is actually a family of multiple distance functions. More importantly, as indicated by Corollary 3.4, the point-to-centroid distance is the only choice of the distance function for K-means clustering, with centroids being the arithmetic means of cluster members.

Table 3.2 lists some popular point-to-centroid distances widely used for K-means clustering. In the table, the squared Euclidean distance (d^2) is most widely used for a variety of clustering applications [18]. The cosine distance (f_{cos}), derived from a convex but not strictly convex ϕ, is equivalent to the cosine similarity used for the so-called spherical K-means [21], which is usually considered as the state-of-the-art method for text clustering. Our focus in this chapter, i.e. the KL-divergence (D) for Info-Kmeans, also belongs to this family. Specifically, according to Definition 3.2, KL-divergence can be rewritten as

$$D(x\|y) = -H(x) + H(y) + (x - y)^t \nabla H(y), \tag{4.6}$$

where $H(x) = \sum_i x_i \log x_i$ is the Shannon entropy [3] of a discrete distribution x.

Based on $D(x\|y)$ in Eq. (4.6), we now lay the theoretical foundation of the **S**ummation-b**A**sed **I**ncremental **L**earning (SAIL) algorithm, a new variant of Info-Kmeans. Specifically, we have the following theorem:

Theorem 4.3 *Let* $p(c_k) = \sum_{x \in c_k} p(x)$*. The objective function of Info-Kmeans:*

$$O_1: \ \min \sum_k \sum_{x \in c_k} p(x) D(p(Y|x) \| p(Y|c_k))$$

is equivalent to

$$O_2: \ \min \sum_k p(c_k) H(p(Y|c_k)). \tag{4.7}$$

Proof By Eq. (4.6), we have

$$\begin{aligned} D(p(Y|x)\|p(Y|c_k)) = &- H(p(Y|x)) + H(p(Y|c_k)) + (p(Y|x) \\ &- p(Y|c_k))^t \nabla H(p(Y|c_k)). \end{aligned}$$

As a result,

$$\begin{aligned}
&\sum_k \sum_{x\in c_k} p(x) D(p(Y|x)\| p(Y|c_k)) \\
&= \underbrace{\sum_k p(c_k) H(p(Y|c_k))}_{(a)} - \underbrace{\sum_x p(x) H(p(Y|x))}_{(b)} \\
&\quad - \underbrace{\sum_k \sum_{x\in c_k} p(x)(p(Y|x) - p(Y|c_k))^t \nabla H(p(Y|c_k))}_{(c)} .
\end{aligned}$$

Since $p(Y|c_k) = \sum_{x\in c_k} p(x)p(Y|x)/p(c_k)$, we have

$$\sum_{x\in c_k} p(x)(p(Y|x) - p(Y|c_k)) = 0.$$

Accordingly,

$$(c) = \sum_k \nabla H(p(Y|c_k))^t \sum_{x\in c_k} p(x)(p(Y|x) - p(Y|c_k)) = 0.$$

Moreover, (b) is a constant given the data set and the weights for the instances. Thus, the goal of O_1 is equivalent to minimize (a), which completes the proof. □

The equivalent O_2 given in Eq. (4.7) is right the objective function of SAIL. That is, by replacing the computations of KL-divergence between instances and centroids by the computations of Shannon entropy of centroids only, SAIL can avoid the zero-value dilemma in information-theoretic clustering of highly sparse texts.

4.4.2 SAIL: Computational Issues

Now, based on the objective function in Eq. (4.7), we establish the computational scheme for SAIL. The major concern here is the efficiency issue.

Generally speaking, SAIL is a greedy scheme which updates the objective-function value "instance by instance". That is, SAIL first selects an instance from the text collection and assigns it to the most suitable cluster. Then the objective-function value and other related variables are updated immediately after the assignment. The process will be repeated until some stopping criteria are met.

Apparently, to find the suitable cluster is the critical point of SAIL. To illustrate this, suppose SAIL randomly selects $p(Y|x')$ from a cluster $c_{k'}$. Then, if we assign $p(Y|x')$ to cluster c_k, the change of the objective-function value will be

$$\begin{aligned}\Delta_k &= O_2(\text{new}) - O_2(\text{old}) \qquad (4.8)\\ &= \underbrace{(p(c_{k'}) - p(x'))H\left(\frac{\sum_{x\in c_{k'}} p(x)p(Y|x) - p(x')p(Y|x')}{p(c_{k'}) - p(x')}\right)}_{(a)}\\ &\quad - \underbrace{p(c_{k'})H\left(\frac{\sum_{x\in c_{k'}} p(x)p(Y|x)}{p(c_{k'})}\right)}_{(b)}\\ &\quad + \underbrace{(p(c_k) + p(x'))H\left(\frac{\sum_{x\in c_k} p(x)p(Y|x) + p(x')p(Y|x')}{p(c_k) + p(x')}\right)}_{(c)}\\ &\quad - \underbrace{p(c_k)H\left(\frac{\sum_{x\in c_k} p(x)p(Y|x)}{p(c_k)}\right)}_{(d)},\end{aligned}$$

where (a)–(b) and (c)–(d) represent the two parts of changes on the objective-function value due to the movement of $p(Y|x')$ from cluster $c_{k'}$ to cluster c_k. Then x' will be assigned to the cluster c with the smallest Δ, i.e. $c = \arg\min_k \Delta_k$.

The computation of Δ_k in Eq. (4.8) has two appealing properties. First, it only relates to the changes occurred within the two involved clusters c_k and $c_{k'}$. Other clusters remain unchanged and thus have no contribution to Δ_k. Second, the computations of both $\sum_{x\in c} p(x)$ and $\sum_{x\in c} p(x)p(Y|x)$ have additivity, which can facilitate the computation of Δ_k. Indeed, these two summations, incrementally updated during the clustering, are the key elements of SAIL.

The computation of Δ_k in Eq. (4.8), however, still suffers from the high costs of computing Shannon entropy in (a) or (c). Let us take the entropy computation in (a) for example. Since the denominator changes from $p(c_{k'})$ to $p(c_{k'}) - p(x')$, every dimension in the numerator $\sum_{x\in c_{k'}} p(x)p(Y|x) - p(x')p(Y|x')$ will have a new value, which requires to recompute the logarithm for each dimension. These computations are indeed a huge cost for text vectors in high dimensionality. Therefore, here comes the question: can we make use of the high sparseness of text vectors to further improve the computational efficiency of SAIL?

The answer is positive. To illustrate this, recall SAIL's objective function O_2 in Eq. (4.7). Let $S(k, y)$ denote $\sum_{x\in c_k} p(x)p(y|x)$. As $p(Y|c_k) = \frac{\sum_{x\in c_k} p(x)p(Y|x)}{p(c_k)}$, we have

$$
\begin{aligned}
&\sum_k p(c_k) H(p(Y|c_k)) \\
&= -\sum_k \sum_y \sum_{x \in c_k} p(x)p(y|x)\left(\log \sum_{x \in c_k} p(x)p(y|x) - \log p(c_k)\right) \\
&= \sum_k \left(\sum_{x \in c_k} p(x) \sum_y p(y|x)\right) \log p(c_k) - \sum_k \sum_y S(k,y) \log S(k,y) \\
&= \sum_k p(c_k) \log p(c_k) - \sum_k \sum_y S(k,y) \log S(k,y).
\end{aligned}
\tag{4.9}
$$

Accordingly, if we move x' from cluster $c_{k'}$ to cluster c_k, the change of the objective-function value will be

$$
\begin{aligned}
\Delta_k = {} & (p(c_k) + p(x')) \log(p(c_k) + p(x')) - p(c_k) \log p(c_k) \\
& + (p(c_{k'}) - p(x')) \log(p(c_{k'}) - p(x')) - p(c_{k'}) \log p(c_{k'}) \\
& + \underbrace{\sum_{\{y|p(y|x') \neq 0\}} S(k,y) \log S(k,y) - S^+(k,y) \log S^+(k,y)}_{(a)} \\
& + \underbrace{\sum_{\{y|p(y|x') \neq 0\}} S(k',y) \log S(k',y) - S^-(k',y) \log S^-(k',y)}_{(b)}\,.
\end{aligned}
\tag{4.10}
$$

where $S^+(k, y) = S(k, y) + p(x')p(y|x')$, and $S^-(k', y) = S(k', y) - p(x')p(y|x')$.

According to Eq. (4.10), only the non-empty features of $p(Y|x')$ have contributions to Δ_k in (a) or (b), and thus will trigger the expensive computations of logarithm. Considering that a text vector $p(Y|x')$ is often very sparse, i.e. has many empty features, the computational saving due to Eq. (4.10) will be significant. As a result, we adopt Eq. (4.10) rather than Eq. (4.8) for the computation of Δ_k in SAIL, and give the comparative results in the experimental section.

Discussion. It is clear that SAIL differs from the traditional K-means. Indeed, SAIL is an incremental algorithm while the traditional K-means usually employs the batch-learning mode. Furthermore, SAIL also differs from the traditional incremental K-means; that is, to decide the assignment of each selected instance, SAIL does not compute the KL-divergence values between the instance and all the centroid vectors. Instead, it computes and updates the Shannon entropies of the centroids. This computation is supported by the two incrementally-maintained summations for each cluster c: $p(c) = \sum_{x \in c} p(x)$ and $p(Y|c) = \sum_{x \in c} p(x)p(Y|x)$. That is why we call this method the Summation-bAsed Incremental Learning (SAIL) algorithm.

Fig. 4.1 The pseudocodes of SAIL

```
[$objVal^*, label^*, \pi$] = SAIL($\mathbb{D}, K, reps, maxIter$)

Input:    $\mathbb{D}$: text collection
          $K$: the number of clusters
          $reps$: the number of repeated clusterings
          $maxIter$: the max number of iterations in a clustering
Output:   $objVal^*$: the optimal objective-function value
          $label^*$: the cluster labels of documents
          $\pi$: the vector of document weights
Variable: $cluSum$: storing $\sum_{x \in c_k} \pi_x$ and $\sum_{x \in c_k} \pi_x x$, $\forall k$

Procedure
 1.   Load $\mathbb{D}$, and compute $\pi = \{\pi_x | x \in \mathbb{D}\}$;
 2.   Preprocess $\mathbb{D}$;
 3.   $\forall x \in \mathbb{D}$, normalize $x$ to a discrete distribution;
 4.   for $i = 1 : reps$
 5.       Initialize $label[i]$, then $objVal[i]$ and $cluSum[i]$;
 6.       for $j = 1 : maxIters$
 7.           LocalSearch($\mathbb{D}, \pi, objVal[i], label[i], cluSum[i]$);
 8.           if $label[i]$ is unchanged
 9.               break;
10.           end if
11.       end for
12.   end for
13.   $t = \arg\min_i objVal[i]$;
14.   return $objVal^* = objVal[t]$, $label^* = label[t]$, $\pi$;
```

Fig. 4.2 The LocalSearch subroutine

```
LocalSearch($\mathbb{D}, \pi, objVal, label, cluSum$)

1.   for $l = 1 : n$
2.       Randomly select without replacement an $x$ from $\mathbb{D}$;
3.       for $s = 1 : K$
4.           $\Delta objVal(k) =$ TestAssign($para$);
5.           // $para \doteq \langle x, \pi_x, cluSum, label(x), k, objVal \rangle$
6.       end for
7.       $k^* = \arg\min_k(\{\Delta objVal(k), k = 1, \cdots, K\})$;
8.       Update $objVal$, $label$ and $cluSum$ accordingly;
9.   end for
```

4.4.3 SAIL: Algorithmic Details

In this section, we present the main process and the implementation details of SAIL. Figures 4.1 and 4.2 show the pseudocodes of the SAIL algorithm.

Lines 1–3 in Fig. 4.1 are for data initialization. In line 1, text collection $\mathbb{D}$ is loaded into the memory. There are two methods for assigning the weights of instances; that is, $\pi_x = n(x)/\sum_x n(x)$, or simply, $\pi_x = 1$. The second one is much simpler and is the default setting in our experiments. The preprocessing of $\mathbb{D}$ in line 2 includes the row and column modeling, e.g. $tf\text{-}idf$, to smooth the instances and assign weights to the features. Then, in line 3, we normalize the instance x to $x = p(Y|x)$ where $p(y|x) = n(x, y)/n(x)$ for each feature y.

Lines 4–12 in Fig. 4.1 show the clustering process. Line 5 is for initialization, where $label_{n\times 1}$ contains the cluster labels of all instances, and $cluSum_{K\times(d+1)}$ stores the summations of the weights and weighted instances in each cluster. That is, for $k = 1, \ldots, K$, $cluSum(k, 1:d) = \sum_{x\in c_k} \pi_x p(Y|x)$, and $cluSum(k, d+1) = \sum_{x\in c_k} \pi_x$, where n, d and K are the numbers of instances, features and clusters, respectively. Two initialization modes are employed in our implementation, i.e. "random label" and "random center" (the default one), and the required variables are then computed.

The LocalSearch subroutine in Line 7 performs clustering for each instance. As shown in Fig. 4.2, it traverses all instances at random as a round. In each round, it assigns each instance to the cluster with the heaviest drop of the objective-function value, and then updates the values of the related variables. The computational details have been given in Sect. 4.4.2. Lines 8–10 show the stopping criterion in addition to *maxIter*; that is, if no instance changes its label after a round, we stop the clustering. Finally, Lines 13–14 choose and return the best clustering result among the *reps* clusterings.

Next, we briefly discuss the convergence issues of SAIL. Since the objective-function value decreases continuously after reassigning each instance, and the combinations of the labels assigned to all instances are limited, SAIL guarantees to converge after limited iterations. However, due to the complexity of the feasible region, SAIL often converges to a local minima or a saddle point. That is why we usually do multiple clusterings in SAIL and choose the one with a lowest objective-function value.

SAIL also preserves the most important advantage of Info-Kmeans—low computational costs. Specially, the space and time requirements are $O((n + K)\bar{d})$ and $O(IKn\bar{d})$, respectively, where I is the number of iterations required for convergence, and $\bar{d}$ is the average number of non-empty features of each instance. Since K is often small and I is typically not beyond 20 (refer to the empirical results in the experimental section), the complexity of SAIL is roughly linear to the size of non-empty elements in a text collection. Also, by employing Eq. (4.10) rather than Eq. (4.8) for SAIL, we can take advantage of the sparseness of text collections to further improve the efficiency of SAIL.

4.5 Beyond SAIL: Enhancing SAIL via VNS and Parallel Computing

SAIL is essentially a combinatorial optimization algorithm. Therefore, like most iterative scheme, SAIL is apt to converge to some local minima or saddle points, especially for text vectors in high dimensionality. To meet this challenge, further

Fig. 4.3 Illustration of VNS

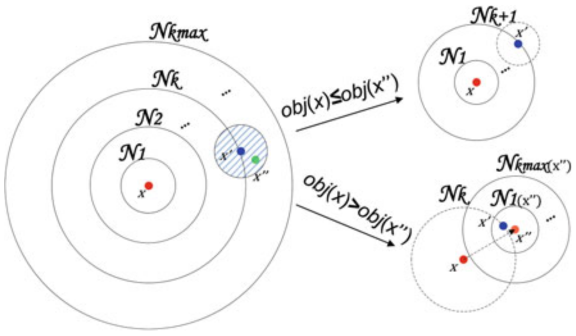


Fig. 4.4 The pseudocodes of V-SAIL

```
[$objVal^*, label^*, \pi$] = V-SAIL($para, k_{max}$)

Input:    $para \doteq \langle \mathbb{D}, K, reps, maxIter \rangle$, as for SAIL
          $k_{max}$: the number of neighborhood structures
Output: $objVal^*$: the optimal objective-function value
          $label^*$: the cluster labels of documents
          $\pi$: the vector of document weights
Variable: $\mathcal{N}_i$: the $i$th neighborhood with a Hamming
          distance $i \times \|\mathbb{D}\|/k_{max}$ from the current label

Procedure
 1.   $[objVal, label, \pi] = \text{SAIL}(\mathbb{D}, K, reps, maxIter)$;
 2.   $i = 1$;
 3.   while $(i{+}{+} \leq k_{max})$
 4.       $label' = \text{Shaking}(\mathcal{N}_i, label)$;
 5.       Update $objVal'$ and $CluSum'$ accordingly;
 6.       for $j = 1 : maxIter$
 7.           $\text{LocalSearch}(\mathbb{D}, \pi, objVal', label', cluSum')$;
 8.           if $label'$ is unchanged
 9.               break;
10.           end if
11.       end for
12.       if $objVal' < objVal$
13.           $objVal = objVal'$, $label = label'$, $i = 1$;
14.       end if
15.       if the stopping condition is met
16.           break;
17.       end if
18.   end while
19.   return $objVal^* = objVal$, $label^* = label$, $\pi$;
```

study is still needed to help SAIL jump out of the inferior points. We here propose to use the Variable Neighborhood Search (VNS) scheme [8, 14], and establish the VNS-enabled SAIL algorithm: V-SAIL.

VNS is a meta-heuristic for solving combinatorial and global optimization problems. The idea of VNS is to conduct a systematic change of neighborhood within the search [8]. Figure 4.3 schematically illustrates the search process of VNS. That is, VNS first finds an initial solution x, and then shakes in the kth neighborhood $\mathcal{N}_k$ to obtain x'. Then VNS centers the search around x' and finds the local minimum x''. If x'' is better than x, x is replaced by x'', and VNS starts to shake in the first neighborhood of x''. Otherwise, VNS continues to search in the $(k+1)$-th neighborhood of x. The set of neighborhoods are often defined by metric function in the solution space, e.g. Hamming distance, Eular distance, k-OPT operator, etc. The stopping condition for VNS can be the size of the neighborhood set, the maximum CPU time, and/or the maximum number of iterations. In what follows, we show how to enhance SAIL via the VNS scheme.

4.5.1 The V-SAIL Algorithm

Figure 4.4 shows the pseudocodes of the V-SAIL algorithm. Generally speaking, V-SAIL is a two-stage algorithm employing a **c**lustering step and a **r**efinement step. In **c**-step, the SAIL algorithm is called to generate a clustering result. This result serves as the starting point of the subsequent **r**-step, which employs the VNS scheme to refine the clustering result to a more accurate one. It is interesting to note that the "LocalSearch" subroutine of SAIL is also called in VNS as a heuristic method for searching local minima. Some important details are as follows.

Lines 3–18 in Fig. 4.4 describe the **r**-step of V-SAIL. In Line 4, for the ith neighborhood $\mathcal{N}_i$, the "Shaking" function is called to generate a solution $label'$ in $\mathcal{N}_i$ that has a Hamming distance $H_i = i \times |\mathbb{D}|/k_{max}$ to the current solution $label$. More specifically, "Shaking" first randomly selects H_i instances, and then changes their labels at random in $label$, which results in a new solution $label'$. Apparently, $label'$ tends to deviate $label$ more heavily as the increase of i. The idea is that once the best solution in a large region has been found, it is necessary to explore an improved one far from the incumbent solution.

In Lines 6–11, the "LocalSearch" subroutine of SAIL is called to search for a local minimum solution initialized on $label'$. This well demonstrates that SAIL is not only a clustering algorithm, but also a combinatorial optimization method. Then in Lines 12–14, if the minimum is smaller than the current $objVal$, we update the related variables, and set the solution as the new starting point of VNS. The stopping condition in Line 15 is very simple; that is, we stop VNS either if a maximum repeat-time of calling "Shaking" is met or when a given CPU time is due.

In summary, V-SAIL well combines the complemental advantages of SAIL and VNS. That is, VNS can help SAIL avoid inferior solutions, while SAIL can help VNS fast locate a good local minimum. We will show the empirical results in the experimental section.

Fig. 4.5 The pseudocodes of PV-SAIL

```
$[objVal^*, label^*, \pi]$ = PV-SAIL$(para, nSub, maxInv)$

Input:  $para \doteq \langle \mathbb{D}, K, reps, maxIter, k_{max} \rangle$ as in V-SAIL
        $nSub$: the number of subthreads
        $maxInv$: the max number of invocation of subthreads
Output: $objVal^*$: the optimal objective-function value
        $label^*$: the cluster labels of documents
        $\pi$: the vector of document weights

Procedure
Main thread:
 1.   $[objVal, label, \pi] = \mathrm{SAIL}(\mathbb{D}, K, reps, maxIter)$;
 2.   Create subthreads: $subThread[s]$, $s = 1, 2, \cdots, nSub$;
 3.   for $i = 1 : maxInv$
 4.       for $j = 1 : nSub$
 5.           $[objVal[j], label[j]] = SubThread[j]$.start();
 6.       end for
 7.       for $j$=1:$nSub$
 8.           $SubThread[j]$.wait();
 9.       end for
10.       $j^* = \arg\min_j \{objVal[j], j = 1, \cdots, nSub\}$;
11.       if $objVal[j^*] < objVal$
12.           $objVal = objVal[j^*], label = label[j^*]$;
13.       end if
14.   end for
15.   return $objVal^* = objVal$, $label^* = label$, $\pi$;
Subthread:      // take $SubThread[j]$ for example
16.   $t = j, objVal[j] = objVal, label[j] = label$;
17.   while $(t++ < j + k_{max})$
18.       $label' = \mathrm{Shaking}(\mathcal{N}_t, label)$;
19.       Update $objVal'$ and $CluSum'$ accordingly;
20.       for $l = 1 : maxIters$
21.           $\mathrm{LocalSearch}(\mathbb{D}, \pi, objVal', label', cluSum')$;
22.           if $label'$ is unchanged
23.               break;
24.           end if
25.       end for
26.       if $objVal' < objVal[j]$
27.           $objVal[j] = objVal', label[j] = label', t = j$;
28.       end if
29.   end while
30.   return $objVal[j]$, $label[j]$;
```

4.5.2 The PV-SAIL Algorithm

V-SAIL may find a better clustering result by searching inside the neighborhoods via the VNS scheme. The critical problem is, however, VNS usually has a high computational cost. As indicated by Fig. 4.4, V-SAIL searches inside the neighborhoods one by one, and gets back to the first neighborhood after finding a better solution. This can make the computational time uncontrollable. That is why we introduce a “hard” stopping criterion in Line 15. But the problem remains unsolved in another way—given a time constraint, V-SAIL can only search a limited number of neighborhoods, which prevents it from further enhancing the clustering accuracy.

To meet this challenge, we propose a multithreading scheme for V-SAIL: PV-SAIL, which aims to parallel V-SAIL by fully exploiting the power of the multi-core CPU. In general, PV-SAIL is based on the central memory to reduce the cost of communications. The subthreads are invoked in multiple rounds. In each round, the subthreads simultaneously search inside the neighborhoods in different Hamming distances, and the best result will be adopted as the new solution for the search in the next round. Figure 4.5 shows the pseudocodes of the PV-SAIL algorithm. Some notable details are as follows.

Lines 1–15 show the process of the main thread. Note that to avoid unpredictable errors, we let the main thread wait for the termination of all subthreads in Lines 7–9. Lines 16–30 describe the process of each subthread, which is similar to the **r**-step in V-SAIL. Note that to expand the search space for a better result, we let the subthreads search within the neighborhoods in different Hamming distances simultaneously, as indicated by $t = j$ in Line 16. Also, we do not set a hard stopping criterion in the subthreads, since the parameter k_{max} in Line 17 is often set to a small value, say not beyond 5 in our experiments. Finally, the best way to set $nSub$ is to keep consistency with the number of CPU cores. For instance, for most computers having a 4-core CPU and if $k_{max} = 5$, PV-SAIL can search within 20 neighborhoods at short notice. This greatly increases the probability of finding a better clustering result.

In summary, PV-SAIL is a multithreaded version of V-SAIL, which aims to improve the clustering quality of V-SAIL given limited computational time.

4.6 Experimental Results

In this section, we demonstrate the effectiveness of SAIL and its variants for text clustering. Specifically, we will show: (1) the impact of the zero-value dilemma to the traditional Info-Kmeans; (2) the superior performance of SAIL compared with the smoothing technique as well as the spherical K-means; (3) the benefit of using VNS in V-SAIL for the search of better clustering results; (4) the benefit of using multithreading in PV-SAIL for the faster search of solutions.

4.6.1 Experimental Setup

We first introduce the experimental setup, including the information of the data, the clustering tools, and the evaluation measures.

Experimental data. For our experiments, we use a number of real-world text collections. Some characteristics of these data sets are shown in Table 4.2, where CV

Table 4.2 Experimental text data sets

ID	Data	#Instance	#Feature	#Class	CV	Density
D1	classic	7094	41681	4	0.547	0.0008
D2	cranmed	2431	41681	2	0.212	0.0014
D3	fbis	2463	2000	17	0.961	0.0799
D4	k1a	2340	21839	20	1.004	0.0068
D5	k1b	2340	21839	6	1.316	0.0068
D6	la1	3204	21604	6	0.493	0.0048
D7	la2	3075	31472	6	0.516	0.0048
D8	la12	6279	31472	6	0.503	0.0047
D9	ohscal	11162	11465	10	0.266	0.0053
D10	re0	1504	2886	13	1.502	0.0179
D11	sports	8580	126373	7	1.022	0.0010
D12	reviews	4069	126373	5	0.640	0.0015
D13	tr11	414	6429	9	0.882	0.0438
D14	tr12	313	5804	8	0.638	0.0471
D15	tr23	204	5832	6	0.935	0.0661
D16	tr31	927	10128	7	0.936	0.0265
D17	tr41	878	7454	10	0.913	0.0262
D18	tr45	690	8261	10	0.669	0.0340
D19	wap	1560	8460	20	1.040	0.0167

is the coefficient of variation statistic [9] used to characterize the class imbalance of the data sets, and "Density" is the ratio of nonzero feature-values in each text collection. A large CV indicates a severe class imbalance, and a small Density indicates a high sparseness.

The `fbis` data set was obtained from the Foreign Broadcast Information Service data of the TREC-5 collection,[1] and the classes correspond to the categorization used in that collection. The `sports` and `reviews` data sets were derived from the San Jose Mercury newspaper articles that were distributed as part of the TREC collection (TIPSTER Vol. 3). The former contains documents about baseball, basketball, bicycling, boxing, football, golfing and hockey, and the latter contains documents about food, movies, music, radio and restaurants. Data sets `tr11`, `tr12`, `tr23`, `tr31`, `tr41` and `tr45` were derived from TREC-5, TREC-6, and TREC-7 collections. The classes of these data sets correspond to the documents that were judged relevant to particular queries. Data sets `la1`, `la2` and `la12` were obtained from articles of Los Angeles Times that was used in TREC-5. The categories include documents from the entertainment, financial, foreign, metro, national, and sports desks. The `ohscal` data set was obtained from the OHSUMED collection [10], which contains documents from various biological sub-fields. Data sets `k1a`, `k1b` and `wap` were from the WebACE project [7]; each document corresponds to a web page listed in the subject hierarchy of Yahoo! In particular, `k1a` and `k1b` contain exactly the same

[1] http://www.trec.nist.gov

set of documents but the former contains a finer-grain categorization. The `classic` data set was obtained by combining the CACM, CISI, CRANFIELD, and MEDLINE abstracts that were used in the past to evaluate various information retrieval systems. Data set `cranmed` was attained in a similar way. Finally, the data set `re0` was from Reuters-21578 collection Distribution 1.0.[2] For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [15].

Clustering tools. In the experiments, we employ four types of clustering tools. The first one is SAIL and its variants V-SAIL and PV-SAIL, coded by ourselves in C++. The other three are well-known software packages for K-means clustering, including MATLAB v7.1,[3] CO-CLUSTER v1.1,[4] and CLUTO v2.1.1.[5]

The MATLAB implementation of K-means is a batch-learning version which computes the distances between instances and centroids. We extend it to include more distance functions such as KL-divergence. In our experiments, it works as an implementation of Info-Kmeans which has to compute the KL-divergence directly.

CO-CLUSTER is a C++ program which implements the information-theoretic co-clustering algorithm [5]. Although it still computes the "instance-centroid" KL-divergences, it provides additional methods to improve the clustering performances such as annealing, batch and local search, etc.

CLUTO is a software package for clustering high-dimensional data sets. Specifically, its K-means implementation with cosine similarity as the proximity function shows superior performances in text clustering [21]. In the experiments, we compare CLUTO with SAIL on a number of real-world data sets.

Note that the parameters of the four K-means implementations are set to match one another for the purpose of comparison, and the cluster number K is set to match the number of true classes of each data set.

Validation measures. Many recent studies on clustering use the Normalized Mutual Information (*NMI*) to evaluate the clustering performance [22]. For the purpose of comparison, we also use *NMI* in our experiments, which can be computed as:

$$NMI = I(X, Y)/\sqrt{H(X)H(Y)}, \tag{4.11}$$

where the random variables X and Y denote the cluster and class sizes, respectively. The value of *NMI* is in the interval: $[0, 1]$, and a larger value indicates a better clustering result. Note that if we use the arithmetic mean $(H(X) + H(Y))/2$ rather than the geometric mean $\sqrt{H(X)H(Y)}$ in Eq. (4.11), *NMI* is equivalent to VI_n used in Chap. 3. Chapter 5 provides more details of external validation measures for K-means clustering.

[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[3] http://www.mathworks.cn/help/toolbox/stats/kmeans.html

[4] http://www.cs.utexas.edu/users/dml/Software/cocluster.html

[5] http://www.glaros.dtc.umn.edu/gkhome/views/cluto

Table 4.3 Clustering results of Info-Kmeans (MATLAB)

Data	*NMI*	CV_0	CV_1
`tr23`	0.035	0.935	2.435
`tr45`	0.022	0.669	3.157

Table 4.4 Clustering results of CO-CLUSTER

Data	Search	Annealing	*NMI*	CV_0	CV_1
tr12	Batch	1.0	0.058	0.638	0.295
		0.5	0.040	0.638	0.374
		None	0.031	0.638	0.376
	Local	1.0	0.045	0.638	0.334
		0.5	0.059	0.638	0.339
		None	0.048	0.638	0.461
tr31	Batch	1.0	0.007	0.936	0.426
		0.5	0.011	0.936	0.362
		None	0.010	0.936	0.405
	Local	1.0	0.014	0.936	0.448
		0.5	0.009	0.936	0.354
		None	0.011	0.936	0.365

4.6.2 The Impact of Zero-Value Dilemma

Here, we demonstrate the negative impact of the zero-value dilemma to Info-Kmeans. Since the MATLAB implementation of Info-Kmeans can handle infinity (denoted as INF), we select `tr23` and `tr45` as the test data sets and apply MATLAB Info-Kmeans for testing without smoothing. The clustering results are shown in Table 4.3, where CV_0 and CV_1 represent the distributions of the class and cluster sizes, respectively.

As indicated by the close-to-zero *NMI* values, the clustering performance of MATLAB Info-Kmeans without smoothing is extremely poor. Also, by comparing the CV_0 and CV_1 values, we found that the distributions of the resulting cluster sizes are much more skewed than the distributions of the class sizes. In fact, for both data sets, nearly all the text vectors have been assigned to ONE cluster! This experimental result well confirms our analysis in Sect. 4.3; that is, Info-Kmeans will face the serious zero-value dilemma when clustering highly sparse text data sets.

Furthermore, we test CO-CLUSTER on `tr12` and `tr31` data sets. As mentioned above, CO-CLUSTER also computes the KL-divergence values in the clustering process, but it provides various search modes and the annealing technique to avoid poor local minima. Table 4.4 shows the clustering results, where "Search" and "Annealing" indicate the search modes and annealing parameters, respectively.

In Table 4.4, we can observe that the use of annealing technique and different search modes does not improve the clustering performance. The near-to-zero *NMI*

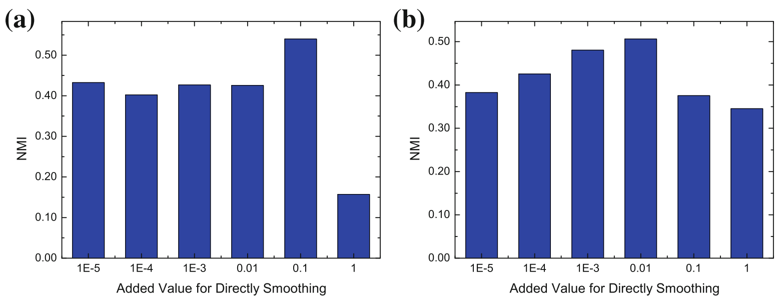


Fig. 4.6 The effect of data smoothing. **a** Data set: `tr11`; **b** Data set: `tr45`

values indicate the poor clustering performance. This again confirms that the direct computation of the KL-divergence values is infeasible for sparse data sets. Another interesting observation is that, the clusters produced by CO-CLUSTER are much more balanced than the clusters produced by MATLAB Info-Kmeans, as indicated by the much smaller CV_1 values.

4.6.3 The Comparison of SAIL and the Smoothing Technique

Here we illustrate the effect of the smoothing technique on sparse data sets. In this experiment, we use MATLAB Info-Kmeans and take seven text collections for illustration. Figure 4.6 shows the clustering results on data sets `tr11` and `tr45`, where the added small values increase gradually along the horizon axis.

One observation is that data smoothing indeed improves the clustering performance of Info-Kmeans, from nearly zero to about 0.3 *NMI*. This result implies that the smoothing technique does help Info-Kmeans get out of the zero-value dilemma, although the performance is still far from satisfactory. Another interesting observation is that the optimal added-value (OAV) is varied for different data sets. For instance, while $OAV \approx 0.1$ for `tr11`, $OAV \approx 0.01$ for `tr45`. This implies one issue with data smoothing in practice; that is, it is difficult to have the optimal smoothing effect. Nevertheless, a general rule is that we should avoid setting extreme values for added values. A tiny value may not help walk out of the zero-value dilemma, but a large value may damage the integrity of the data instances, and thus lead to poorer clustering performance instead. Figure 4.6b well illustrates this point, in which an added-value smaller or larger than 0.01 will do harm to the clustering quality.

For the purpose of comparison, we also test SAIL on these data sets. The parameters are set as follows: (1) $\pi_x = 1$, for any $x \in \mathbb{D}$; (2) no row or column modeling in Line 2 of Fig. 4.1; (3) the initialization mode in Line 5 of Fig. 4.1 is "random center"; (4) $reps = 1$, but we repeat SAIL 10 times for each data set and have the averaged *NMI* value returned. Unless otherwise stated, these are the default settings of SAIL in all of our experiments.

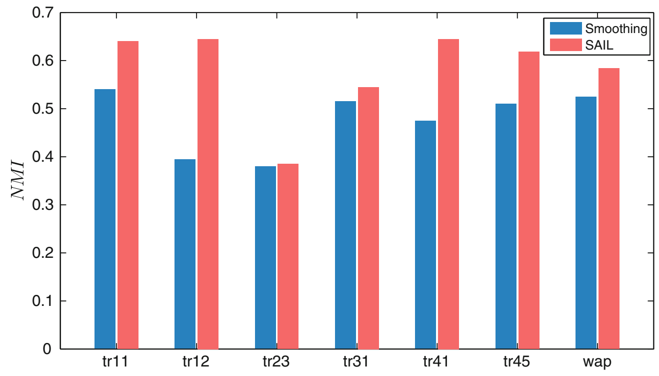


Fig. 4.7 Smoothing versus SAIL

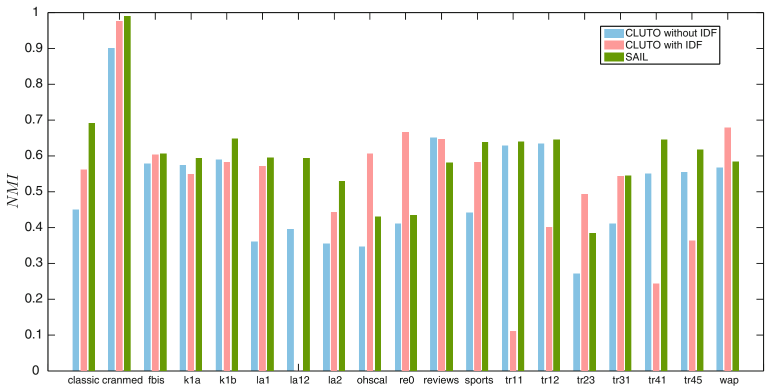


Fig. 4.8 Spherical K-means versus SAIL

Figure 4.7 shows the comparison results. As can be seen, the clustering results of SAIL are consistently superior to the results using the smoothing technique. For some data sets such as `tr11`, `tr12`, `tr41` and `tr45`, SAIL takes the lead with a wide margin. Note that for the smoothing method, we tried a series of added values, i.e. 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, and selected the best one for comparison.

In summary, while the traditional data smoothing technique can improve the performance of Info-Kmeans on sparse data sets, it changes the data integrity and has difficulty in setting the optimal value added to the data. In contrast, SAIL has no parameter setting issue and can lead to consistently better clustering performances than the smoothing technique. This indicates that SAIL is a better solution for the zero-value dilemma.

4.6.4 *The Comparison of SAIL and Spherical K-means*

In this section, we compare the clustering performances between SAIL and the spherical K-means. In the literature, people have shown that spherical K-means usually produces better clustering results than traditional K-means [22]. And the CLUTO version of the spherical K-means even shows superior performances on text collections, which makes it the benchmark method for text clustering. However, we would like to show in this experiment that the performance of SAIL is comparable to or even slightly better than the spherical K-means in CLUTO.

In this experiment, the parameter settings in CLUTO are as follows: clmethod = direct, crfun = i2, sim = cosine, colmodel = none, ntrials = 10. For SAIL, we use the default settings for the previous experiment. Figure 4.8 shows the clustering results, where "CLUTO without IDF" indicates the spherical clustering results. As can be seen, SAIL shows consistently higher clustering quality on nearly all 19 data sets except `reviews`. For some data sets, for example `classic`, `la1`, `la2`, `la2`, `sports` and `tr31`, SAIL even shows dominant advantages. This result demonstrates that SAIL is particularly suitable for text clustering, even compared with the state-of-the-art methods.

Since it is reported that feature weighting often makes great impact to the spherical K-means [21], we also compare the clustering performances of SAIL and CLUTO with Inverse-Document-Frequency (IDF) weighting [20]. Figure 4.8 shows the comparison result. Two observations are notable as follows. First, although CLUTO with IDF improves the clustering quality of CLUTO without IDF on 11 out of 19 data sets, it still shows poorer performance than SAIL to 14 out of 19 data sets. Second, for some data sets, such as `la12`, `tr11`, `tr12`, `tr41` and `tr45`, the IDF scheme actually seriously degrades the clustering performance of CLUTO. These observations imply that feature weighting is an X-factor for the spherical K-means without the guidance of extra information. In contrast, SAIL with default settings shows consistent clustering performances and therefore is more robust in practice.

In summary, compared with the benchmark spherical K-means algorithm, SAIL shows merits in providing competitive and robust clustering results on a number of real-world text collections.

4.6.5 *Inside SAIL*

In this section, we take a further step to explore the properties of SAIL. Specifically, we will first examine the computational efficiency of SAIL and its convergence, and then study how feature weighting, instance weighting, and bisecting scheme can take effect on SAIL.

First, we observe the impact of data sparseness on SAIL. To this end, we compute the correlation between "Density" in Table 4.2 and the clustering results of SAIL measured by *NMI*. The result is -0.245, which indicates a weak negative

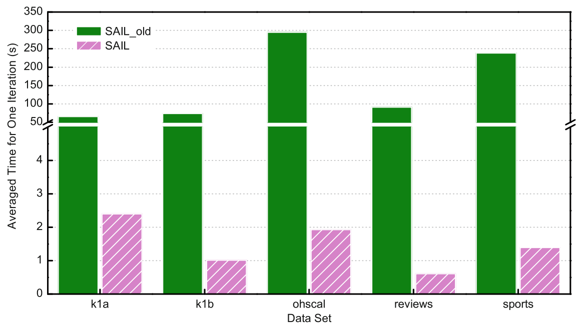


Fig. 4.9 SAIL versus SAIL_old on computational time

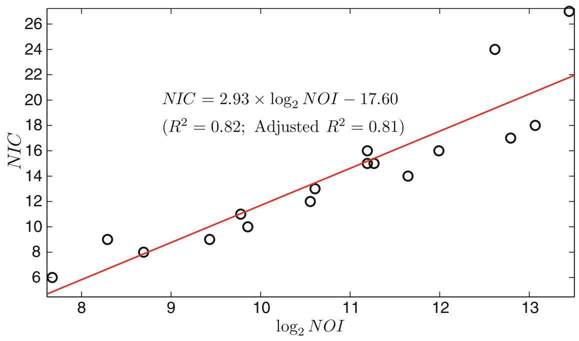


Fig. 4.10 The fast convergence of SAIL

correlation. In other words, SAIL seems to be more appealing in dealing with sparse text collections, which used to be the source of the zero-value dilemma.

Next, we study the efficiency of SAIL. As mentioned in Sect. 4.4.2, we employ Eq. (4.10) instead of Eq. (4.8) for the computation of SAIL. Here we illustrate the empirical results using these two schemes in Fig. 4.9, where "SAIL_old" represents SAIL using Eq. (4.8). As can be seen, by using Eq. (4.10), SAIL improves the clustering efficiency greatly. For instance, for the large-scale data sets such as `ohscal`, `reviews` and `sports`, SAIL is faster than SAIL_old by roughly two orders of magnitude. Note that the vertical axis of Fig. 4.9 represents the average time consumed by running one iteration (refer to Line 7 of Fig. 4.1) of SAIL.

Figure 4.10 then shows the convergence situation of SAIL, which presents the relationship between the number of instances (NOI) and the number of iterations for convergence (NIC). In the figure, we can observe that NIC is typically smaller than 20, except for data sets `la12`, `la2` and `ohscal` whose NIC values are larger than 20 but smaller than 30. Also, there is a significant linear correlation between NIC and $\log_2$ NOI, which indicates that NOI can only lead to a logarithmic increase in NIC. Note that NIC in Fig. 4.10 is the average value of 10 runs of SAIL, and we have deleted the outliers `cranmed` (NIC $= 3$) and `la12` (NIC $= 29$) for a better fit in Fig. 4.10.

Table 4.5 Impact of parameter settings on SAIL

ID	Data	Default	Feat. Wgt.	Text Wgt.	Bisecting		
			(IDF)	(Prob.)	Size	Obj	Obj/Size
D1	classic	0.691	0.668	0.632	0.668	**0.729**	0.598
D2	cranmed	0.990	**0.995**	0.990	0.990	0.990	0.990
D3	fbis	0.606	**0.608**	0.554	0.569	0.569	0.579
D4	k1a	0.594	0.551	**0.599**	0.548	0.567	0.532
D5	k1b	0.648	0.568	0.607	0.538	0.554	0.416
D6	la1	0.595	0.571	0.559	0.578	0.574	0.471
D7	la12	0.594	0.578	0.329	0.559	0.554	0.484
D8	la2	0.530	0.501	0.498	**0.549**	0.535	0.492
D9	ohscal	0.430	0.402	0.291	0.408	0.407	0.384
D10	re0	0.434	0.360	0.408	0.401	0.407	0.252
D11	sports	0.685	0.650	0.533	0.641	0.002	0.001
D12	reviews	0.581	0.508	0.338	0.545	0.570	0.542
D13	tr11	0.640	0.584	0.578	**0.642**	**0.658**	0.498
D14	tr12	0.645	0.556	0.447	0.553	0.545	0.415
D15	tr23	0.385	**0.403**	0.147	0.364	0.327	0.229
D16	tr31	0.545	0.525	**0.550**	**0.616**	**0.550**	**0.569**
D17	tr41	0.645	0.623	**0.684**	0.585	0.589	0.592
D18	tr45	0.618	**0.685**	0.527	**0.630**	0.602	0.543
D19	wap	0.584	0.543	**0.594**	0.558	0.548	0.549

In what follows, we investigate other factors that may impact the performance of SAIL. Specifically, we introduce the feature weighting, text weighting and bisecting schemes for SAIL, and observe the change of clustering performance. Table 4.5 shows the results. Note that "Default" represents SAIL with defaulting settings, "feature weighting" represents SAIL using the IDF scheme, and "text weighting" represents SAIL for documents weighted by $p(x)$ in Theorem 4.1. As to "Bisecting", it represents a top-down divisive variant of SAIL. That is, it first divides all instances into two clusters using SAIL; and then repeatedly selects one cluster according to a certain criterion, and divides that cluster into two sub-clusters using SAIL again; the procedure will continue unless the desired K clusters are found. In our experiment, we use three cluster-selection criteria for bisecting SAIL, where "Size" chooses the largest cluster, "Obj" chooses the one with the largest objective-function value, and "Obj/Size" chooses the one with the largest averaged objective-function value.

As can be seen in Table 4.5, one observation is that "Text Weighting", "Bisecting with Obj" and "Bisecting with Obj/Size" often produce very poor clustering results (scores underlined), and therefore cannot be used for SAIL enhancement. In contrast, "Feature Weighting" and "Bisecting with Size" produce relatively robust results, and in some cases even improve SAIL slightly (scores in bold), thus can be considered as valuable supplements to SAIL. Moreover, the bisecting scheme has another appealing merit; that is, the bisecting SAIL is often more efficient than SAIL, as illustrated by Fig. 4.11. Nonetheless, SAIL with default settings still produce competitive

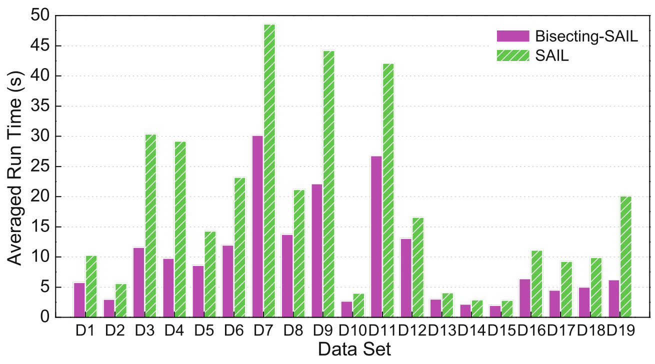


Fig. 4.11 SAIL versus Bisecting-SAIL on computational time

clustering results in most cases, and therefore is the most robust one among different settings.

4.6.6 The Performance of V-SAIL and PV-SAIL

Here we illustrate the improvements of SAIL due to the introduction of the VNS and multithreading schemes.

For the experiments of V-SAIL, we set $k_{max} = 12$, and return the average *NMI* values of 10 repetitions. The stopping criterion for V-SAIL is the maximum times for calling "LocalSearch", which we set to 500. Figure 4.12 shows the comparison results of V-SAIL and SAIL. As can be seen from the figure, V-SAIL generally achieves higher *NMI* scores than SAIL. For some data sets, such as `fbis`, `la2`, `tr31`, `tr41`, and `tr45`, the improvements are quite significant. Nonetheless, it is worthy of noting that for many data sets, V-SAIL only achieves slightly better clustering results than SAIL. This in turn implies that in many cases SAIL alone is capable of finding good enough solutions.

Next, we demonstrate the benefits of using a multithreading scheme for SAIL. To this end, we compare PV-SAIL with V-SAIL by observing their objective-function values at each time point. Three large-scale data sets, i.e. `ohscal`, `sports` and `reviews`, are selected for the comparison study. For V-SAIL, we set $k_{max} = 12$. Since the PC used for experiments has a four-core CPU, we run PV-SAIL with 2, 3 and 4 subthreads, respectively. In each subthread, we let $k_{max} = 3$. The stopping criteria of V-SAIL and PV-SAIL are removed for the comparison purpose. Figure 4.13 depicts the results. As can be seen, PV-SAIL typically obtains substantially lower objective-function values at each time point after the first call of SAIL in the initialization (highlighted by a vertical line named "SAIL finished").

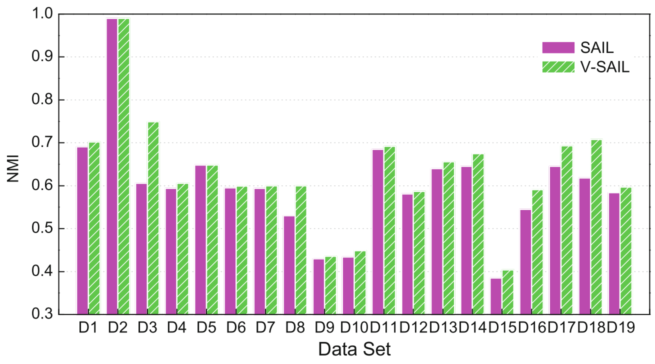


Fig. 4.12 SAIL versus V-SAIL

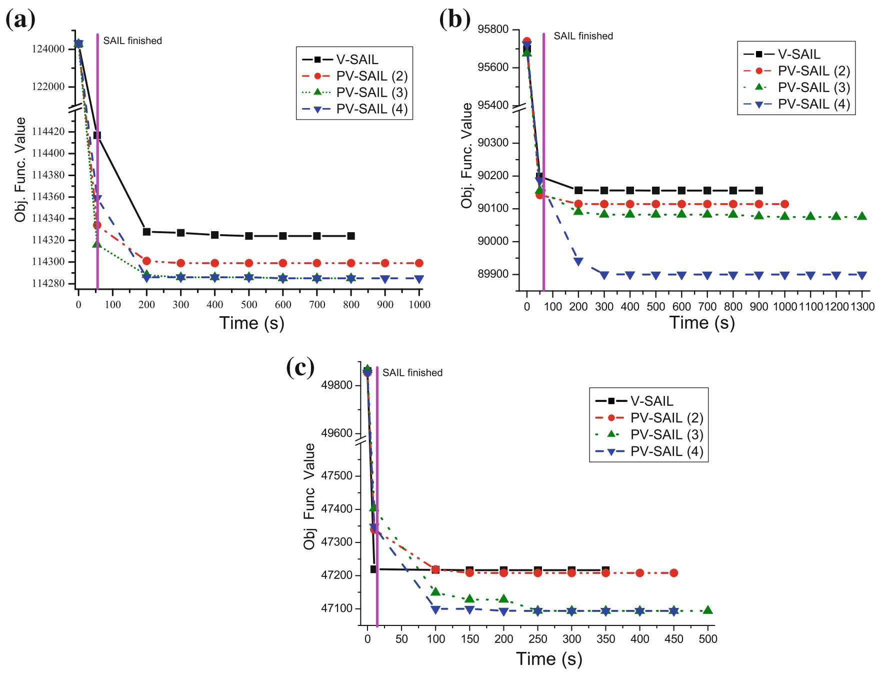


Fig. 4.13 PV-SAIL versus V-SAIL on computational time. **a** Data set: `ohscal`; **b** Data set: `sports`; **c** Data set: `reviews`

In summary, V-SAIL indeed improves the clustering performance of SAIL by using the VNS scheme. The higher time-cost of V-SAIL is then lowered by further introducing the multithreading scheme, which gives birth to the PV-SAIL algorithm.

4.7 Related Work

In the literature, great research efforts have been devoted to incorporating information-theoretic measures into existing clustering algorithms, such as K-means [4–6, 16, 19, 22]. However, the zero-value dilemma remains a critical challenge.

For instance, Dhillon et al. proposed information-theoretic K-means, which used the KL-divergence as the proximity function [4]. While the authors noticed the "infinity" values when computing the KL-divergence, they did not provide specific solutions to this dilemma. In addition, Dhillon et al. [5] further extended information-theoretic K-means to the so-called information-theoretic co-clustering. This algorithm is the two-dimensional version of information-theoretic K-means which monotonically increases the preserved mutual information by interwinding both the row and column clusterings at all stages. Again, however, there is no solution provided for handling the zero-value dilemma when computing the KL-divergence. The information bottleneck (IB) is similar to Info-Kmeans in preserving mutual information [19]. Slonim and Tishby [16] also found that the IB-based word clustering can lead to the zero-value dilemma. They suggested to use the smoothing method by adding 0.5 to each entry of the text data. Meila and Heckerman [13] compared hard and soft assignment strategies for text clustering using multinomial models from which Info-Kmeans can be derived. However, they also omitted the details of how to handle the zero denominator in computations.

Since many proximity functions such as the squared Euclidean distance and the cosine similarity can be used in K-means clustering [18], a natural idea is to compare their performances in practice. [22] is one of such studies. The authors argued that the spherical K-means produces better clustering results than Info-Kmeans. Steinbach et al. [17], Zhao and Karypis [21] and Banerjee et al. [1] also showed the merits of spherical K-means in text clustering. In many studies, the spherical K-means in CLUTO has become the benchmark for text clustering. However, these studies did not tell us why Info-Kmeans shows inferior performance to spherical K-means and how to enhance Info-Kmeans.

Our study indeed fills this crucial void by proposing the SAIL algorithm and its variants to handle the zero-value dilemma, and improving the clustering performance of Info-Kmeans to a level competitive to the spherical K-means.

4.8 Concluding Remarks

This chapter studied the problem of exploiting KL-divergence for information-theoretic K-means clustering (denoted as Info-Kmeans for short). In particular, we revealed the dilemma of Info-Kmeans for handling high-dimensional sparse text data; that is, the centroids in sparse data usually contain zero-value features, and thus lead to infinite KL-divergence values. This makes it difficult to use KL-divergence as a criterion for assigning objects to the centroids. To deal with this, we developed a Summation-bAsed Incremental Learning (SAIL) algorithm, which can avoid the zero-value dilemma by computing the Shannon entropy instead of the KL-divergence. The effectiveness of this replacement is guaranteed by an equivalent mathematical transformation in the objective function of Info-Kmeans. Moreover, we proposed two variants, i.e. V-SAIL and PV-SAIL, to further enhance the clustering ability of SAIL. Finally, as demonstrated by extensive text corpora in our experiments, SAIL can greatly improve the performance of Info-Kmeans on high-dimensional sparse text data. V-SAIL and PV-SAIL further improve the clustering performance of SAIL in terms of quality and efficiency, respectively.

References

1. Banerjee, A., Dhillon, I., Ghosh, J., Sra, S.: Clustering on the unit hypersphere using von mises-fisher distributions. J. Mach. Learn. Res. **6**, 1345–1382 (2005)
2. Brand, L.: Advanced Calculus: An Introduction to Classical Analysis. Dover, New York (2006)
3. Cover, T., Thomas, J.: Elements of Information Theory, 2nd edn. Wiley-Interscience, New York (2006)
4. Dhillon, I., Mallela, S., Kumar, R.: A divisive information-theoretic feature clustering algorithm for text classification. J. Mach. Learn. Res. **3**, 1265–1287 (2003a)
5. Dhillon, I., Mallela, S., Modha, D.: Information-theoretic co-clustering. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 89–98 (2003b)
6. Elkan, C.: Clustering documents with an exponential-family approximation of the dirichlet compound multinomial distribution. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 289–296 (2006)
7. Han, E.H., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Webace: a web agent for document categorization and exploration. In: Proceedings of the 2nd International Conference on Autonomous Agents, pp. 408–415 (1998)
8. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**, 449–467 (2001)
9. Hendricks, W., Robey, K.: The sampling distribution of the coefficient of variation. Ann. Math. Stat. **7**(3), 129–132 (1936)
10. Hersh, W., Buckley, C., Leone, T., Hickam, D.: Ohsumed: an interactive retrieval evaluation and new large test collection for research. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 192–201 (1994)
11. Kullback, S., Leibler, R.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
12. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
13. Meila, M., Heckerman, D.: An experimental comparison of model-based clustering methods. Mach. Learn. **42**, 9–29 (2001)
14. Mladenovic, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
15. Porter, M.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
16. Slonim, N., Tishby, N.: The power of word clusters for text classification. In: Proceedings of the 23rd European Colloquium on Information Retrieval Research (2001)
17. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of the KDD Workshop on Text Mining (2000)

18. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Upper Saddle River (2005)
19. Tishby, N., Pereira, F., Bialek, W.: The information bottleneck method. In: Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing (1999)
20. Wu, H., Luk, R., Wong, K., Kwok, K.: Interpreting tf-idf term weights as making relevance decisions. ACM Trans. Inf. Syst. **26**(3), 1–37 (2008)
21. Zhao, Y., Karypis, G.: Criterion functions for document clustering: experiments and analysis. Mach. Learn. **55**(3), 311–331 (2004)
22. Zhong, S., Ghosh, J.: Generative model-based document clustering: a comparative study. Knowl. Inf. Syst. **8**(3), 374–384 (2005)

Chapter 5
Selecting External Validation Measures for K-means Clustering

5.1 Introduction

Clustering validation has long been recognized as one of the critical issues essential to the success of clustering applications [12]. Despite the vast amount of research efforts devoted to this problem [9], there is yet no consistent and conclusive solution to cluster validation, and the best suitable measures to use in practice remain unclear. Some interesting problems, such as the effect of measure normalization, the similar behaviors of some measures, and the mathematical properties of measures, are still open for a systematic study. Given the fact that different validation measures may be appropriate for different clustering algorithms, it is necessary to have a focused study of cluster validation measures on a specified clustering algorithm at one time.

To that end, in this chapter, we limit our scope to provide an organized study of external validation measures for K-means clustering [15]. The rationale of this pilot study is as follows. On one hand, while K-means is a well-known, widely used, and successful clustering method, the way to validate the clustering results of K-means is far from normative and tends to be arbitrary in the literature. On the other hand, as internal validation measures often make latent assumptions on the formation of cluster structures, and usually have much higher computational complexity, more research in recent years prefers to use external measures for cluster validity, when the purpose is only to assess clustering algorithms and the class labels are available.

Along this line, we present a thorough study of 16 external measures for K-means clustering validation. Specifically, we first establish a filtering criterion based on the uniform effect of K-means to identify the defective measures that cannot detect the uniform effect. We also show the interesting fact that some other existing measures are right the enhanced versions of these defective measures. In addition, we study the normalization issues of external measures and provide normalization solutions to the measures in the pool. The key challenge here is to identify the lower and upper bounds of validation measures based on the multivariate hypergeometric distribution [3]. The importance of measure normalization is also carefully examined from various aspects. Finally, we reveal some key properties of these external measures, such as consistency, sensitivity, and symmetry. These properties can serve as the guidance for the selection of validation measures in different application scenarios.

Most importantly, we provide a guide line to select the most suitable validation measures for K-means clustering. After carefully profiling these validation measures, we believe it is most suitable to use the normalized van Dongen criterion (VD_n), the normalized Variation of Information measure (VI_n), and the normalized Rand statistic (R_n) for K-means clustering validation. VD_n has a simple computation form, and satisfies some mathematical properties. For the case that the clustering performance is hard to distinguish, however, we may want to use VI_n instead for its high sensitivity. R_n is more complicated in computation, but has a clear statistical meaning and a wide range of value.

The remainder of this chapter is organized as follows. Section 5.2 presents the external measures we aim to study. In Sect. 5.3, we identify and explore some defective measures. Section 5.4 highlights the importance of and the way to measure normalization. In Sect. 5.5, some interesting properties are presented to guide the selection of measures. Finally, we conclude our work in Sect. 5.6.

5.2 External Validation Measures

In this section, we introduce a suite of 16 widely used external clustering validation measures. To the best of our knowledge, these measures represent a good coverage of the validation measures available in different fields, such as data mining, information retrieval, machine learning, and statistics. A common ground of these measures is that they can be computed by the contingency matrix as follows.

The contingency matrix. Given a data set D with n objects, assume that we have a partition $P = \{P_1, \ldots, P_K\}$ of D, where $\bigcup_{i=1}^{K} P_i = D$ and $P_i \bigcap P_j = \phi$ for $1 \le i \ne j \le K$, and K is the number of clusters. If we have the class labels (i.e. true cluster labels) of the data, we can have another partition on D: $C = \{C_1, \ldots, C_{K'}\}$, where $\bigcup_{i=1}^{K'} C_i = D$ and $C_i \bigcap C_j = \phi$ for $1 \le i \ne j \le K'$, where K' is the number of classes. Let n_{ij} denote the number of objects in cluster P_i from class C_j, then the overlapped information between the two partitions can be written in the form of a contingency matrix, as shown in Table 5.1. Throughout this chapter, we will use the notations in this contingency matrix.

The measures. Table 5.2 shows the list of measures to be studied. The "Computation" column gives the computational forms of the measures by using the notations in the contingency matrix. Next, we briefly introduce these measures.

The Entropy and Purity are frequently used external measures for K-means [21, 24]. They measure the "purity" of the clusters with respect to the given class labels.

F-measure was originally designed for the evaluation of hierarchical clustering [14, 20], but has also been employed for partitional clustering [21]. It combines the *precision* and *recall* concepts stemming from the information retrieval community.

Table 5.1 The contingency matrix

		Partition C				
		C_1	C_2	$\cdots$	$C_{K'}$	$\sum$
Partition P	P_1	n_{11}	n_{12}	$\cdots$	$n_{1K'}$	$n_{1\cdot}$
	P_2	n_{21}	n_{22}	$\cdots$	$n_{2K'}$	$n_{2\cdot}$
	$\cdot$	$\cdot$	$\cdot$	$\cdots$	$\cdot$	$\cdot$
	P_K	n_{K1}	n_{K2}	$\cdots$	$n_{KK'}$	$n_{K\cdot}$
	$\sum$	$n_{\cdot 1}$	$n_{\cdot 2}$	$\cdots$	$n_{\cdot K'}$	n

The Mutual Information and Variation of Information were developed in the field of information theory [4]. MI measures how much information one random variable can tell about another one [22]. VI measures the amount of information that is lost or gained in changing from the class set to the cluster set [17].

The Rand statistic [19], Jaccard coefficient, Fowlkes and Mallows Index [7], and Hubert's two statistics [10, 11] evaluate the clustering quality by the agreements and/or disagreements of the pairs of data objects in different partitions.

The Minkowski score [1] measures the difference between the clustering results and a reference clustering (true clusters). And the difference is computed by counting the disagreements of the pairs of data objects in two partitions.

The Classification Error takes a classification view on clustering [2]. It tries to map each class to a different cluster so as to minimize the total misclassification rate. The "σ" in Table 5.2 is the mapping of class j to cluster $\sigma(j)$.

The van Dongen criterion [6] was originally proposed for evaluating graph clustering. It measures the representativeness of the majority objects in each class and each cluster.

Finally, the Micro-Average Precision, Goodman and Kruskal coefficient [8], and Mirkin metric [18] are also popular measures. However, the former two are equivalent to the purity measure, and the Mirkin metric is equivalent to the Rand statistic ($M/2\binom{n}{2}+R=1$). As a result, we will not discuss these three measures in the follow study.

In summary, we have 13 (out of 16) candidate measures in the pool. Among them, P, F, MI, R, J, FM, Γ, and Γ' are positive measures—a higher value indicates a better clustering performance. The remaining measures, however, are based on the distance notion, and therefore are negative measures. Throughout this chapter, we will use the acronyms of these measures.

5.3 Defective Validation Measures

In this section, we identify some defective validation measures that tend to generate misleading evaluations for K-means clustering. The *uniform effect* of K-means, thoroughly studied in Chap. 2, is used here as a filtering criterion. That is,

Table 5.2 External cluster validation measures

ID	Measure	Notation	Computation	Range
1	Entropy	E	$-\sum_i p_i(\sum_j \frac{p_{ij}}{p_i} \log \frac{p_{ij}}{p_i})$	$[0, \log K']$
2	Purity	P	$\sum_i p_i(\max_j \frac{p_{ij}}{p_i})$	(0,1]
3	F-measure	F	$\sum_j p_j \max_i (2\frac{p_{ij}}{p_i}\frac{p_{ij}}{p_j}/(\frac{p_{ij}}{p_i}+\frac{p_{ij}}{p_j}))$	(0,1]
4	Variation of information	VI	$-\sum_i p_i \log p_i - \sum_j p_j \log p_j - 2\sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j}$	$[0, 2\log\max(K, K')]$
5	Mutual information	MI	$\sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j}$	$(0, \log K']$
6	Rand statistic	R	$(\binom{n}{2} - \sum_i \binom{n_{i\cdot}}{2} - \sum_j \binom{n_{\cdot j}}{2} + 2\sum_{ij}\binom{n_{ij}}{2})/\binom{n}{2}$	(0,1]
7	Jaccard coefficient	J	$\sum_{ij}\binom{n_{ij}}{2}/(\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2} - \sum_{ij}\binom{n_{ij}}{2})$	[0,1]
8	Fowlkes and Mallows index	FM	$\sum_{ij}\binom{n_{ij}}{2}/\sqrt{\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}}$	[0,1]
9	Hubert Γ statistic I	Γ	$\frac{\binom{n}{2}\sum_{ij}\binom{n_{ij}}{2} - \sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}}{\sqrt{\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}(\binom{n}{2}-\sum_i \binom{n_{i\cdot}}{2})(\binom{n}{2}-\sum_j \binom{n_{\cdot j}}{2})}}$	(−1,1]
10	Hubert Γ statistic II	Γ'	$(\binom{n}{2} - 2\sum_i \binom{n_{i\cdot}}{2} - 2\sum_j \binom{n_{\cdot j}}{2} + 4\sum_{ij}\binom{n_{ij}}{2})/\binom{n}{2}$	[0,1]
11	Minkowski score	MS	$\sqrt{\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2} - 2\sum_{ij}\binom{n_{ij}}{2}}/\sqrt{\sum_j \binom{n_{\cdot j}}{2}}$	$[0, +\infty)$
12	Classification error	ε	$1 - \frac{1}{n}\max_\sigma \sum_j n_{\sigma(j),j}$	[0,1)
13	van Dongen criterion	VD	$(2n - \sum_i \max_j n_{ij} - \sum_j \max_i n_{ij})/2n$	[0, 1)
14	Micro-average precision	MAP	$\sum_i p_i(\max_j \frac{p_{ij}}{p_i})$	(0,1]
15	Goodman–Kruskal coefficient	GK	$\sum_i p_i(1 - \max_j \frac{p_{ij}}{p_i})$	[0,1)
16	Mirkin metric	M	$\sum_i n_{i\cdot}^2 + \sum_j n_{\cdot j}^2 - 2\sum_i \sum_j n_{ij}^2$	$[0, 2\binom{n}{2})$

Note $p_{ij} = n_{ij}/n,\ p_i = n_{i\cdot}/n,\ p_j = n_{\cdot j}/n$

Table 5.3 Two clustering results

	C_1	C_2	C_3	C_4	C_5
Result I					
P_1	10	0	0	0	0
P_2	10	0	0	0	0
P_3	10	0	0	0	0
P_4	0	0	0	10	0
P_5	0	2	6	0	2
Result II					
P_1	27	0	0	2	0
P_2	0	2	0	0	0
P_3	0	0	6	0	0
P_4	3	0	0	8	0
P_5	0	0	0	0	2

If an external validation measure cannot capture the uniform effect of K-means, this measure is not suitable for the cluster validity of K-means.

In what follows, we apply this criterion for the detection of defective measures through a simulation study. The Coefficient of Variation statistic (CV), first introduced in Chap. 2, is also employed to measure the degree of class imbalance. In particular, CV_0 denotes the CV value of the true cluster sizes (i.e. the class sizes), and CV_1 denotes the CV value of the cluster sizes generated by K-means clustering. The difference between CV_1 and CV_0, i.e. $DCV = CV_1 - CV_0$, then indicates the degree of deviation.

5.3.1 The Simulation Setup

Assume we have a text data set containing 50 documents obtained from five classes. The class sizes are 30, 2, 6, 10, and 2, respectively. Thus, we have $CV_0 = 1.166$, which implies a highly skewed class distribution.

For this data set, we assume there are two clustering results, as shown in Table 5.3. In the table, the first result consists of five clusters with extremely balanced sizes, i.e. $CV_1 = 0$. It can be regarded as the clustering result due to the uniform effect of K-means. In contrast, the second result contains five clusters with varying sizes, which lead to $CV_1 = 1.125$. Therefore, judged by the CV statistic, the second result should be better than the first one.

Indeed, if we take a closer look at Result I in Table 5.3, we can find that the first clustering partitions the objects of the largest class C_1 into three balanced subclusters. Meanwhile, the two small classes C_2 and C_5 are totally "disappeared" —

Table 5.4 The cluster validation results

Result	E	P	F	MI	VI	R	J	FM	Γ	Γ'	MS	ε	VD
I	**0.274**	**0.920**	0.617	**1.371**	1.225	0.732	0.375	0.589	0.454	0.464	0.812	0.480	0.240
II	0.396	0.900	**0.902**	1.249	**0.822**	**0.857**	**0.696**	**0.821**	**0.702**	**0.714**	**0.593**	**0.100**	**0.100**

they are overwhelmed in cluster P_5 by the objects from class C_3. In contrast, we can easily identify all the classes in the second clustering result, as they have the majority of objects in the corresponding clusters. Therefore, we can conclude that the first clustering is indeed much worse than the second one.

Next, we proceed to see which external validation measures can rank the two results properly, and thus can detect the uniform effect of K-means.

5.3.2 The Cluster Validation Results

Table 5.4 shows the evaluation results for the two clusterings in Table 5.3, using all 13 external validation measures. The better evaluation given by each validation measure is highlighted in bold.

As shown in Table 5.4, only three measures, E, P and MI, give a higher rank to the first clustering. This implies that they cannot capture the uniform effect of K-means, and therefore are not suitable for evaluating K-means clustering. In other words, these three are defective validation measures.

5.3.3 Exploring the Defective Measures

Here, we explore the issues with the defective measures. First, the problem of the entropy measure (E) lies in the fact that it cannot evaluate the integrity of the classes.

We know $E = -\sum_i p_i \sum_j \frac{p_{ij}}{p_i} \log \frac{p_{ij}}{p_i}$. If we take a random variable view on cluster P and class C, then $p_{ij} = n_{ij}/n$ is the joint probability of the event: $\{P = P_i \bigwedge C = C_j\}$, and $p_i = n_{i\cdot}/n$ is the marginal probability. Therefore,

$$E = \sum_i p_i \sum_j -p(C_j|P_i) \log p(C_j|P_i) = \sum_i p_i H(C|P_i) = H(C|P),$$

where $H(\cdot)$ is the Shannon entropy [4]. The above implies that the entropy measure is nothing but the entropy of C conditioned on P. In other words, if the objects in each large partition are mostly from the same class, the entropy value tends to be small and indicates an excellent clustering quality. This is usually the case for K-means clustering on highly imbalanced data sets, for K-means tends to partition a large class into several pure sub-clusters, and thus damage the integrity of the objects from the same class. The entropy measure cannot capture this information and penalize it.

The Mutual Information measure (MI) is strongly related to the entropy measure. We illustrate this by the following Lemma.

Lemma 5.1 *$MI \Leftrightarrow E$.*

Proof According to the information theory,

$$MI = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{p_i p_j} = H(C) - H(C|P) = H(C) - E.$$

Since $H(C)$ is a constant for any given data set, MI is essentially equivalent to E. □

The purity measure (P) works in a similar way as the entropy measure. That is, it measures the "purity" of each cluster by the ratio of objects from the major class in that cluster. Thus, it has the same problem as the entropy measure for evaluating K-means clustering.

In summary, entropy, purity, and Mutual Information are defective measures for K-means clustering validation.

5.3.4 Improving the Defective Measures

Here, we give the improved versions of the above three defective measures: Entropy, Mutual Information, and Purity. Interestingly, the enhanced measures, i.e. Variation of Information (VI) [17] and van Dongen criterion (VD), have been proposed in the literature for a long time, but hardly any research realizes that they relate to the three defective measures.

Lemma 5.2 *VI is an improved version of E.*

Proof If we view cluster P and class C as two random variables, it has been shown that $VI = H(C) + H(P) - 2MI = H(C|P) + H(P|C)$ [17]. The component $H(C|P)$ is nothing but the entropy measure, and the component $H(P|C)$ is a valuable supplement to $H(C|P)$. That is, $H(P|C)$ evaluates the integrity of each class along different clusters. Thus, we complete the proof. □

By Lemma 5.1, we know MI is equivalent to E. Therefore, VI is also an improved version of MI.

Lemma 5.3 *VD is an improved version of P.*

Proof It is easy to show

$$VD = \frac{2n - \sum_i \max_j n_{ij} - \sum_j \max_i n_{ij}}{2n} = 1 - \frac{1}{2}P - \frac{\sum_j \max_i n_{ij}}{2n}.$$

Apparently, $\sum_j \max_i n_{ij}/n$ reflects the integrity of the classes and is a supplement to the purity measure. □

5.4 Measure Normalization

In this section, we show the importance of measure normalization and provide normalization solutions to some measures whose normalized forms are not available yet.

5.4.1 Normalizing the Measures

Generally speaking, normalizing techniques can be divided into two categories. One is based on a statistical view, which formulates a baseline distribution to correct the measure for randomness. A clustering can then be termed "valid" if it has an unusually high or low value, as measured with respect to the baseline distribution. The normalization scheme takes the form as

$$S_n = \frac{S - E(S)}{\max(S) - E(S)}, \tag{5.1}$$

where $\max(S)$ is the maximum value of measure S, and $E(S)$ is the expected value of S based on the baseline distribution. The difficulty of this scheme usually lies in the computation of $E(S)$.

The other technique uses the minimum and maximum values to normalize the measure into the range of [0,1]. We can also take a statistical view on this technique with the assumption that each measure takes a uniform distribution over the value interval. The normalization scheme is formalized as

$$S_n = \frac{S - \min(S)}{\max(S) - \min(S)}, \tag{5.2}$$

where $\max(S)$ and $\min(S)$ are the maximum and minimum values of measure S. To obtain tighter extreme values is often the key factor that impacts the effect of this normalization scheme.

The normalization of R, FM, $\boldsymbol{\Gamma}$, $\boldsymbol{\Gamma}$', J, and MS. These measures were derived from the statistical community, and usually take the first normalization scheme in Eq. (5.1).

Table 5.5 The normalized measures

ID	Notation	Computation
1	R_n, Γ'_n	$(m - m_1 m_2/M)/(m_1/2 + m_2/2 - m_1 m_2/M)$
2	J'_n, MS'_n	$(m_1 + m_2 - 2m)/(m_1 + m_2 - 2m_1 m_2/M)$
3	FM_n	$(m - m_1 m_2/M)/(\sqrt{m_1 m_2} - m_1 m_2/M)$
4	Γ_n	$(mM - m_1 m_2)/\sqrt{m_1 m_2 (M - m_1)(M - m_2)}$
5	VI_n	$1 + 2\sum_i \sum_j p_{ij} \log(p_{ij}/p_i p_j)/(\sum_i p_i \log p_i + \sum_j p_j \log p_j)$
6	VD_n	$(2n - \sum_i \max_j n_{ij} - \sum_j \max_i n_{ij})/(2n - \max_i n_{i\cdot} - \max_j n_{\cdot j})$
7	F_n	$(F - F_-)/(1 - F_-)$
8	ε_n	$(1 - \frac{1}{n} \max_\sigma \sum_j n_{\sigma(j),j})/(1 - 1/\max(K, K'))$

Note (1) $m = \sum_{i,j} \binom{n_{ij}}{2}$, $m_1 = \sum_i \binom{n_{i\cdot}}{2}$, $m_2 = \sum_j \binom{n_{\cdot j}}{2}$, $M = \binom{n}{2}$ (2) $p_i = n_{i\cdot}/n$, $p_j = n_{\cdot j}/n$, $p_{ij} = n_{ij}/n$ (3) Refer to Table 5.2 for F, and Procedure 1 for F_-

Specifically, Hubert and Arabie [11] suggested to use the multivariate hypergeometric distribution as the baseline distribution, in which the row and column sums are fixed in Table 5.1, but the partitions are randomly selected. This determines the expected value as follows:

$$E\left(\sum_i \sum_j \binom{n_{ij}}{2}\right) = \frac{\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}}{\binom{n}{2}}. \tag{5.3}$$

Based on this value, we can easily compute the expected values of R, FM, Γ and Γ', respectively, since they are the linear functions of $\sum_i \sum_j \binom{n_{ij}}{2}$ under the multivariate hypergeometric distribution assumption. Furthermore, although to obtain the exact maximum values of the measures are computationally prohibitive, we can still reasonably approximate them by 1. Then, according to Eqs. (5.1) and (5.3), we can finally have the normalized R, FM, Γ, and Γ' measures, as shown in Table 5.5.

The normalization of J and MS is a bit complex, since they are not linear to $\sum_i \sum_j \binom{n_{ij}}{2}$. Nevertheless, we can still normalize the equivalent measures converted from them. Let $J' = \frac{1-J}{1+J} = \frac{2}{1+J} - 1$ and $MS' = MS^2$. It is easy to show $J' \Leftrightarrow J$ and $MS' \Leftrightarrow MS$. Then based on the hypergeometric distribution assumption, we have the normalized J' and MS' as shown in Table 5.5. As J' and MS' are negative measures—a lower value implies a better clustering, we normalize them by modifying Eq. (5.1) as $S_n = (S - \min(S))/(E(S) - \min(S))$, where $\min(S)$ indicates a best clustering performance.

Finally, we would like to point out some interrelationships between these measures as follows:

Proposition 5.1

$$(1)\quad (R_n = \Gamma'_n) \Leftrightarrow (J'_n = MS'_n).$$

$$(2)\quad \Gamma_n = \Gamma.$$

Proof (1) Let $m = \sum_{i,j}\binom{n_{ij}}{2}$, $m_1 = \sum_i \binom{n_{i\cdot}}{2}$, $m_2 = \sum_j \binom{n_{\cdot j}}{2}$, $M = \binom{n}{2}$. Then we have

$$R = \frac{M - m_1 - m_2 + 2m}{M},$$

$$\Gamma' = \frac{M - 2m_1 - 2m_2 + 4m}{M}.$$

As m_1, m_2 and M are constants, and the expected value $E(m) = m_1 m_2/M$ under the assumption of multivariate hypergeometric distribution, we can easily compute $E(R)$ and $E(\Gamma')$, and finally have

$$R_n = \frac{R - E(R)}{1 - E(R)} = \Gamma'_n = \frac{\Gamma' - E(\Gamma')}{1 - E(\Gamma')} = \frac{m - m_1 m_2/M}{m_1/2 + m_2/2 - m_1 m_2/M}. \tag{5.4}$$

Likewise, as

$$J' = \frac{2(m_1 + m_2 - m)}{m_1 + m_2}, \quad \text{and}$$
$$MS' = \frac{m_1 + m_2 - 2m}{m_2},$$

we have

$$J'_n = \frac{J' - 0}{E(J') - 0} = MS'_n = \frac{MS' - 0}{E(MS') - 0} = \frac{m_1 + m_2 - 2m}{m_1 + m_2 - 2m_1 m_2/M}. \tag{5.5}$$

According to Eqs. (5.4) and (5.5), we can easily have $R_n + J'_n = 1$, which indicates that R_n is equivalent to J'_n for cluster validity.

(2) As

$$\Gamma_n = \frac{\Gamma - E(\Gamma)}{1 - E(\Gamma)},$$

$\Gamma_n = \Gamma$ if $E(\Gamma) = 0$. As we know,

$$\Gamma = \frac{Mm - m_1 m_2}{\sqrt{m_1 m_2 (M - m_1)(M - m_2)}}.$$

So

$$E(\Gamma) = \frac{M \cdot E(m) - m_1 m_2}{\sqrt{m_1 m_2 (M - m_1)(M - m_2)}} = 0.$$

We thus complete the proof. □

According to Proposition 5.1, the three normalized measures, i.e. Γ'_n, J'_n, and MS'_n, are redundant measures and therefore should be filtered out. We can only consider the three independent normalized measures: R_n, FM_n, and Γ_n, for further study.

The normalization of VI and VD. These two measures often take the second normalization scheme in Eq. (5.2). However, to know the exact maximum and minimum values is often impossible. So we usually turn to a reasonable approximation, e.g. the upper bound of the maximum, or the lower bound of the minimum.

When the cluster structure matches the class structure perfectly, $VI = 0$. So, we have $\min(VI) = 0$. However, finding the exact value of $\max(VI)$ is computationally infeasible. Meila suggested to use $2\log\max(K, K')$ to approximate $\max(VI)$ [17], and the resulted normalized VI is $\frac{VI}{2\log\max(K,K')}$.

The VD in Table 5.2 can be regarded as a normalized measure. In this measure, $2n$ has been taken as the upper bound [6], and $\min(VD) = 0$.

However, we found that the above normalized VI and VD cannot well capture the uniform effect of K-means, because the proposed upper bound for VI or VD is not tight enough. Therefore, we propose new upper bounds as follows:

Lemma 5.4 *Let random variables C and P denote the class and cluster sizes, respectively, and H(·) be the entropy function. Then*

$$VI \leq H(C) + H(P) \leq 2\log\max(K', K).$$

Proof By the definition of VI [16], we have

$$VI = H(C|P) + H(P|C).$$

According to the information theory [4], we have

$$H(C|P) \leq H(C) \leq \log K', \text{and } H(P|C) \leq H(P) \leq \log K,$$

which complete the proof. □

Lemma 5.4 gives an upper bound $H(C) + H(P)$ tighter than $2\log\max(K', K)$ provided by Meila [17]. With this new upper bound, we can have the normalized VI measure, as indicated by VI_n in Table 5.5. It is interesting to note that, if we use $H(P)/2+H(C)/2$ as the upper bound to normalize the defective Mutual Information measure, VI_n is indeed equivalent to the normalized Mutual Information measure MI_n, i.e. $VI_n+MI_n = 1$. This implies that a defective measure may be also corrected by a well designed normalization scheme.

Lemma 5.5 *Let $n_{i\cdot}$, $n_{\cdot j}$ and n be the values in Table 5.1. Then*

$$VD \leq \frac{2n - \max_i n_{i\cdot} - \max_j n_{\cdot j}}{2n} < 1.$$

Proof It is easy to show

$$\sum_i \max_j n_{ij} \geq \max_j \sum_i n_{ij} = \max_j n_{\cdot j},$$

$$\sum_j \max_i n_{ij} \geq \max_i \sum_j n_{ij} = \max_i n_{i\cdot}.$$

Therefore,

$$VD = \frac{2n - \sum_i \max_j n_{ij} - \sum_j \max_i n_{ij}}{2n} \leq \frac{2n - \max_i n_{i\cdot} - \max_j n_{\cdot j}}{2n} < 1.$$

Thus we complete the proof. □

The above two lemmas imply that the tighter upper bounds of VI and VD are the functions of the class and cluster sizes. Using these two new upper bounds, we can derive the normalized VI and VD measures, as shown in Table 5.5.

Procedure 1: The computation of F_-.

```
1: Let $n^* = \max_i n_{i\cdot}$;
2: Sort the class sizes so that $n_{\cdot[1]} \leq n_{\cdot[2]} \leq \cdots \leq n_{\cdot[K']}$;
3: Let $a_j = 0$, for $j = 1, 2, \cdots, K'$;
4: for $j = 1 : K'$
5:    if $n^* \leq n_{\cdot[j]}$
6:       $a_j = n^*$, break;
7:    else
8:       $a_j = n_{\cdot[j]}$, $n^* \leftarrow n^* - n_{\cdot[j]}$;
9:    end if
10: end for
11: $F_- = (2/n) \sum_{j=1}^{K'} a_j / (1 + \max_i n_{i\cdot}/n_{\cdot[j]})$;
```

The normalization of F and $\boldsymbol{\varepsilon}$. As we know, $\max(F) = 1$. Now the goal is to find a tight lower bound. We have the following lemma, which finds a lower bound for F.

Lemma 5.6 *Given F_- computed by Procedure 1, $F \geq F_-$.*

Proof It is easy to show:

$$F = \sum_j \frac{n_{\cdot j}}{n} \max_i \frac{2n_{ij}}{n_{i\cdot} + n_{\cdot j}} \geq \frac{2}{n} \max_i \sum_j \frac{n_{ij}}{n_{i\cdot}/n_{\cdot j} + 1}. \tag{5.6}$$

Let us consider an optimization problem as follows.

$$\min_{x_{ij}} \sum_j \frac{x_{ij}}{n_{i\cdot}/n_{\cdot j} + 1}$$
$$s.t. \sum_j x_{ij} = n_{i\cdot};\ \forall j,\ x_{ij} \leq n_{\cdot j};\ \forall j,\ x_{ij} \in \mathbb{Z}_+.$$

For this optimization problem, to have the minimum objective value, we need to assign as many objects as possible to the cluster with highest $n_{i\cdot}/n_{\cdot j}+1$, or equivalently, with smallest $n_{\cdot j}$. Let $n_{\cdot[0]} \le n_{\cdot[1]} \le \cdots \le n_{\cdot[K']}$ where the virtual $n_{\cdot[0]}=0$, and assume $\sum_{j=0}^{l} n_{\cdot[j]} < n_{i\cdot} \le \sum_{j=0}^{l+1} n_{\cdot[j]}$, $l \in \{0,1,\ldots,K'-1\}$. We have the optimal solution:

$$x_{i[j]} = \begin{cases} n_{\cdot[j]}, \ 1 \le j \le l; \\ n_{i\cdot} - \sum_{k=1}^{l} n_{\cdot[k]}, \ j = l+1; \\ 0, \ l+1 < j \le K'. \end{cases}$$

Therefore, according to Eq. (5.6), $F \ge \frac{2}{n}\max_i \sum_{j=1}^{K'} \frac{x_{i[j]}}{n_{i\cdot}/n_{\cdot[j]}+1}$.

Let $F_i = \frac{2}{n}\sum_{j=1}^{K'} \frac{x_{i[j]}}{n_{i\cdot}/n_{\cdot[j]}+1} = \frac{2}{n}\sum_{j=1}^{K'} \frac{x_{i[j]}/n_{i\cdot}}{1/n_{\cdot[j]}+1/n_{i\cdot}}$. Denote "$x_{i[j]}/n_{i\cdot}$" as "$y_{i[j]}$", and "$\frac{1}{1/n_{\cdot[j]}+1/n_{i\cdot}}$" as "$p_{i[j]}$", we therefore have $F_i = \frac{2}{n}\sum_{j=1}^{K'} p_{i[j]}y_{i[j]}$. Next, we remain to show

$$\arg\max_i F_i = \arg\max_i n_{i\cdot}.$$

Assume $n_{i\cdot} \le n_{i'\cdot}$, and for some l, $\sum_{j=0}^{l} n_{\cdot[j]} < n_{i\cdot} \le \sum_{j=0}^{l+1} n_{\cdot[j]}$, $l \in \{0,1,\ldots,K'-1\}$. This implies that

$$y_{i[j]} \begin{cases} \ge y_{i'[j]}, \ 1 \le j \le l; \\ \le y_{i'[j]}, \ l+1 < j \le K'. \end{cases}$$

Since $\sum_{j=1}^{K'} y_{i[j]} = \sum_{j=1}^{K'} y_{i'[j]} = 1$ and $j \uparrow \ \Rightarrow \ p_{i[j]} \uparrow$, we have $\sum_{j=1}^{K'} p_{i[j]}y_{i[j]} \le \sum_{j=1}^{K'} p_{i[j]}y_{i'[j]}$. Furthermore, according to the definition of $p_{i[j]}$, we have $p_{i[j]} \le p_{i'[j]}$, $\forall\, j \in \{1,\ldots,K'\}$. Therefore,

$$F_i = \frac{2}{n}\sum_{j=1}^{K'} p_{i[j]}y_{i[j]} \le \frac{2}{n}\sum_{j=1}^{K'} p_{i[j]}y_{i'[j]} \le \frac{2}{n}\sum_{j=1}^{K'} p_{i'[j]}y_{i'[j]} = F_i',$$

which implies that "$n_{i\cdot} \le n_{i'\cdot}$" is the sufficient condition for "$F_i \le F_i'$". Therefore, by Procedure 1, we have $F_- = \max_i F_i$, which finally leads to $F \ge F_-$. Thus we complete the proof. □

Therefore, $F_n = (F - F_-)/(1 - F_-)$, as listed in Table 5.5. Finally, we have the following lemma for the ε measure:

Lemma 5.7 *Given $K' \le K$, $\varepsilon \le 1 - 1/K$.*

Proof Assume $\sigma_1 : \{1,\ldots,K'\} \to \{1,\ldots,K\}$ is the optimal mapping of the classes to different clusters, i.e.

$$\varepsilon = 1 - \frac{\sum_{j=1}^{K'} n_{\sigma_1(j),j}}{n}.$$

Then we construct a series of mappings $\sigma_s : \{1, \ldots, K'\} \mapsto \{1, \ldots, K\}$ ($s = 2, \ldots, K$) which satisfy

$$\sigma_{s+1}(j) = \mathrm{mod}(\sigma_s(j), K) + 1, \ \forall j \in \{1, \ldots, K'\},$$

where "mod(x, y)" returns the remainder of positive integer x divided by positive integer y. By definition, σ_s ($s = 2, \ldots, K$) can also map $\{1, \ldots, K'\}$ to K' different indices in $\{1, \ldots, K\}$ as σ_1. More importantly we have $\sum_{j=1}^{K'} n_{\sigma_1(j),j} \geq \sum_{j=1}^{K'} n_{\sigma_s(j),j}$, $\forall s = 2, \ldots, K$, and $\sum_{s=1}^{K} \sum_{j=1}^{K'} n_{\sigma_s(j),j} = n$.

Accordingly, we have $\sum_{j=1}^{K'} n_{\sigma_1(j),j} \geq \frac{n}{K}$, which implies $\varepsilon \leq 1 - 1/K$. The proof is completed. □

Therefore, we can use $1 - 1/K$ as the upper bound of ε, and normalize ε to ε_n, as shown in Table 5.5.

5.4.2 *The Effectiveness of DCV for Uniform Effect Detection*

Here, we present some experiments to show the importance of DCV (i.e. $CV_1 - CV_0$) for selecting validation measures.

Experimental Data. Some synthetic data sets were generated as follows. Assume we have a two-dimensional mixture of two Gaussian distributions. The means of the two distributions are [−2,0] and [2,0], respectively. And their covariance matrices are exactly the same as: $[\sigma^2\ 0;\ 0\ \sigma^2]$.

Therefore, given any specific value of σ^2, we can generate a simulated data set with 6000 instances, n_1 instances from the first distribution, and n_2 instances from the second one, where $n_1 + n_2 = 6000$. To produce simulated data sets with imbalanced class sizes, we set a series of n_1 values: {3000, 2600, 2200, 1800, 1400, 1000, 600, 200}. If $n_1 = 200$, $n_2 = 5800$, we have a highly imbalanced data set with $CV_0 = 1.320$. For each mixture model, we generated 8 simulated data sets with CV_0 ranging from 0 to 1.320. Further, to produce data sets with different clustering tendencies, we set a series of σ^2 values: {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}. As σ^2 increases, the mixture model tends to be more unidentifiable. Finally, for each pair of σ^2 and n_1, we repeated the sampling 10 times, thus we can have the average performance evaluation. In summary, we produced $8 \times 10 \times 10 = 800$ data sets. Fig. 5.1 shows a sample data set with $n_1 = 1000$ and $\sigma^2 = 2.5$.

We also did sampling on a real-world data set `hitech` to get some sample data sets with imbalanced class distributions. This data set was derived from the San Jose

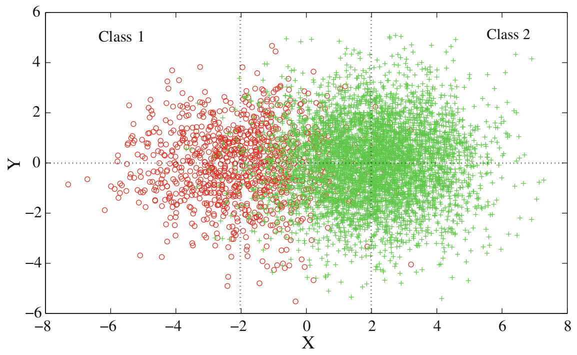


Fig. 5.1 A simulated data set ($n_1 = 1000$, $\sigma^2 = 2.5$). Reprinted from Ref. [23] © 2009 Association for Computing Machinery, Inc. Reprinted by permission

Table 5.6 The sizes of the sampled data sets

Data set	1	2	3	4	5	6	7	8
Class 1	100	90	80	70	60	50	40	30
Class 2	100	90	80	70	60	50	40	30
Class 3	100	90	80	70	60	50	40	30
Class 4	250	300	350	400	450	500	550	600
Class 5	100	90	80	70	60	50	40	30
Class 6	100	90	80	70	60	50	40	30
CV_0	0.490	0.686	0.880	1.078	1.270	1.470	1.666	1.860

Mercury newspaper articles,[1] which contains 2301 documents about computers, electronics, health, medical, research and technology. Each document is characterized by 126373 terms, and the class sizes are 485, 116, 429, 603, 481 and 187, respectively. We carefully set the sampling ratio for each class, and get 8 sample data sets with the class-size distributions (CV_0) ranging from 0.490 to 1.862, as shown in Table 5.6. For each data set, we repeated sampling 10 times, so we can observe the average clustering performance.

Clustering Tools. We used the MATLAB 7.1[2] and CLUTO 2.1.1[3] implementations of K-means. The MATLAB version with the squared Euclidean distance is suitable for low-dimensional and dense data sets, while CLUTO with the cosine similarity is used to handle high-dimensional and sparse data sets. Note that the number of clusters, i.e. K, was set to match the number of true classes.

The Effectiveness of DCV. Figure 5.2 shows the clustering results evaluated by DCV. As can be seen in Fig. 5.2a, for the extreme case of the simulated data sets when $\sigma^2 = 5$, the DCV value decreases as the CV_0 value increases. Note that DCV

[1] http://trec.nist.gov

[2] http://www.mathworks.cn/help/toolbox/stats/kmeans.html

[3] http://glaros.dtc.umn.edu/gkhome/views/cluto

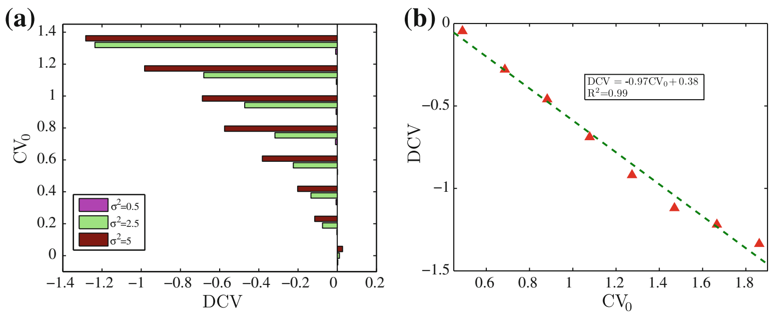


Fig. 5.2 The relationship between CV_0 and DCV. Reprinted from Ref. [23] © 2009 Association for Computing Machinery, Inc. Reprinted by permission

values are usually negative since K-means tends to produce clustering results with relative uniform cluster sizes ($CV_1 < CV_0$). This means that, when the true cluster sizes of data become more skewed, the clustering results generated by K-means tend to be worse. As a result, we can select measures by observing the relationship between the evaluations of the measures and the DCV values. As the DCV value goes down, a good measure is expected to indicate worse clustering performances. Note that, in this experiment, we used the MATLAB version of K-means.

A similar trend can be found in Fig. 5.2b for the sampled data sets. That is, as the CV_0 value goes up, the DCV value decreases, which implies worse clustering performances. Indeed, DCV is a good indicator for finding the measures which cannot capture the uniform effect by K-means clustering. Note that, in this experiment, we used the CLUTO version of K-means to handle the high-dimensionality of text data.

5.4.3 The Effect of Normalization

In this subsection, we show the importance of measure normalization. Along this line, we first apply K-means clustering on the simulated data sets with $\sigma^2 = 5$ and the sampled data sets from `hitech`. Then, both unnormalized and normalized measures are used for cluster validation. Finally, the correlation between DCV and the measures are computed.

We here use the Kendall's rank correlation (κ) [13] to measure the correlation between the measure values and the DCV values. Note that, $\kappa \in [-1, 1]$, with $\kappa = 1$ or -1 indicating the perfect positive or negative rank correlation, and $\kappa = 0$ indicating no significant correlation. Therefore, a good measure is expected to have a high positive κ value.

Table 5.7 shows the results. As can be seen, if we use the unnormalized measures to do cluster validation, only three measures, namely R, Γ, Γ', have strong consistency with DCV on both groups of data sets. VI, VD, and MS even show strong conflicts with DCV on the sampled data sets, for their κ values are all close to -1. In addition, we notice that F, ε, J and FM show weak correlation with DCV.

Table 5.7 The correlation between DCV and the validation measures

κ	VI	VD	MS	ε	F	R	J	FM	Γ	Γ'
Simulated data	**−0.71**	**0.79**	**−0.79**	1.00	1.00	1.00	0.91	**0.71**	1.00	1.00
Sampled data	**−0.93**	**−1.00**	**−1.00**	**0.50**	**0.21**	1.00	**0.50**	**−0.43**	0.93	1.00
κ	VI_n	VD_n	MS'_n	ε_n	F_n	R_n	J'_n	FM_n	Γ_n	Γ'_n
Simulated data	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Sampled data	1.00	1.00	1.00	**0.50**	**0.79**	1.00	1.00	1.00	0.93	1.00

Note Poor correlations are highlighted in bold

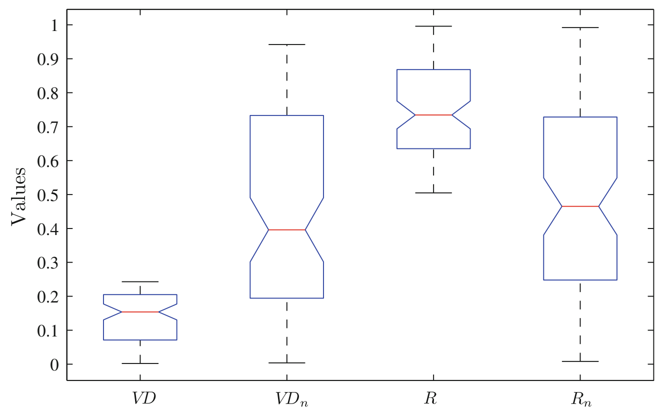


Fig. 5.3 The comparison of value range between unnormalized and normalized measures. Reprinted from Ref. [23] © 2009 Association for Computing Machinery, Inc. Reprinted by permission

Table 5.7 also shows the rank correlations between DCV and the normalized measures. As can be seen, all the normalized measures show perfect consistency with DCV except for F_n and ε_n. This indicates that the normalization is crucial for evaluating K-means clustering. The proposed bounds for the measures are tight enough to capture the uniform effect in the clustering results.

We then focus on F_n and ε_n, both of which are not consistent with DCV in Table 5.7. This indicates that the normalization does not help F and ε too much. The reason is that the proposed lower bound of F and the upper bound of ε are not very tight. Indeed, the normalization of F and ε is very challenging. This is because they both exploit relatively complicated optimization schemes. As a result, we cannot easily compute the expected values or the tighter bounds of extreme values based on the multivariate hypergeometric distribution assumption.

Nevertheless, the above experiments show that the normalization is very important for external measures. In addition, Fig. 5.3 shows the cluster validation results of the measures on all the simulated data sets with σ^2 ranging from 0.5 to 5. It is clear that the normalized measures have much wider value range than the unnormalized

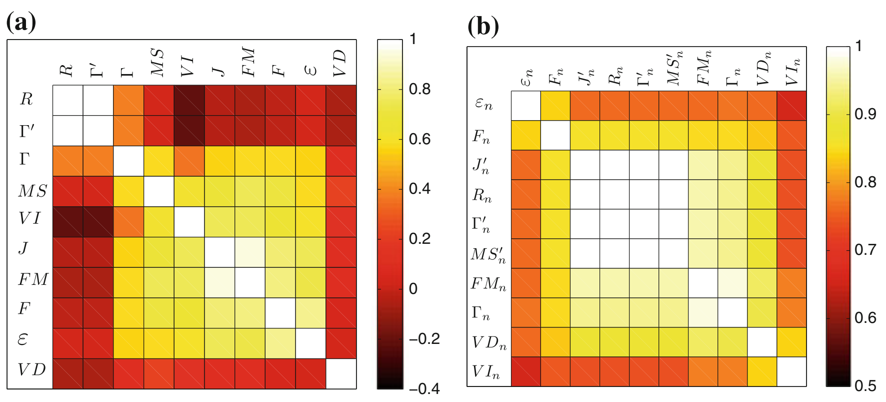


Fig. 5.4 Correlations of the measures. **a** Unnormalized measures; **b** Normalized measures. Reprinted from Ref. [23] © 2009 Association for Computing Machinery, Inc. Reprinted by permission

measures in [0,1], which is also very important for the comparison of clustering results.

In summary, measure normalization can help external validation measures to detect the uniform effect of K-means and gain a wide range of value in [0,1], and therefore is crucial to the success of cluster validity.

5.5 Measure Properties

In this section, we investigate measure properties, which can serve as the guidance for the selection of measures.

5.5.1 The Consistency Between Measures

Here, we define the consistency between a pair of measures in terms of the similarity between their rankings on a series of clustering results. The similarity is measured by the Kendall's rank correlation. And the clustering results are produced by the CLUTO version of K-means clustering on 29 benchmark real-world data sets listed in Table 5.8. In the experiment, the cluster number is set to the true cluster number of each data set.

Figure 5.4a, b show the correlations between the unnormalized and normalized measures, respectively. One interesting observation is that the normalized measures have much stronger consistency than the unnormalized measures. For instance, the correlation between VI and R is merely -0.21, but it reaches 0.74 for the corresponding normalized measures. This observation indeed implies that the normalized

Table 5.8 The benchmark data sets

ID	Data	Source	#Class	#Instance	#Feature	CV_0
1	cacmcisi	CA/CI	2	4663	41681	0.53
2	Classic	CA/CI	4	7094	41681	0.55
3	cranmed	CR/ME	2	2431	41681	0.21
4	fbis	TREC	17	2463	2000	0.96
5	Hitech	TREC	6	2301	126373	0.50
6	k1a	WebACE	20	2340	21839	1.00
7	k1b	WebACE	6	2340	21839	1.32
8	la1	TREC	6	3204	31472	0.49
9	la2	TREC	6	3075	31472	0.52
10	la12	TREC	6	6279	31472	0.50
11	mm	TREC	2	2521	126373	0.14
12	ohscal	OHSUMED	10	11162	11465	0.27
13	re0	Reuters	13	1504	2886	1.50
14	re1	Reuters	25	1657	3758	1.39
15	Sports	TREC	7	8580	126373	1.02
16	tr11	TREC	9	414	6429	0.88
17	tr12	TREC	8	313	5804	0.64
18	tr23	TREC	6	204	5832	0.93
19	tr31	TREC	7	927	10128	0.94
20	tr41	TREC	10	878	7454	0.91
21	tr45	TREC	10	690	8261	0.67
22	Wap	WebACE	20	1560	8460	1.04
23	DLBCL	KRBDSR	3	77	7129	0.25
24	Leukemia	KRBDSR	7	325	12558	0.58
25	LungCancer	KRBDSR	5	203	12600	1.36
26	Ecoli	UCI	8	336	7	1.16
27	Pageblocks	UCI	5	5473	10	1.95
28	Letter	UCI	26	20000	16	0.03
29	Pendigits	UCI	10	10992	16	0.04
-	MIN	-	2	77	7	0.03
-	MAX	-	26	20000	126373	1.95

Note CA-CACM, CI-CISI, CR-CRANFIELD, ME-MEDLINE

measures tend to give more robust validation results, which also agrees with our previous analysis.

Let us take a closer look on the normalized measures in Fig. 5.4b. According to the colors, we can roughly find that R_n, Γ'_n, J'_n, MS'_n, FM_n and Γ_n are more similar to one another, while VD_n, F_n, VI_n and ε_n show inconsistency with others in varying degrees. To gain the precise understanding, we do hierarchical clustering on the measures by using their correlation matrix. The resulting hierarchy can be found in Fig. 5.5 ("s" means the similarity). As we know before, R_n, Γ'_n, J'_n and MS'_n are equivalent, so they have perfect correlation to one another, and form the first group. The second group contains FM_n and Γ_n. These two measures behave similarly, and have just slightly weaker consistency with the measures in the first

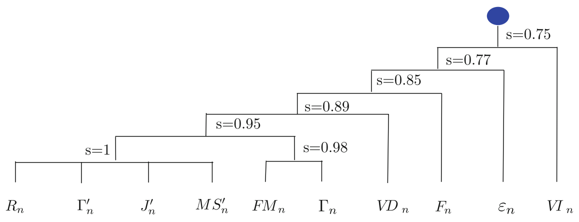


Fig. 5.5 The similarity hierarchy of the measures. Reprinted from Ref. [23] © 2009 Association for Computing Machinery, Inc. Reprinted by permission

Table 5.9 $M(\mathfrak{R}_1) - M(\mathfrak{R}_2)$

	R_n	FM_n	Γ_n	VD_n	F_n	ε_n	VI_n
R_n	0.00	0.09	0.13	0.08	0.10	0.26	−0.01
FM_n	0.09	0.00	0.04	0.00	0.10	0.22	−0.10
Γ_n	0.13	0.04	0.00	0.04	0.14	0.22	−0.06
VD_n	0.08	0.00	0.04	0.00	0.05	0.20	−0.18
F_n	0.10	0.10	0.14	0.05	0.00	0.08	−0.08
ε_n	0.26	0.22	0.22	0.20	0.08	0.00	0.04
VI_n	−0.01	−0.10	−0.06	−0.18	−0.08	0.04	0.00

group. Finally, VD_n, F_n, ε_n and VI_n have obviously weaker consistency with other measures in a descending order.

Furthermore, we explore the source of the inconsistency among the measures. To this end, we divide the data sets in Table 5.8 into two repositories, where $\mathfrak{R}_1$ contains data sets with $CV_0 < 0.8$, and $\mathfrak{R}_2$ contains the rest. Then we compute the correlation matrices of the measures on the two repositories respectively (denoted as $M(\mathfrak{R}_1)$ and $M(\mathfrak{R}_2)$), and observe their difference ($M(\mathfrak{R}_1) - M(\mathfrak{R}_2)$) in Table 5.9. As can be seen, roughly speaking, all the measures except VI_n show weaker consistency with one another on data sets in $\mathfrak{R}_2$. In other words, most measures tend to disagree with one another on data sets with highly imbalanced classes.

5.5.2 Properties of Measures

In this subsection, we investigate some key properties of external clustering validation measures.

Sensitivity. The measures have different sensitivity to the clustering results. Let us illustrate this by an example. For two clustering results in Table 5.10, the differences between them are the numbers in bold. Then we employ the measures on these two clusterings. Validation results are shown in Table 5.11. As can be seen, all the measures show different validation results for the two clusterings except for VD_n and F_n. This implies that VD_n and F_n are less sensitive than other measures. This is due to the fact that both VD_n and F_n use maximum functions, which may loose

Table 5.10 Two clustering results

	C_1	C_2	C_3	$\sum$
Result I				
P_1	**3**	**4**	12	19
P_2	**8**	**3**	12	23
P_3	12	12	0	24
$\sum$	23	19	24	66
Result II				
P_1	**0**	**7**	12	19
P_2	**11**	**0**	12	23
P_3	12	12	0	24
$\sum$	23	19	24	66

Table 5.11 The cluster validation results

	R_n	FM_n	Γ_n	VD_n	F_n	ε_n	VI_n
I	0.16	0.16	0.16	0.71	0.32	0.77	0.78
II	0.24	0.24	0.24	0.71	0.32	0.70	0.62

some information in the contingency matrix. Furthermore, VI_n is the most sensitive measure, since the difference of VI_n values for the two clusterings is the largest.

Impact of the Number of Clusters. We use the data set `la2` in Table 5.8 to show the impact of the number of clusters on the validation measures. We change the number of clusters from 2 to 15 for K-means clustering, and see whether the measures can indicate the number of true clusters. As shown in Fig. 5.6, the normalized measures including VI_n, VD_n and R_n clearly capture the optimal cluster number: 5. Similar results can also be observed for other normalized measures, such as F_n, FM_n and Γ_n. This implies that measure normalization indeed improves the validation performance of the external measures.

Math Properties. We here summarize without proof the five math properties of the validation measures in Table 5.12. Details are illustrated as follows:

Property 5.1 (Symmetry) *A measure O is symmetric, if $O(M^T) = O(M)$ for any contingence matrix M.*

The *symmetry* property treats the pre-defined class structure as one of the partitions. Therefore, the task of cluster validation is the same as the comparison of partitions. This means transposing two partitions in the contingency matrix should not bring any difference to the measure value. This property is not true for F_n which is a typical measure in asymmetry. Also, ε_n is symmetric if and only if $K = K'$.

Property 5.2 (N-invariance) *For a contingence matrix M and a positive integer λ, a measure O is n-invariant, if $O(\lambda M) = O(M)$, where n is the number of objects.*

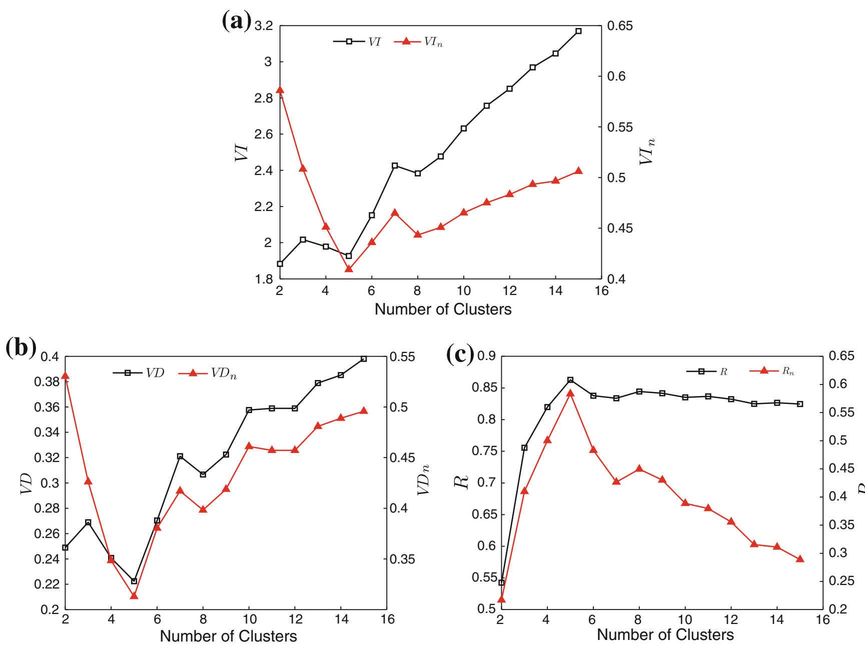


Fig. 5.6 Impact of the number of clusters. Reprinted from Ref. [23] © 2009 Association for Computing Machinery, Inc. Reprinted by permission

Table 5.12 Math properties of measures

Property	F_n	VI_n	VD_n	ε_n	R_n	FM_n	Γ_n
P1	No	Yes	Yes	Yes[a]	Yes	Yes	Yes
P2	Yes	Yes	Yes	Yes	No	No	No
P3	Yes[b]	Yes[b]	Yes[b]	Yes[b]	No	No	No
P4	No	Yes	Yes	No	No	No	No
P5	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Note [a] Yes for $K = K'$ [b] Yes for the unnormalized measures

Intuitively, a mathematically sound validation measure should satisfy the *n-invariance* property. However, three measures, namely R_n, FM_n and Γ_n cannot fulfill this requirement. Nevertheless, we can still treat them as the asymptotically n-invariant measures, since they tend to be n-invariant as the increase of n.

Property 5.3 (Convex additivity) *Let $P = \{P_1, \ldots, P_K\}$ be a clustering, P' be a refinement of P (i.e. P' is the descendant node of node P in the lattice of partitions [17]), and P'_l be the partitioning induced by P' on P_l. Then a measure O is convex additive, if $O(M(P, P')) = \sum_{l=1}^{K} \frac{n_l}{n} O(M(I_{P_l}, P'_l))$, where n_l is the number of data points in P_l, I_{P_l} represents the partitioning on P_l into one cluster, and $M(X, Y)$ is the contingency matrix of X and Y.*

The *convex additivity* property was introduced by Meila [17]. It requires the measures to show additivity along the lattice of partitions. Unnormalized measures including F, VD, VI and ε hold this property. However, none of the normalized measures mentioned in this study holds this property.

Property 5.4 (Left-domain-completeness) *A measure O is left-domain-complete, if, for any contingence matrix M with statistically independent rows and columns,*

$$O(M) = \begin{cases} 0, & O \text{ is a positive measure;} \\ 1, & O \text{ is a negative measure.} \end{cases}$$

When the rows and columns in the contingency matrix are statistically independent, we should expect to see the poorest values of the measures, i.e. 0 for positive measures and 1 for negative measures. Among all the measures, however, only VI_n and VD_n can meet this requirement.

Property 5.5 (Right-domain-completeness) *A measure O is right-domain-complete, if, for any contingence matrix M with perfectly matched rows and columns,*

$$O(M) = \begin{cases} 1, & O \text{ is a positive measure;} \\ 0, & O \text{ is a negative measure.} \end{cases}$$

This property requires measures to show optimal values when the class structure matches the cluster structure perfectly. The above normalized measures hold this property.

5.5.3 Discussions

In a nutshell, among 16 external validation measures shown in Table 5.2, we first know that Mirkin metric (M) is equivalent to Rand statistic (R), and Micro-Average Precision (MAP) and Goodman-Kruskal coefficient (GK) are equivalent to Purity (P) by observing their computational forms. Therefore, the scope of our measure selection is reduced from 16 measures to 13 measures. In Sect. 5.3, our analysis shows that Purity, Mutual Information (MI), and Entropy (E) are defective measures for evaluating K-means clustering. Also, we know that Variation of Information (VI) is an improved version of MI and E, and van Dongen criterion (VD) is an improved version of P. As a result, our selection pool is further reduced to 10 measures.

In addition, as shown in Sect. 5.4, it is necessary to use the normalized measures for evaluating K-means clustering, since the normalized measures can capture the uniform effect by K-means and allow to compare different clustering results on different data sets. By Proposition 5.1, we know that the normalized Rand statistic (R_n) is the same as the normalized Hubert Γ statistic II (Γ'_n). Also, R_n is equivalent to J'_n, which is the same as MS'_n. Therefore, we only need to further consider R_n and can exclude J'_n, Γ'_n, and MS'_n from the pool. The results in Sect. 5.4 also show that the normalized F-measure (F_n) and Classification Error (ε_n) cannot well capture the uniform effect by K-means. Also, as mentioned in Sect. 5.5, these two measures do not satisfy some math properties in Table 5.12. As a result, we can exclude them. Now, we have five normalized measures: VI_n, VD_n, R_n, FM_n, and Γ_n. In Fig. 5.5, we know that the validation performances of R_n, FM_n, and Γ_n are very similar to each other. Therefore, we only need to consider to use R_n.

From the above study, we believe it is most suitable to use the normalized van Dongen criterion (VD_n), the normalized Variation of Information measure (VI_n), and the normalized Rand statistic (R_n) for K-means clustering validation. VD_n has a simplest computation form, and satisfies some mathematical properties. When different clustering performances are hard to distinguish, however, we may want to use VI_n instead for its high sensitivity. R_n is most complicated in computation, but has a clear statistical meaning and a wide range of value.

Some recent studies on information-theoretic clustering use the normalized Mutual Information (NMI) to evaluate the clustering performance [5, 25]. That is, $NMI = MI(P,C)/\sqrt{H(C)H(P)}$, where the random variables P and C denote the cluster and class sizes, respectively. Apparently, the value of NMI is in the interval: [0,1], and a larger value indicates a better clustering result. It is interesting to note that if we substitute the geometric mean $\sqrt{H(P)H(C)}$ by the arithmetic mean $(H(P)+H(C))/2$ in NMI, NMI reduces to MI_n, which is equivalent to VI_n. From this perspective, NMI is a validation measure similar to VI_n, but tends to give higher scores to clustering results than VI_n due to the smaller geometric mean.

5.6 Concluding Remarks

In this chapter, we illustrated how to select the right external measures for K-means clustering validation. The ability to detect the uniform effect of K-means is highlighted as an important criterion for measure selection. Moreover, we unveil that it is necessary to normalize validation measures before using them, since unnormalized measures may lead to inconsistent or even misleading results, particularly for data with highly imbalanced class distributions. Along this line, we provide normalization solutions to the popular measures by computing their expected or extreme values in two normalization schemes. Furthermore, we explore the correlation between these measures, and present some key properties that may impact their validation performance. We finally summarize the whole procedure for the measure selection, and suggest using the three normalized measures for K-means clustering: VD_n, VI_n, and R_n.

References

1. Ben-Hur, A., Guyon, I.: Detecting stable clusters using principal component analysis. Methods Mol. Biol. **224**(2003), 159–182 (2003)
2. Brun, M., Sima, C., Hua, J., Lowey, J., Carroll, B., Suh, E., Dougherty, E.: Model-based evaluation of clustering validation measures. Pattern Recognit. **40**, 807–824 (2007)
3. Childs, A., Balakrishnan, N.: Some approximations to the multivariate hypergeometric distribution with applications to hypothesis testing. Comput. Stat. Data Anal. **35**(2), 137–154 (2000)
4. Cover, T., Thomas, J.: Elements of Information Theory, 2nd edn. Wiley-Interscience, New York (2006)
5. Dhillon, I., Mallela, S., Modha, D.: Information-theoretic co-clustering. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 89–98 (2003)
6. van Dongen, S.: Performance criteria for graph clustering and markov cluster experiments. Technical report, Amsterdam, The Netherlands (2000)
7. Fowlkes, E., Mallows, C.: A method for comparing two hierarchical clusterings. J. Am. Stat. Assoc **78**, 553–569 (1983)
8. Goodman, L., Kruskal, W.: Measures of association for cross classification. J. Am. Stat. Assoc **49**, 732–764 (1954)
9. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Cluster validity methods: Part i. SIGMOD Rec. **31**(2), 40–45 (2002)
10. Hubert, L.: Nominal scale response agreement as a generalized correlation. Br. J. Math. Stat. Psychol. **30**, 98–103 (1977)
11. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**, 193–218 (1985)
12. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
13. Kendall, M.: Rank Correlation Methods. Hafner Publishing Company, New York (1955)
14. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 16–22 (1999)
15. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
16. Meila, M.: Comparing clusterings by the variation of information. In: Proceedings of the 16th Annual Conference on Computational Learning Theory, pp. 173–187 (2003)
17. Meila, M.: Comparing clusterings–an axiomatic view. In: Proceedings of the 22nd International Conference on Machine learning, pp. 577–584 (2005)
18. Mirkin, B.: Mathematical Classification and Clustering. Kluwer Academic Press, Dordrecht (1996)
19. Rand, W.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**, 846–850 (1971)
20. Rijsbergen, C.: Information Retrieval, 2nd edn. Butterworths, London (1979)
21. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of the KDD Workshop on Text Mining (2000)
22. Strehl, A., Ghosh, J., Mooney, R.: Impact of similarity measures on web-page clustering. In: Proceedings of the AAAI Workshop on AI for Web Search (2000)
23. Wu, J., Xiong, H., Chen, J.: Adapting the right measures for k-means clustering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 877–886. New York, NY, USA (2009)
24. Zhao, Y., Karypis, G.: Criterion functions for document clustering: Experiments and analysis. Mach. Learn. **55**(3), 311–331 (2004)
25. Zhong, S., Ghosh, J.: Generative model-based document clustering: a comparative study. Knowl. Inf. Syst. **8**(3), 374–384 (2005)

Chapter 6
K-means Based Local Decomposition for Rare Class Analysis

6.1 Introduction

Classification provides insight into the data by assigning objects to one of several predefined categories. A longstanding critical challenge for classification is to address the so-called "imbalanced classes" in the data. Specifically, people are interested in predicting rare classes in the data sets with imbalanced class distributions. For example, in the domain of network intrusion detection, the number of malicious network activities is usually very small compared to the number of normal network connections. It is crucial and challenging to build a learning model which has the prediction power to capture future network attacks with low false-positive rates. Indeed, rare class analysis is often of great value and is demanded in many real-world applications, such as the detection of oil spills in satellite radar images [17], the prediction of financial distress in enterprises [33], and the prediction of telecommunication equipment failures [29].

To meet the above challenge, considerable research efforts have been focused on the algorithm-level improvement of the existing classifiers for rare class analysis. There are two promising research directions: The use of re-sampling techniques or the use of cost-sensitive learning techniques [26]. These two methods indeed show encouraging performances in some cases by directly or indirectly adjusting the class sizes to a relatively balanced level. Nevertheless, in this chapter, we reveal that the class imbalance problem is strongly related to the presence of complex concepts (inherent complex structures) in the data. For imbalanced data sets with complex concepts, it is often not sufficient to simply manipulate class sizes. In fact, our experimental results show that adjusting class sizes alone usually can improve the predictive accuracy of rare classes slightly, but at the cost of seriously decreasing the accuracy of large classes. As a result, one way to tackle this problem is to develop a classification method which follows two criteria below:

- *The ability to divide imbalanced classes into relatively balanced classes for classification.*
- *The ability to decompose complex concepts within a class into simple concepts.*

Indeed, this study fills this crucial void by designing a method for classification using local clustering (COG). Specifically, for a data set with an imbalanced class distribution, we perform clustering within each large class and produce sub-classes with relatively balanced sizes. Then, we apply traditional supervised learning algorithms, such as Support Vector Machines (SVMs) [1, 27], for classification. Since the clustering is conducted independently within each class rather than across the entire data set, we call it *local clustering*, which is the essential part of the COG method. By exploiting local clustering within large classes, we can decompose the complex concepts (e.g., linearly inseparable concepts difficult for linear classifiers) into relatively simple ones (e.g., linearly separable concepts). Another effect of local clustering is to produce subclasses with relatively uniform sizes, by using the well-known K-means algorithm [20] as the clustering tool. In addition, for data sets with highly skewed class distributions, we further integrate the over-sampling technique into the COG scheme and propose the COG with the over-sampling method (COG-OS).

The merit of COG lies in three aspects. First, COG has the ability to divide imbalanced classes into relatively balanced and small sub-classes, and thus provide the opportunities in exploiting traditional classification algorithms for better predicting rare classes. Second, similar to the re-sampling schemes, COG is not a "bottom-level" algorithm but provides a general framework which can incorporate various existing classifiers. Finally, COG is especially effective on improving the performance of linear classifiers. This is noteworthy, since linear classifiers have shown their unique advantages, such as simplicity and understandability, higher executive efficiency, less parameters, and less generalization errors, in many real-world applications [8, 24].

We have conducted extensive experiments on a number of real-world data sets. The experimental results show that, for data sets with imbalanced classes, COG and COG-OS show much better performances in predicting rare classes than two popular re-sampling schemes as well as two state-of-the-art rule induction classifiers without compromising the prediction accuracies of large classes. In addition, for data sets with balanced classes, we show that COG can also improve the performance of traditional linear classifiers, such as SVMs, by decomposing the linearly inseparable concepts into linearly separable ones. To further demonstrate the effectiveness of COG in real-world applications, we have applied COG and COG-OS for two real-world applications: the credit card fraud detection and the network intrusion detection. Finally, we discuss the limitations of COG and COG-OS due to the inconsistent uses of the features in unsupervised and supervised learning, and illustrate the effect of feature selection on alleviating this problem.

The remainder of this chapter is organized as follows. In Sect. 6.2, we describe the preliminaries and define the problem studied in this chapter. Section 6.3 introduces the local clustering technique and presents a discussion of its properties. Then, in Sect. 6.4, we give the COG and COG-OS schemes for rare class analysis. Section 6.5 shows experimental results and real-world case studies. Finally, we present related work in Sect. 6.6, and draw conclusions in Sect. 6.7.

6.2 Preliminaries and Problem Definition

In this section, we first review the main issues related to rare class analysis. Then, we present the problem definition.

6.2.1 Rare Class Analysis

Rare events are often of special interests. Indeed, people often want to uncover subtle patterns that may be hidden in massive amounts of data [28]. In the literature, the problem of rare class analysis has been addressed from the following two perspectives.

The Class Imbalance Perspective. From this viewpoint, it is the skewed class distribution that brings the challenge for traditional classifiers. A natural solution is to re-balance the numbers of training instances in different classes. The re-sampling schemes meet this requirement nicely. Specifically, over-sampling replicates the rare instances so as to match the scale of the normal class. In contrast, under-sampling cuts down the size of normal classes by performing random sampling. In addition, a combined use of over-sampling and under-sampling techniques may be more beneficial in some cases.

The Cost Sensitive Perspective. From this viewpoint, the costs of misclassifications are different; that is, assigning rare instances to the normal class should be penalized more heavily. One way to incorporate the cost information is to use the cost matrix explicitly during model learning. For example, in SVMs, we can set different penalty parameters for misclassifying rare and normal instances. Another way to incorporate the cost information is through the algorithm-independent schemes, such as boosting. For instance, AdaCost [10], one of the variants of AdaBoost [11], uses the additional cost adjustment function in the weight updating rule for the purpose of rare class analysis.

While the above mentioned methods can often improve the prediction accuracy of rare classes, they often lead to large classification errors for normal classes. Let us illustrate this by some examples shown in Fig. 6.1. In the figure, the solid lines denoted as "T" indicate the true borders, and the dashed lines denoted as "F" are the classification borders learned by applying the over-sampling or cost-sensitive schemes. As can be seen, these two schemes tend to assign the normal instances to the rare class. This is due to the fact that we usually do over-sampling for or assign higher costs to all the rare instances so as to match the scale of the normal class. However, the normal instances can be sparsely distributed so that many of them are far away from the true border, and thus have little impact during the border learning process. As a result, rare instances tend to have more power to decide the border. This power may "shift" the border into the normal class, and lead to a large misclassification rate for the normal class. Furthermore, if the rare class consists of some small disjoints, as shown in Fig. 6.1c, the re-sampling and cost-sensitive

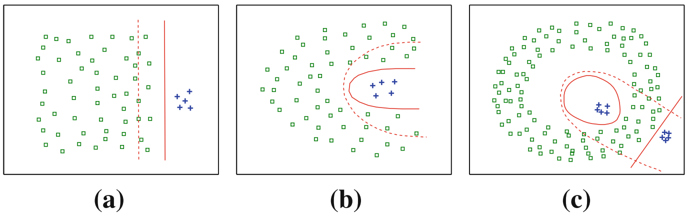


Fig. 6.1 Illustrative examples. **a** Example 1. **b** Example 2. **c** Example 3. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

methods may encounter even more problems, since more normal instances can be assigned to the rare class.

Another widely used approach to handle rare classes is to be focused on learning rare classes and produce classification rules that only predict rare classes. RIPPER [3] is such an algorithm. It utilizes a separate-and-conquer approach to build rules to cover previously uncovered training instances. It generates rules for each class from the most rare class to the second most common class (the most common class is the default class). Each rule is grown by adding conditions until no negative instances are covered. Thus, for a two-class data set, RIPPER will only learn rules for the rare class, and assign the negative instances to the normal class. Although this type of learning systems has high efficiency and good performances in many applications, they are easier to get overfitting, which results in a relatively poor generalization ability. Also, when the number of rare events is extremely small, there may be no enough training instances to induce the rules for RIPPER.

To sum up, further investigation is still needed to improve the classification performance on rare classes and meanwhile avoid the significant negative impact on the classification performance on normal classes.

6.2.2 Problem Definition

For traditional classifiers, all the instances from a class are used to represent a specific concept and are used as a whole for building the classification model. However, it is quite normal to see that the instances from a large class can be divided into different sub-concepts. In this study, we define three types of concepts. First, the linearly separable concepts consist of linearly separable sub-concepts. Second, the nonlinearly separable concepts include sub-concepts which are not linearly separable, but nonlinearly separable. Finally, the complex concept includes sub-concepts which may not be easily separated by either linear or nonlinear methods. For instance, the three sub-figures in Fig. 6.1 well illustrate the linearly separable, non-linearly separable, and complex concepts, respectively.

Problem Definition. Given data sets with rare classes, we are going to develop a method for Classification using lOcal clusterinG (COG). Specifically, we will exploit linear-classifiers for rare class analysis even if data sets include instances with linearly inseparable or complex concepts. The key is the use of "local clustering", which will be detailed in the following section. Indeed, local clustering can help to decompose instances with linearly inseparable or complex concepts in a class into several linearly separable sub-groups. Note that the reason that we target on linear classifiers is to leverage their distinct merits, such as the high efficiency, the good interpretability, and the high generalizability [24].

6.3 Local Clustering

In this section, we introduce the Local Clustering (LC) technique, and explore its unique properties for rare class analysis.

6.3.1 The Local Clustering Scheme

People typically do clustering on the entire data to find the inherent data structure. However, due to the existence of complex concepts in real-world data sets, clustering algorithms usually produce clusters with a mix of data instances from different classes. Therefore, clustering on the entire data cannot help to decompose complex concepts in one class into several linearly separable simple concepts. To meet this challenge, a natural way is to do clustering within each class to divide complex structures/concepts in a class into several simple sub-strutures/sub-concepts, which are usually much easier to be separated by linear classifiers. This is the basic idea of the use of local clustering for classification.

Local clustering is a partitional strategy which exploits clusterings inside each class rather than across the entire data set. By using the external information of instance labels, local clustering can produce simple structures for the subsequent classification tasks. Figure 6.2 illustrates the local clusterings on the three sample data sets in Fig. 6.1. For sample data sets 1 and 2, we only do local clusterings on the normal classes. But for the presence of some small disjoints in the rare class, to do local clustering on both the normal and rare classes are more beneficial, as shown in Fig. 6.2c.

6.3.2 Properties of Local Clustering for Classification

Here, we explore some properties of local clustering for classification. Specifically, we demonstrate the "balance effect" and the "decomposition effect" of local clustering.

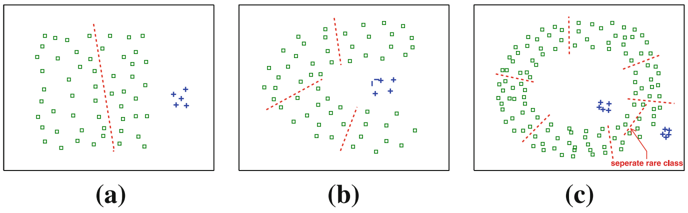


Fig. 6.2 Illustrations of local clusterings. **a** Example 1. **b** Example 2. **c** Example 3. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

The **balance effect** refers to the fact that local clustering can re-balance the scales of normal and rare classes. For instance, local cluster can help to divide the instances of a normal class into several sub-groups which have much smaller sizes compared to the original class. If we label those sub-groups with pseudo-class labels, we can have a more balanced multi-class training data set. This property is very important for rare class analysis, since it can help to alleviate the skewed class distributions in the training data. Also, local clustering can help to achieve this without using oversampling and undersampling techniques and can keep the integrity of the original data for the learning process.

The **decomposition effect** refers to the fact that local clustering can help to decompose linearly inseparable or complex concepts in a large class into several simple structures/concepts, which are much easier to be separated by linear classifiers. Indeed, this is crucial for the successful use of linear classifiers for rare class analysis. For example, in Fig. 6.1b, the true border for the normal and rare classes (denoted as “T”) is non-linear. However, local clustering can help to divide the normal class into four groups, which can be labeled as four new pseudo-classes. Now, the four new classes and the rare class are linearly separable, as can be seen in Fig. 6.2b.

In summary, the balance and decomposition effects by local clustering provide unique opportunities for the successful use of linear classifiers for rare class analysis.

6.4 COG for Rare Class Analysis

In this section, we introduce COG (Classification using lOcal clusterinG) and COG-OS (COG with Over-Sampling) algorithms.

6.4.1 COG and COG-OS

In a nutshell, COG provides a general framework which can incorporate various linear classifiers and improve their classification performances on data sets with complex concepts as well as imbalanced class distributions. As to COG-OS, it is

Fig. 6.3 The COG algorithm

```
COG (Classification using lOcal clusterinG)

Input:    $\mathscr{D}$: a training data set.
          $\mathscr{T}$: a test data set.
          $\mathbb{L}$: a linear classifier, such as SVMs.
          $K$: a vector specifies the number of local clusters
              for each class.
          $r$: a vector specifies the over-sampling rate for
              each class. (for COG-OS only).

Output:   $\mathbb{M}$: the classification model built on $\mathscr{D}$.
          $p$: a vector indicates the predicted labels for $\mathscr{T}$.

Procedure:
Phase I: local clustering
 1.       for class $i = 1$ to $c$   // "c" represents #classes
 2.           clusterLabel($i$) = Clustering($\mathscr{D}(i)$, $K(i)$);
 3.           $\mathscr{D}(i)^*$ = changeLabel($\mathscr{D}(i)$, clusterLabel($i$));
 4.       end for
Phase II: over-sampling (for COG-OS only)
 5.       for class $j = 1$ to $c$
 6.           $\mathscr{D}(j)^{**}$ = replicate($\mathscr{D}(j)^*$, $r(j)$);
 7.       end for
 8.       $\mathscr{D}^{**} = \bigcup_{j=1}^{c}(\mathscr{D}(j)^{**})$;
Phase III: training
 9.       $\mathbb{M}$ = train($\mathscr{D}^{**}$, $\mathbb{L}$);
Phase IV: predicting
 10.      $p'$ = predict($\mathscr{T}$, $\mathbb{M}$);
 11.      $p$ = convertLabel($p'$);
```

an extended version of COG, which integrates the over-sampling technique into the COG scheme for the purpose of better predicting the rare classes in data sets with extremely imbalanced class distributions.

Figure 6.3 shows the pseudo-codes of COG including four phases. In Phase I, we employ local clustering on class i ($i = 1, \ldots, c$) according to the user specified cluster number $\boldsymbol{K}(i)$, and change the instance labels of class i with the cluster labels, thus form a multi-class data set with $\sum_{i=1}^{c} \boldsymbol{K}(i)$ subclasses. Typically, we suggest using K-means [20] as the default local clustering tool. This is due to the following reasons: (1) K-means tends to partition the data into clusters in relatively uniform sizes (as indicated in Chap. 2), which is of great help to show the balance effect of local clustering; and (2) the efficiency of K-means is approximately linear in the number of training instances [26], which is the best among various popular clustering methods. Moreover, we recommend not using a large $\boldsymbol{K}(i)$ for each class i, for the purpose of improving the modeling efficiency and reducing the generalization errors of the learning model. In most cases, we let $\boldsymbol{K}(i) = 4$. Finally, we usually do local clustering on the normal classes. However, as indicated in Fig. 6.2c, when the rare class consists of some small disjoints, we can also locally cluster the rare instances.

This happens when we achieve a high false-positive rate by only employing local clustering on normal classes.

Phase II is optional and only for COG-OS. In this phase, we replicate $\boldsymbol{r}(j)$ times the instances of class j ($j = 1, \ldots, c$), to form a more balanced data set. Typically, $\boldsymbol{r}(j) > 1$ when class j is a rare class. Furthermore, if the rare class has been partitioned into small disjoints, we may set different sampling ratios for them.

Phase III is to build the model on the modified data set using a user specified linear classifier such as SVMs and BMR.[1] In detail, we only learn linear models to separate each pair of sub-classes derived from DIFFERENT parent-classes. In other words, we do not discriminate the instances from the same parent-class. Therefore, exactly $\sum_{1 \le i < j \le c} \boldsymbol{K}(i)\boldsymbol{K}(j)$ linear hyper-planes will be learned in this phase.

Phase IV is simply for prediction. However, two points should be further addressed. First, we use the comparison rather than voting mechanism to determine the label of a new instance. Second, the assigned pseudo sub-class label for each new instance must be converted into the label of its parent-class.

6.4.2 An Illustration of COG

Here, we use a synthetic data set with complex concepts to illustrate the process of COG. The scales of three classes are 133, 60 and 165, respectively, as shown in sub-figure "Original Data" of Fig. 6.4. The classifier is LIBSVM[2] with the linear kernel.

First, we build the classification model by simply applying SVMs on the original data set, and the result is shown in the "'Pure' SVMs" subplot. In this subplot, the solid line represents the maximal margin hyperplane (MMH) learned by SVMs algorithm. One interesting observation is that the instances of the small class, i.e., class 2, have totally "disappeared"; that is, they are all assigned to either class 1 or class 3 according to the only one MMH. This indicates that the complex concept indeed hinders the rare class analysis.

Instead, we employ COG. First, we apply local clustering on class 1 and 3, respectively, given the cluster number is two. The clustering results can be seen in Subplot "COG: Phase I". In other words, class 1 and class 3 are divided into two sub-classes respectively by K-means. Thus, we obtain a modified data set with 5 relatively balanced and linearly separable classes. Next, we apply SVMs on this five-class data set and get results shown in Subplot "COG: Phase III". As can be seen, eight MMHs appear in the model, which enables the model to identify the instances of class 2. Finally, for each instance, we convert its predicted label of some sub-class into the label of the parent-class, as indicated by the "COG: Phase IV" subplot. Therefore, by applying COG, we build up a more accurate model which can identify the instances from rare classes at the presence of complex concepts.

[1] http://www.stat.rutgers.edu/~madigan/BMR/

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

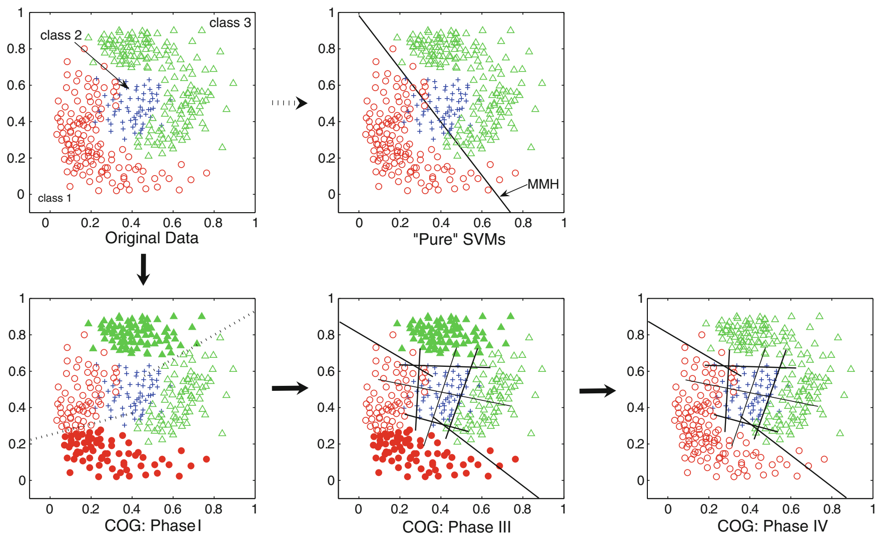


Fig. 6.4 Illustration of the COG procedure. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

6.4.3 Computational Complexity Issues

In this section, we analyze the computational complexity of COG. By examining the COG procedure as shown in Fig. 6.3, we know that there are two major computational components. One is local clustering, and the other is model learning.

For local clustering, we take K-means as an example to illustrate the use of time and space. Since we only store the data instances and centroids, the space requirement for K-means is $O((p + K)n)$, where p and n are the numbers of attributes and instances, and K is the number of clusters. Therefore, for the whole local clustering procedure, the space complexity is $O(\max_i(p + K_i)n_i)$, where i denotes class i, $i \in \{1, \cdots, c\}$. The time requirement for K-means is $O(IKpn)$, where I is the number of iterations required for convergence. Therefore, the time complexity for local clustering is $O(p \sum_i I_i K_i n_i)$. Since I_i is often small and can usually be safely bounded, and we often set a small K_i for each class, the efficiency of local clustering is approximately linear in the number of all data instances: $n = \sum_i n_i$.

For model learning, we take SVMs as an example to illustrate the use of time and space. Specifically, we would like to compare the computational complexities of SVMs with and without using local clustering. Given a training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $y_i \in \{1, -1\}$, SVMs with linear kernel solves the following primal problem:

$$\begin{aligned} &\min_{\mathrm{w},b,\xi} \ \tfrac{1}{2}\|\mathrm{w}\|^2 + C\sum_i \xi_i, \\ &\text{s.t.} \qquad y_i(\mathrm{w}^T\mathrm{x}_i + b) + \xi_i \geq 1, \\ &\qquad\qquad \xi_i \geq 0, \ \ i = 1, \ldots, n, \end{aligned} \tag{6.1}$$

where C is the penalty of misclassification, and ξ_i is the vector of slack variables. The dual problem of Eq. (6.1) is as follows.

$$\begin{aligned} &\min_{\alpha} \ \tfrac{1}{2}\sum_{i=1}^n \sum_{j=1}^n y_i y_j \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j \mathrm{x}_i^T \mathrm{x}_j - \sum_{i=1}^n \alpha_i, \\ &\text{s.t.} \quad \sum_{i=1}^n y_i\alpha_i = 0, \\ &\qquad 0 \leq \alpha_i \leq C, \ i = 1, \ldots, n. \end{aligned} \tag{6.2}$$

This is a typical convex-quadratic-programming problem. Existing methods, such as the Newton-PCG algorithm and the Quadratic Interior Point method, can be used to solve this problem. In general, these methods require to store the $n \times n$ kernel matrix. Therefore, the space requirement for SVMs is roughly $O(n^2 + np)$. However, once employing local clustering, we can divide the original large-scale problem into some smaller scale problems. For instance, let n_{is} denote the number of instances from cluster s produced by local clustering on class i. Since we only learn a hyperplane for each pair of sub-classes from different parent-classes, the space requirement for SVMs in COG is only $O(\max_{1\leq i<j\leq c,\ 1\leq s\leq K_i,\ 1\leq t\leq K_j}(n_{is} + n_{jt})^2 + np)$, which is much less than $O(n^2 + np)$. Therefore, local clustering can help to reduce the space requirement for SVMs.

The time requirement for SVMs in COG is often much less than the pure SVMs (without using local clustering) in real-world applications, because local clustering helps to decompose complex structures/concepts into smaller and simpler concepts which are much easier to learn by SVMs. For SVMs, we need to compute the $n \times n$ kernel matrix for data with n instances. After employing local clustering, the time complexity for learning SVMs is $O\left(\sum_{1\leq i<j\leq c}\sum_{1\leq s\leq K_i}\sum_{1\leq t\leq K_j}(n_{is} + n_{jt})^2\right)$, which is less than $O\left(\left(\sum_{i=1}^c \sum_{s=1}^{K_i} n_{is}\right)^2\right)$, i.e., $O(n^2)$ for learning SVMs without local clustering. Also, for large-scale data sets, people usually employ iterative methods to approach the true SVMs gradually. And the convergence speed is strongly related to the number of support vectors—less support vectors often lead to a faster convergence. Since local clustering can decompose the non-linearly separable or even the complex concepts into some linearly separable concepts, it can help to reduce the number of misclassified instances. This results in a smaller number of support vectors eventually.

In summary, while COG introduces additional cost for local clustering, COG can greatly reduce the time and space uses for SVMs learning. In real-world applications, the time used for local clustering is usually much less than the time used for SVMs learning, therefore the cost for local clustering can usually be ignored. As a result, COG can significantly improve the computational performances of the underlying

Table 6.1 Some characteristics of experimental data sets

Dataset	Source	#Objects	#Features	#Classes	MinClassZize	MaxClassSize	CV
UCI data sets							
breast-w[a]	UCI	683	9	2	239	444	0.424
pima-diabetes	UCI	768	8	2	268	500	0.427
letter	UCI	20000	16	26	734	813	0.030
optdigits[b]	UCI	3823/1797	64	10	376/174	389/183	0.014/0.015
page-blocks	UCI	5473	10	5	28	4913	1.953
pendigits[b]	UCI	7494/3498	16	10	719/335	780/364	0.042/0.042
satimage[b]	UCI	4435/2000	36	6	415/211	1072/470	0.425/0.368
vowel[b]	UCI	528/462	10	11	48/42	48/42	0.000/0.000
Document data sets							
k1b	WebACE	2340	21839	6	60	1389	1.316
la12	TREC	6279	31472	6	521	1848	0.503
LIBSVM data sets							
fourclass	LIBSVM	862	2	2	307	555	0.407
german.numer	LIBSVM	1000	24	2	300	700	0.567
splice	LIBSVM	1000	60	2	483	517	0.048
SVMguide1	LIBSVM	3089	4	2	1089	2000	0.417

Notes The numbers before and after "/" are for the training and test sets, respectively [a]16 instances with missing data have been deleted [b]These data sets contain test sets provided by the sources

learning models, such as SVMs. In the following section, we will also provide an empirical study to validate the above analysis.

6.5 Experimental Results

In this section, we present experimental results to validate the performance of the COG and COG-OS methods.

6.5.1 Experimental Setup

We first introduce the experimental setup, including the data information and the classifiers.

Experimental Data. We used a number of benchmark data sets that were obtained from different application domains. Some characteristics of these data sets are shown in Table 6.1. In the table, the Coefficient of Variation (CV) [5] shows the dispersion of the distribution of the class sizes for each data set. In general, the larger the CV value is, the greater the variability in the data.

UCI Data Sets. In the experiments, we used eight well-known benchmark data sets from the UCI Repository.[3] Among them two data sets, `breast-w` and `pima-diabetes`, are binary data sets from the medical domain. The `breast-w` data set contains two types of results from real-world breast cancer diagnosis, and the `pima-diabetes` data set is about the information of whether the patient shows signs of diabetes according to the WHO criteria. The rest six data sets are frequently used in the pattern recognition community. `letter`, `optdigits` and `pendigits` are data sets containing the information of handwritings; that is, `letter` has the letter information from A to Z, and the other two have the number information from 0 to 9. The `satimage` data set contains the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image. The `page-blocks` data set contains the information of five types of blocks from a document page layout. And the last data set `vowel` was designed for the task of speaker independent recognition of the eleven steady state vowels of British English.

Document Data Sets. We also used high-dimensional document data sets in our experiments. The data set `k1b` was from the WebACE project [12]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo!. The `la12` data set was obtained from articles of the Los Angeles Times in TREC-5.[4] The categories correspond to the *desk* of the paper that each article appeared and include documents from the entertainment, financial, foreign, metro, national, and sports desks. For these two document data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [23].

LIBSVM Data Sets. Finally, four benchmark binary data sets: `fourclass`, `german.numer`, `splice`, and `SVMguide1`, were selected from the LIBSVM repository.[5]

Note that for any data set without a given test set, we did random, stratified sampling on it and had 70 % samples as the training set and the rest as the test set.

Experimental Tools. In the experiment, we used four types of classifiers: Support vector machines (SVMs), Bayesian logistic regression, decision trees, and classification rules. Their corresponding implementations are LIBSVM, BMR, C4.5,[6] and RIPPER [3], respectively. In all the experiments, default settings were used except that the kernel of LIBSVM was set to linearity. LIBSVM and BMR are linear classifiers, and C4.5 and RIPPER are non-linear classifiers.

We applied K-means for local clustering in our COG scheme. K-means is a widely used clustering method which tends to produce clusters with relatively uniform sizes. During the K-means clustering, for data sets with relatively few dimensions, squared Euclidean distance was used as the proximity measure. However, for high-dimensional data sets, the cosine similarity was used instead. This is due to the fact that the Euclidean notion of proximity is not very meaningful for high-dimensional data sets, such as document data sets.

[3] http://www.ics.uci.edu/~mlearn/MLRepository.html

[4] http://trec.nist.gov

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[6] http://www.rulequest.com/Personal/

Table 6.2 Sampled data sets

Data	Class	Sampling ratio	#Instances	CV
breast-w	1/2	0.20/1.00	48/444	1.14
pima-diabetes	1/2	0.20/1.00	54/500	1.14
fourclass	1/2	0.20/1.00	62/555	1.13
german.numer	1/2	0.20/1.00	60/700	1.19
splice	1/2	0.10/1.00	49/517	1.17
SVMguide1	1/2	0.05/1.00	53/2000	1.34

Finally, in addition to the COG and COG-OS schemes, we also used the under-sampling (US) and over-sampling (OS) techniques. The default classifier used in the experiment is SVMs. If some other classifier is used instead, we will point it out explicitly. For instance, we use COG(BMR) to denote "BMR in the COG scheme".

6.5.2 COG and COG-OS on Imbalanced Data Sets

In this section, we show how COG and COG-OS can improve the performance of linear classifiers on imbalanced data sets. Specifically, we employ COG-OS on binary data sets with rare classes, and COG on multi-class data sets with imbalanced classes.

For each binary data set, we did K-means clustering on the normal class, and set the cluster number consistently to four. Then we did over-sampling on the rare class, and made the size approximately to be one fourth of the size of the normal class. In this way, we can have much more balanced data sets. For each multi-class data set, however, we set different local-clustering numbers for different normal classes according to the sizes of normal classes.

It is noteworthy that the binary data sets with rare classes were generated by sampling various benchmark data sets. Specifically, we did random sampling on the small class to turn it into a rare class, then combined it with the original large class to form a sample data set. Detailed information of the samples can be found in Table 6.2. We repeated sampling ten times for each data set such that we can have the average classification accuracy. Finally, since SVMs shows best classification performances in many cases [4], we used it as the base classifier for all the experiments in this section.

Results by COG-OS on Two-class Data Sets. Table 6.3 shows the performances of COG-OS and pure SVMs on six two-class data sets. As can be seen, for three data sets `pima-diabetes`, `fourclass` and `german.numer`, pure SVMs assigned all the instances to the normal classes. This indicates that pure SVMs has no prediction power on rare classes for these three data sets. In contrast, COG-OS successfully identified more than 30 % instances of the rare classes. Indeed, the F-measure values of the rare classes by COG-OS are consistently higher than the F-measure values

Table 6.3 Classification results of sampled data sets by COG-OS

Data Set	Method	Class	#Clusters	#Repetitions	Recall	Precision	F-measure
breast-w	SVMs	1	N/A	N/A	0.871	0.878	*0.872*
		2	N/A	N/A	0.986	0.987	0.985
	COG-OS	1	1	3	0.907	0.850	*0.873*
		2	4	1	0.982	0.990	0.986
pima-diabetes	SVMs	1	N/A	N/A	0.000	#DIV/0!	N/A
		2	N/A	N/A	1.000	0.904	0.949
	COG-OS	1	1	2	0.344	0.428	*0.373*
		2	4	1	0.949	0.931	0.940
fourclass	SVMs	1	N/A	N/A	0.000	#DIV/0!	N/A
		2	N/A	N/A	1.000	0.902	0.948
	COG-OS	1	1	2	0.606	0.699	*0.646*
		2	4	1	0.971	0.958	0.964
german.numer	SVMs	1	N/A	N/A	0.000	#DIV/0!	N/A
		2	N/A	N/A	1.000	0.921	0.959
	COG-OS	1	1	3	0.317	0.239	*0.270*
		2	4	1	0.915	0.940	0.927
splice	SVMs	1	N/A	N/A	0.314	0.289	*0.298*
		2	N/A	N/A	0.929	0.938	0.933
	COG-OS	1	1	3	0.493	0.270	*0.344*
		2	4	1	0.874	0.950	0.910
SVMguide1	SVMs	1	N/A	N/A	0.069	0.261	*0.104*
		2	N/A	N/A	1.000	0.976	0.988
	COG-OS	1	1	9	0.913	0.365	*0.518*
		2	4	1	0.956	0.998	0.976

Notes 1. For SVMs: -t 0 2.“#clusters” indicates the cluster numbers for K-means in local clustering 3.“#repetitions” indicates the over-sampling ratio 4.“N/A”: not applicable; “#DIV/0!”: divided by zero

produced by pure SVMs for all six data sets, as indicated in Table 6.3. For instance, for data sets `SVMguide1` and `fourclass`, COG-OS leads to the increase of the F-measure values by more than 0.4.

Another observation is that, for COG-OS, the increase of the F-measure value of the rare class is **NOT** at the significant cost of the prediction accuracy of the normal class. For example, for data sets `breast-w` and `fourclass`, the F-measure values of both the normal and rare classes by COG-OS are higher than the F-measure values produced by pure SVMs. Also, for the rest four data sets, the F-measure values of the normal classes by COG-OS are just slightly smaller. This is acceptable since the rare class is usually the major concern in many real-world applications.

In addition, we also investigate how the F-measure value changes as the increase of the sampling ratio on the small class. As an example on the `SVMguide1` data set, Table 6.4 shows the information of various samples as the increase of the size of the rare class. Figure 6.5 shows the classification results on these samples. Please note that for each sampling ratio, we repeated sampling ten times and therefore got ten results for each sample, as indicated by the box plots in Fig. 6.5. As can be seen,

Table 6.4 Information of `SVMguide1` samples

Sample ID	1	2	3	4	5
Sampling ratio	0.03	0.05	0.07	0.09	0.11
Size of rare class	33	55	77	99	120
#clusters for normal class	4	4	4	4	4
#repetitions for rare class	15	9	6	5	4

Note The size of the normal class is 2000

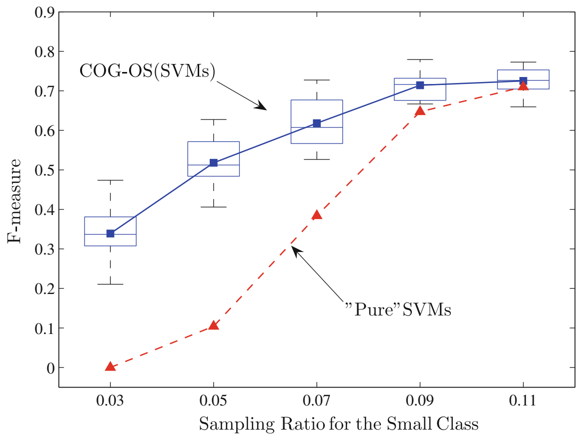


Fig. 6.5 The effect of rare class sizes on COG-OS. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

the F-measure values by COG-OS are consistently higher than the ones produced by pure SVMs, no matter what the sampling ratio is. Moreover, COG-OS performs much better than pure SVMs when the rare class size is relatively small. However, as the increase of the size of the rare class, the performance gap is narrowed.

Results by COG on Multi-class Data Sets. In addition to the two-class data sets, we also used some multi-class data sets with imbalanced classes to validate the performance of the COG method. For these multi-class data sets, we simply used the COG scheme. Since the Euclidean distance is not very meaningful for the high-dimensional document data sets `k1b` and `la12`, in the local clustering phase, we used the CLUTO implementation of K-means[7] on these two data sets with the cosine similarity as the proximity measure. Table 6.5 shows the classification results by pure SVMs and COG. As can be seen, for data set `k1b`, the F-measure value for every class using COG is higher than that produced by pure SVMs. Meanwhile, the results on the `page-blocks` and `la12` data sets show a similar trend as `k1b`. In summary, COG indeed can improve the prediction performances on rare classes, and this improvement is achieved without a big loss of the prediction power on normal classes.

[7] http://glaros.dtc.umn.edu/gkhome/views/cluto

Table 6.5 Classification results of COG on multi-class data sets

		Pure SVMs				COG			
Class	#Instances	#Clusters	Recall	Precision	F-measure	#Clusters	Recall	Precision	F-measure
`page-blocks` ($CV = 1.953$)									
1	4913	N/A	0.990	0.971	0.980	10	0.993	0.977	0.985
2	329	N/A	0.780	0.839	0.808	2	0.880	0.898	0.889
3	28	N/A	0.556	1.000	0.714	1	0.556	0.833	0.667
4	88	N/A	0.815	0.917	0.863	1	0.815	0.917	0.863
5	115	N/A	0.514	0.900	0.655	1	0.543	0.950	0.691
Total	5473	N/A	0.962	0.962	0.962	N/A	0.971	0.971	0.971
`k1b` (CV=1.316)									
1	494	N/A	0.968	1.000	0.984	3	0.974	1.000	0.987
2	1389	N/A	0.990	0.963	0.976	9	0.993	0.981	0.987
3	142	N/A	0.979	0.939	0.958	1	1.000	0.940	0.969
4	114	N/A	0.829	0.967	0.892	1	0.886	0.969	0.925
5	60	N/A	0.810	1.000	0.895	1	0.905	1.000	0.950
6	141	N/A	0.955	0.955	0.955	1	1.000	0.978	0.989
Total	2340	N/A	0.969	0.969	0.969	N/A	0.982	0.982	0.982
`la12` (CV=0.503)									
1	1848	N/A	0.921	0.561	0.697	2	0.886	0.572	0.695
2	1497	N/A	0.757	0.912	0.827	2	0.708	0.922	0.801
3	1042	N/A	0.663	0.759	0.707	1	0.721	0.728	0.725
4	729	N/A	0.465	0.896	0.612	1	0.564	0.866	0.683
5	642	N/A	0.672	0.794	0.728	1	0.687	0.793	0.736
6	521	N/A	0.329	0.917	0.485	1	0.353	0.881	0.504
Total	6279	N/A	0.709	0.709	0.709	N/A	0.713	0.713	0.713

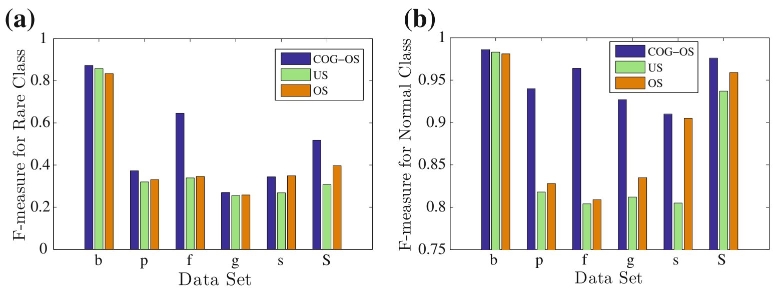


Fig. 6.6 COG-OS versus re-sampling. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

6.5.3 COG-OS Versus Re-Sampling Schemes

In previous sections, we mentioned that re-sampling is a widely used technique to improve the classification performance on imbalanced data sets. Here, we compare the performances of COG-OS with two re-sampling strategies: under-sampling and over-sampling [26, 21].

In this experiment, we also used the six sampled data sets in Table 6.2. We set the sampling ratio for under-sampling or over-sampling carefully so that the modified size of the rare class is approximately the same as the normal class, and the classifier used here is also SVMs. Figure 6.6 shows the results. One observation is that, for all data sets, COG-OS performs the best for the rare classes, except for one data set: `splice`, on which COG-OS and over-sampling show comparable results. Another observation is that COG-OS achieved better performances on both rare classes and normal classes.

In summary, compared to the two widely used re-sampling schemes, COG-OS shows appealing performances on rare class analysis, yet keeps much better performances on normal classes.

6.5.4 COG-OS for Network Intrusion Detection

Here, we demonstrate an application of COG-OS for network intrusion detection. For this experiment, we used a real-world network intrusion data set, which is provided as part of the KDD-CUP-99 classifier learning contest, and now is a benchmark data set in the UCI KDD Archive.[8]

The KDD Cup Data Set. This data set was collected by monitoring a real-life military computer network that was intentionally peppered with various attacks that hackers would use to break in. Original training set has about 5 million records

[8] http://kdd.ics.uci.edu/

belonging to 22 subclasses and 4 classes of attacks, i.e., DoS, Probe, R2l and U2R, and still one normal class. In this experiment, we used the 10 % sample of this original set which is also supplied as part of the KDD CUP contest. We present results for two rare classes: `Probe` and `R2l`, whose populations in the 10 % sample training set are 0.83 % and 0.23 %, respectively. The provided test set has some new subclasses that are not present in the training data, so we deleted the instances of new subclasses, and the percentages of `Probe` and `R2l` in the test set are 0.81 % and 2.05 %, respectively. Table 6.6 shows the detailed information of these data sets. Note that we obtained the `probe_binary` data set by making the `probe` class as the rare class, and the rest four classes as one large class. The other data set, i.e., `r2l_binary`, was prepared in a similar way.

The Benchmark Classifiers. In this experiment, we applied four classifiers: COG-OS(SVMs), pure SVMs, RIPPER [3], and PNrule [14]. For COG-OS, the cluster number for the large class is four, and over-sampling ratios for the rare classes of `probe_binary` and `r2l_binary` are 30 and 120, respectively. For SVMs, we set the parameters as: -t 0. Ripper and PNrule are two rule induction classifiers. RIPPER builds rules first for the smallest class and will not build rules for the largest class. Hence, one might expect that RIPPER can provide a good performance on the rare class. As to PNrule, it consists of positive rules (P-rules) that predict presence of the class, and negative rules (N-rules) that predict absence of the class. It is the existence of N-rules that can ease the two problems induced by the rare class: splintered false positives and error-prone small disjuncts. These two classifiers have shown appealing performances on classifying the modified binary data sets in Table 6.6, and the PNrule classifier even shows superior performance [14]. To our best knowledge, we used the same source data as Joshi et al., and the pre-processing procedure for the modified data sets is also very similar to the one used by them. Therefore, we simply adopted the results of PNrule in [14] for our study.

The Results. Table 6.7 shows the classification results by various methods on the `probe_binary` data set. As can be seen, COG-OS performs much better than pure SVMs and RIPPER on predicting the rare class as well as the normal class, while PNrule shows slightly higher F-measure on the rare class. For data set `r2l_binary`, however, COG-OS shows the best performance among all classifiers. As indicated in Table 6.7, the F-measure value of the rare class by COG-OS is 0.496, far more higher than the ones produced by the rest classifiers. Meanwhile, the predictive accuracy of the large class by COG-OS is also higher than that of pure SVMs and RIPPER. This real-world application nicely illustrates the effectiveness of COG-OS—the combination of local clustering and over-sampling schemes. We believe that COG-OS is a prospective solution to the difficult classification problem induced by complex concepts and imbalanced class distributions.

We also observed the training efficiency of COG-OS, where the experimental platform is Windows XP with an Intel Core2 1.86 GHz cpu and 4 GB memory. As can be seen in Fig. 6.7, for the `probe_binary` data set, the training time for COG-OS is 173 s, which is far less than the training time for pure SVMs: 889 s. Note that the scale of the training data set for COG-OS is 613124 after over-sampling, which is a much larger number than the scale of the training data set for pure SVMs:

Table 6.6 Information of data sets

Dataset	Source	#Objects	#Features	#Classes	MinClassZize	MaxClassSize	CV
kddcup99data	UCI KDD	494021/292300	41	5	52/39	391458/223298	1.708/1.634
probe_binary[a]	UCI KDD	494021/292300	41	2	4107/2377	489914/289923	1.391/1.391
r2l_binary[a]	UCI KDD	494021/292300	41	2	1126/5993	492895/286307	1.408/1.356

Note we deleted 18729 instances from the test set, since the labels of these instances are not present in the training set. [a]Modified binary data sets from `kddcup99data`

Table 6.7 Results on modified data sets

probe_binary	SVMs	RIPPER	PNrule	*COG-OS*
rare class	0.806	0.798	0.884	*0.881*
huge class	0.998	0.998	N/A	*0.999*
total	0.996	0.996	N/A	*0.998*
r2l_binary	SVMs	RIPPER	PNrule	*COG-OS*
rare class	0.262	0.360	0.230	*0.496*
huge class	0.991	0.992	N/A	*0.993*
total	0.983	0.984	N/A	*0.986*

Note “N/A” means results were not provided by the source paper [14]

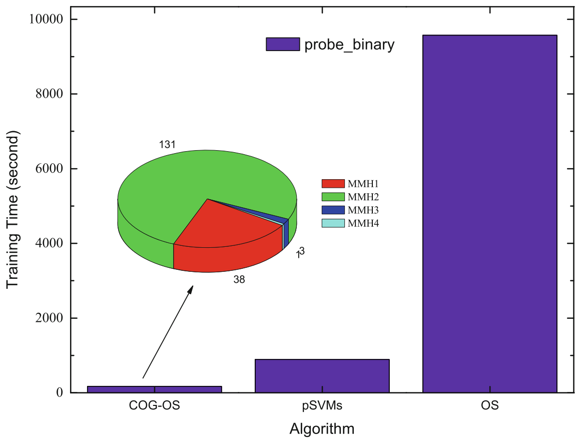


Fig. 6.7 The computational performance on the network intrusion data. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

494021. This implies that local clustering indeed helped to reduce the training time for SVMs. Actually, if we only learn SVMs but not employ local clustering on the over-sampled data set, as indicated by the “OS” column in Fig. 6.7, the training time increases dramatically up to over 2.5 h! To better illustrate the efficiency of COG-OS, we further recorded the time consumed for learning each of the four hyper-planes in COG-OS, as shown by the pie plot in Fig. 6.7. As can be seen, for four sub-classes produced by local clustering on the normal class, only one sub-class is much harder to be distinguished from the rare class, since it took longer time to find a hyper-plane for this sub-class and the rare class. For the rest three sub-groups, it took much less time to find the corresponding hyper-planes. The above indicates that local clustering can help to divide a complex structure/concept into several simple structures/concepts which are easy to be linearly separated and take less training time to find hyper-planes.

Table 6.8 Some characteristics of the credit card data set

Dataset	#instances		#features	*CV*
	Normal class	Rare class		
Training set	67763	13374	324	0.948
Test set	67763	13374	324	0.948

Table 6.9 Results on the credit card data set

	SVMs			RIPPER			COG		
	R	P	F	R	P	F	R	P	F
Normal class	0.969	0.856	0.909	0.955	0.874	0.913	0.949	0.881	0.914
Rare class	0.176	0.530	0.264	0.304	0.573	0.397	0.353	0.577	0.438
total	0.838	0.838	0.838	0.848	0.848	0.848	0.851	0.851	0.851

Note R recall, *P* precision, *F* F-measure

6.5.5 COG for Credit Card Fraud Detection

In this section, we showcase the application of COG for credit card fraud detection.

The Data Set. The experimental data set is from a security company. There are two classes in the data set: one normal class and one rare class. Table 6.8 shows some characteristics of the training and test data sets. As can be seen, the size of the rare class is approximately 20 % of the size of the normal class. And the sizes of the training and test data sets are the same here.

Tools and Settings. In this experiment, we also used pure SVMs, RIPPER, and COG(SVMs). For COG, the cluster number for the normal class was set to 5, and we did not over-sample the rare class. As to pure SVMs and RIPPER, we also used the default settings except that the kernel of SVMs is linear.

The Results. In Table 6.9, we can observe that COG shows the best performances on both the normal and rare classes. If we take a closer look on the recalls produced by pure SVMs and COG, we can notice that COG greatly improved the ability of SVMs in detecting more fraud cases. This is extremely important for credit card risk management, since the cost of missing one fraud case is often much higher than that of rejecting one normal application.

We also compared the computational efficiencies of pure SVMs and COG. As can be seen in Fig. 6.8, pure SVMs took 23.8 h in learning the hyper-planes. In contrast, COG only used 2.5 h for learning five hyper-planes. The runtime of COG is approximately ten percent of that of pure SVMs. Let us also take a close look at the time consumed in learning each of the five hyper-planes when applying COG. As shown in the pie plot of Fig. 6.8, approximately 77 % of training time consumed for learning two hyper-planes. This implies that we may further improve the performances of COG by further decomposing the sub-concepts separated by these two hyper-planes.

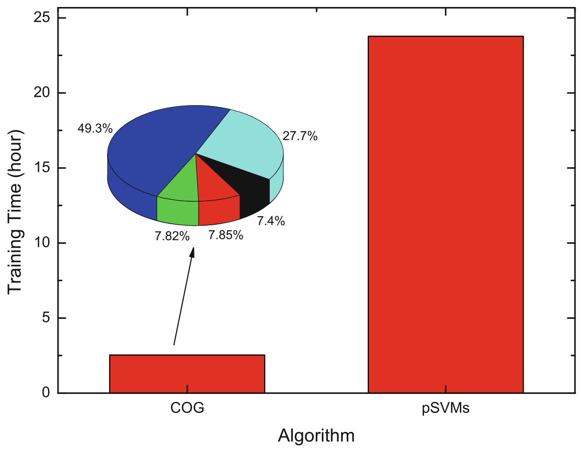


Fig. 6.8 The computational performance on the credit card fraud data. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

6.5.6 COG on Balanced Data Sets

In previous sections, we have shown that COG and COG-OS can improve the rare class analysis. Here, we would like to show COG is also applicable to data sets with balanced class sizes.

In this experiment, we used five balanced data sets including `letter`, `optdigits`, `pendigits`, `satimage` and `vowel`. The CV values of these data sets are less than 0.5. Among them, four data sets have been split into training and test sets by the UCI repository except for `letter`. Four classifiers including SVMs, BMR, C4.5 and RIPPER were used for the purpose of comparison. The local clustering method is K-means with the squared Euclidean distance and the cluster number for each class in a data set is exactly the same, in the range of four to eight.

Results by COG with Linear Classifiers. Table 6.10 shows the experimental results on these balanced data sets. As can be seen, for linear classifiers SVMs and BMR, COG can improve the classification accuracies no matter what the cluster number is. For instance, for the data set `letter`, the accuracies achieved by pure SVMs and BMR are merely 0.851 and 0.750, respectively (as indicated by the italic numbers). In contrast, COG with SVMs and BMR increased the prediction accuracies of the rare classes steadily as the increase of the cluster number, and finally up to 0.952 and 0.850, respectively when the cluster number was eight (as indicated by the bold numbers). Indeed, the resulting prediction accuracies are 10 % higher than the ones achieved by pure SVMs and BMR.

Next, we take a closer look at the performance of COG at the class level. Table 6.11 shows the classification accuracies on the data set `optdigits` by pure SVMs and COG. As can be seen, COG simultaneously improved the classification accuracies for nearly all the classes of `optdigits`. In addition, Fig. 6.9 shows the ratio of classes with accuracy gain by COG in the five balanced data sets (“#clusters”=8). A very similar improvement trend can be observed for all five data sets. Indeed, COG

Table 6.10 Performances of COG with different classifiers on balanced data sets

Data set	#Clusters	Classifier			
		COG(SVMs)	COG(BMR)	COG(C4.5)	COG(RIPPER)
letter	N/A	*0.851*	*0.750*	***0.862***	***0.839***
	2	0.872	0.762	0.846	0.821
	4	0.923	0.812	0.858	0.826
	6	0.946	0.844	0.854	0.818
	8	**0.952**	**0.850**	0.856	0.812
optdigits	N/A	*0.965*	*0.949*	*0.858*	***0.874***
	2	0.973	0.958	**0.880**	0.840
	4	0.973	0.970	0.855	0.816
	6	0.979	0.972	0.866	0.805
	8	**0.981**	**0.973**	0.860	0.771
pendigits	N/A	*0.953*	*0.901*	*0.921*	***0.925***
	2	0.969	0.937	0.920	0.918
	4	0.979	0.964	**0.927**	0.917
	6	**0.981**	0.962	0.923	0.897
	8	0.977	**0.967**	0.920	0.867
satimage	N/A	*0.852*	*0.834*	*0.854*	*0.854*
	2	0.860	0.837	0.841	**0.855**
	4	0.871	0.841	**0.857**	0.850
	6	**0.883**	0.862	0.850	0.848
	8	0.881	**0.863**	0.847	0.847
vowel	N/A	*0.517*	*0.448*	***0.517***	***0.468***
	2	**0.602**	0.517	0.392	0.312
	4	0.582	**0.541**	0.385	0.370
	6	0.580	0.491	0.370	0.314
	8	0.597	0.513	0.346	0.251

Notes 1. All classifiers used default settings except for SVMs: -t 0 2. For K-means, maxIteration = 500, repeat = 10

Table 6.11 Classification accuracies on `optdigits` in the class-wise level

Class ID	1	2	3	4	5	6	7	8	9	10
pure SVMs	0.994	0.967	0.960	0.934	0.989	0.989	0.989	0.950	0.920	0.956
COG(SVMs)	1.000	0.989	0.994	0.973	1.000	0.995	0.994	0.950	0.954	0.956
pure BMR	0.972	0.940	0.977	0.918	0.972	0.978	0.978	0.911	0.902	0.939
COG(BMR)	1.000	0.978	0.989	0.967	0.967	0.973	0.989	0.961	0.948	0.961

Note For COG, the cluster number is set to be 8 for every class

can divide the complex concepts in the data into the linearly separable concepts so as to simultaneously improve the performances of linear classifiers for most of the classes in the data.

Another interesting observation is that, the accuracy improvement gained by COG with SVMs and BMR are quite close. To illustrate this, we compute the *maximal accuracy gain* for each data set; that is, we first select the highest accuracy among

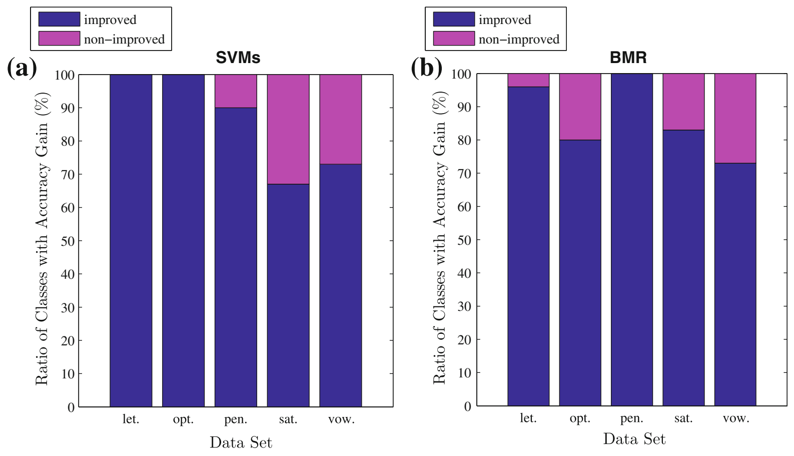
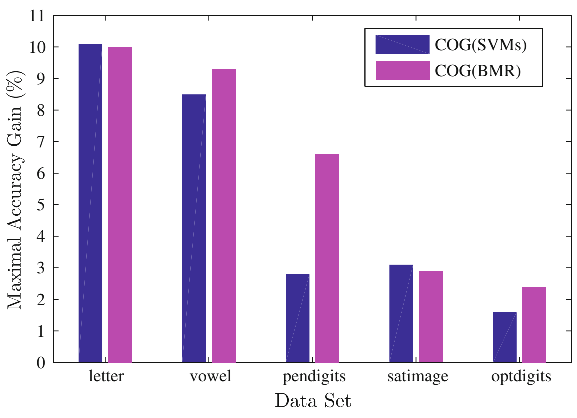


Fig. 6.9 Ratio of the classes with accuracy gain by COG. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

Fig. 6.10 A comparison of the maximal accuracy gains by COG(SVMs) and COG(BMR). Reprinted from Ref. [31], with kind permission from Springer Science+Business Media



four values achieved in different "#clusters" levels, then subtract it by the accuracy obtained by the pure classifier. Figure 6.10 shows the results. As can be seen, the maximal accuracy gains of all data sets by COG with SVMs and BMR are quite close except for `pendigits`. This implies that the complex concept in the data is the bottle-neck that hinders the analysis of linear classifiers.

Results by COG with Non-linear Classifiers. Table 6.10 also shows the results of COG with non-linear classifiers such as RIPPER and C4.5 on five balanced data sets. In the table, we can see that COG(RIPPER) has worse performances than pure RIPPER on all five data sets. This is due to the fact that the rule learning algorithm aims to build up a rule set in a greedy fashion by employing the standard divide-and-conquer strategy. Meanwhile, COG partitions instances of the same class into

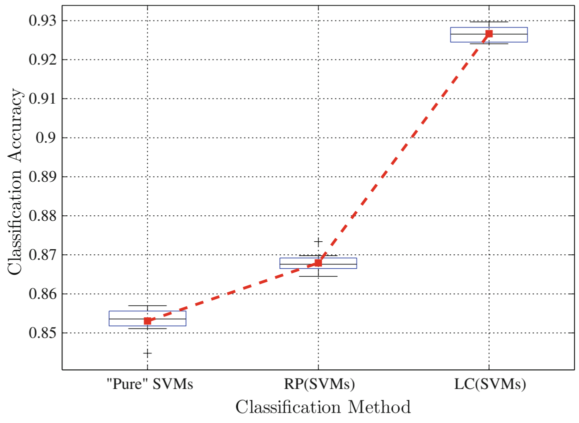


Fig. 6.11 Local clustering versus random partitioning on the letter data set. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

different sub-classes. This can increase the number of negative examples for some target rules, and ultimately results in missing such rules.

Finally, for another widely used non-linear classifier C4.5, the performance of COG with C4.5 is not consistent on five balanced data sets, as shown in Table 6.10. For instance, COG improved the classification accuracy of optdigits, but led to worse performances on letter, vowel and satimage. This is due to the fact that COG can increase the number of sub-classes so as to make the branch-splitting decision even harder to make. In other words, the splitting attributes of the tree can be better or worse selected in such "uncertain" scenarios, which results in the inconsistent performances.

Local Clustering Versus Random Partitioning. In this experiment, we compare the effect of local clustering in the COG scheme with that of simple random partitioning. To this end, we take the data set letter and classifier SVMs to illustrate this. First, we randomly split the letter data set into two parts, 70 % of which as the training set and the rest as the test set (the class size distribution holds). Then we performed training and testing in a very similar fashion to the procedure of COG except that we use random partitioning instead of local clustering on each class.

Figure 6.11 shows all the experimental results. Please note that for each class, the number of local clusters is four. As indicated by Fig. 6.11, while the performance of random partitioning with SVMs, i.e., RP(SVMs), is slightly better than the results of pure SVMs, they are much worse than the results by local clustering with SVMs, i.e., LC(SVMs). This indicates that the local clustering phase in COG is very important. In fact, random partitioning can only provide the balance effect, which cannot help to decompose complex concepts into simple and linearly separable concepts.

In summary, COG is of great use on improving the classification accuracy of linear classifiers by eliminating or mitigating the complex concepts in the data. But for the non-linear classifiers, such as C4.5 and RIPPER, COG shows no competitive results.

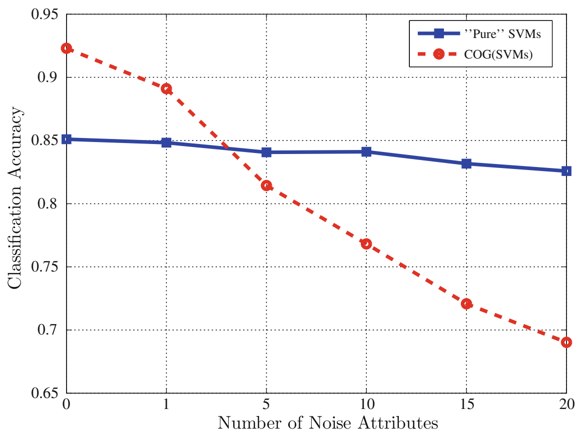


Fig. 6.12 Illustration of the effect of noise attributes on COG. Reprinted from Ref. [31], with kind permission from Springer Science+Business Media

6.5.7 Limitations of COG

Despite of various merits described above, COG also has some limitations. For example, since COG combines the use of supervised classification techniques with unsupervised local clustering technique, if there are some inconsistent uses of the set of attributes for classification and clustering, the performance of COG can be degraded.

For classification, class labels can help to build up a classifier mainly based on some relevant attributes; that is, classification algorithms can inherently select attributes related to class labels. However, due to the lack of information of class labels, clustering methods tend to assign equal weights to all the attributes, thus cannot avoid the negative impact of irrelevant or weakly relevant attributes. Therefore, the set of attributes for classification can be different from the set of attributes for local clustering in COG. This inconsistent use of attributes can lead to difficulties for the COG scheme. Indeed, when there are many noise attributes in the data, COG may not be very effective. Let us illustrate this by an example as follows.

In this example, we added some "noise" attributes to the `letter` data set, which contains 16 attributes originally. The added attributes consist of numbers generated randomly from the uniform distribution and scaled to the same value interval as the original attributes. We gradually increased the number of noise attributes and observed the change of the classification accuracies by pure SVMs and COG (the cluster number is 4 for each class). Figure 6.12 shows the results. As can be seen, the classification accuracy of pure SVMs does not change much as the increase of the number of noise attributes. This indicates that noise features do not have much influence on the performance of pure SVMs. In contrast, the accuracy of COG is significantly reduced as the increase of noise features. This means noise attributes indeed have a strong negative impact on the performance of the COG method. The reason is that the quality of local clustering in COG is sensitive to the use of attributes.

Table 6.12 The effect of feature selection on COG

Data set	Type	#Features	Pure SVMs	COG
`segment`	Without F.S.	18	0.941	0.899
	with F.S.	7	0.939	**0.950**
`splice`	Without F.S.	60	0.848	0.838
	with F.S.	12	0.840	**0.880**

To further demonstrate this, we exploited feature selection on two data sets: `segment` and `splice`. COG does not work well on these two data sets. Table 6.12 shows the classification results before and after feature selection. An interesting observation is that, by feature selection, COG indeed achieved a much better accuracy than that produced by pure SVMs (with or without feature selection). Note that the feature selection scheme we used here is "SVMAttributeEval" available in WEKA 3.5 [30], and the corresponding search method is "Ranker", all with default settings.

In summary, while COG works very well for linear classifiers in many cases, it is sensitive to the presence of irrelevant or noise attributes. However, by applying appropriate feature selection methods, we can ease this problem significantly.

6.6 Related Work

In the literature, there are a number of methods addressing the class imbalance problem. The sampling based methods are one of the simplest yet effective ones [26]. For instance, the over-sampling scheme replicates the small classes to match the sizes of large classes [19]. In contrast, the under-sampling method cuts down the large class sizes to achieve a similar effect [18]. Drummond and Holte provided detailed comparisons on these two re-sampling schemes [7]. Chawla et al. demonstrated that the combination of over-sampling and under-sampling can achieve better performances [2]. Another popular method is the cost-sensitive learning scheme which takes the cost matrix into consideration during model building and generates a model that has the lowest cost. The properties of a cost matrix had been studied by Elkan [9]. Margineantu and Dietterich examined various methods for incorporating cost information into the C4.5 learning algorithm [22]. Other cost-sensitive learning methods that are algorithm-independent include AdaCost [10], MetaCost [6], and Costing [32]. In addition, Joshi et al. discussed the limitations of boosting algorithms for rare class modeling [15], and proposed PNrule, a two-phase rule induction algorithm, to handle the rare class purposefully [14]. Other algorithms developed for mining rare classes include BRUTE [25], SHRINK [16], RIPPER [3], etc. A good survey paper is given by [28].

Finally, Japkowicz showed the idea of "supervised learning with unsupervised output separation" [13]. This work shares some common grounds with our COG method in terms of combining supervised and unsupervised learning techniques. However, we have a novel perspective on rare class analysis in this study. We develop the foundation of classification using local clustering (COG) for enhancing linear classifiers on handling both balanced and imbalanced classification problems.

6.7 Concluding Remarks

In this chapter, we proposed a method for classification using local clustering (COG). The key idea is to perform local clustering within each class and produce linearly separable sub-classes with relatively balanced sizes. For data sets with imbalanced class distributions, the COG method can improve the performance of traditional supervised learning algorithms, such as Support Vector Machines (SVMs), on rare class analysis. In addition, the COG method has the capability in decomposing complex structures/concepts in the data into simple and linearly separable concepts, and thus enhancing linear classifiers on data sets containing linearly inseparable classes. Finally, as demonstrated by our experimental results on various real-world data sets, COG with over-sampling can have much better prediction accuracy on rare classes than state-of-the-art methods. Also, the COG method can significantly improve the computational efficiency of SVMs.

References

1. Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual Workshop on Computational Learning Theory, pp. 144–152 (1992)
2. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
3. Cohen, W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning, pp. 115–123 (1995)
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)
5. DeGroot, M., Schervish, M.: Probability and Statistics, 3rd edn. Addison Wesley, Reading (2001)
6. Domingos, P.: Metacost: a general method for making classifiers cost-sensitive. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 155–164 (1999)
7. Drummond, C., Holte, R.: C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: Proceedings of the 20th International Conference on Machine Learning, Workshop on Learning from Imbalanced Data Sets II (2003)
8. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley-Interscience, New York (2000)
9. Elkan, C.: The foundations of cost-sensitive learning. In: Proceedings of the 2001 International Joint Conferences on Artificial Intelligence, pp. 973–978 (2001)
10. Fan, W., Stolfo, S., Zhang, J., Chan, P.: Adacost: misclassification cost-sensitive boosting. In: Proceedings of the 16th Internation Conference on Machine Learning, pp. 97–105 (1999)

11. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proceedings of the 2nd European Conference on Computational Learning Theory, pp. 23–37 (1995)
12. Han, E.H., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Webace: a web agent for document categorization and exploration. In: Proceedings of the 2nd International Conference on Autonomous Agents (1998)
13. Japkowicz, N.: Supervised learning with unsupervised output separation. In: Proceedings of the 6th International Conference on Artificial Intelligence and Soft Computing, pp. 321–325 (2002)
14. Joshi, M., Agarwal, R., Kumar, V.: Mining needle in a haystack: classifying rare classes via two-phase rule induction. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, pp. 91–102 (2001)
15. Joshi, M., Kumar, V., Agarwal, R.: Evaluating boosting algorithms to classify rare classes: comparison and improvements. In: Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 257–264 (2001)
16. Kubat, M., Holte, R., Matwin, S.: Learning when negative examples abound. In: Proceedings of the 9th European Conference on Machine Learning, pp. 146–153 (1997)
17. Kubat, M., Holte, R., Matwin, S.: Machine learning for the detection of oil spills in satellite radar images. Machine Learning **30**, 195–215 (1998)
18. Kubat, M., Matwin, S.: Addressing the curse of imbalanced training sets: one-sided selection. In: Proceedings of the 14th International Conference on Machine Learning, pp. 179–186 (1997)
19. Ling, C., Li, C.: Data mining for direct marketing: problems and solutions. In: Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 73–79 (1998)
20. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
21. Maimon, O., Rokach, L. (eds.): The Data Mining and Knowledge Discovery Handbook. Springer, Heidelberg (2005)
22. Margineantu, D., Dietterich, T.: Learning decision trees for loss minimization in multi-class problems. Tech. Rep. TR 99–30-03, Oregon State University (1999)
23. Porter, M.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
24. Raudys, S., Jain, A.: Small sample size effects in statistical pattern recognition: recommendations for practitioners. IEEE Trans. Pattern Anal. Mach. Intell. **13**(3), 252–264 (1991)
25. Riddle, P., Segal, R., Etzioni, O.: Representation design and brute-force induction in a boeing manufacturing design. Appl. Artif. Intell. **8**, 125–147 (1994)
26. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2005)
27. Vapnik, V.: The Nature of Statistical Learning. Springer, New York (1995)
28. Weiss, G.: Mining with rarity: a unifying framework. ACM SIGKDD Explor. **6**(1), 7–19 (2004)
29. Weiss, G., Hirsh, H.: Learning to predict rare events in event sequences. In: Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 359–363 (1998)
30. Witten, I., Frank, E.: Data mining: practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
31. Wu, J., Xiong, H., Chen, J.: Cog: local decomposition for rare class analysis. Data Min. Knowl. Discov. **20**, 191–220 (2010)
32. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: Proceedings of the 2003 IEEE International Conference on Data Mining, pp. 435–442 (2003)
33. Zurada, J., Foster, B., Ward, T.: Investigation of artificial neural networks for classifying levels of financial distress of firms: the case of an unbalanced training sample. In: Abramowicz, W., Zurada, J. (eds.) Knowledge Discovery for Business Information Systems, pp. 397–423. Kluwer, Dordrecht (2001)

Chapter 7
K-means Based Consensus Clustering

7.1 Introduction

Consensus clustering [14, 16], also known as *cluster ensemble* or *clustering aggregation*, aims to find a single clustering from multi-source basic clusterings on the same group of data objects. It has been widely recognized that consensus clustering has merits in generating better clusterings, finding bizarre clusters, handling noise, outliers and sample variations, and integrating solutions from multiple distributed sources of data or attributes [15].

The problem of consensus clustering is NP-complete in essence. In the literature, many algorithms have been proposed to address the computational challenge, such as the co-association matrix based methods [6], the graph-based methods [16], the prototype-based clustering methods [18, 19], and other heuristic approaches [11]. Among these research studies, the K-means based consensus clustering method [18] is of particular interests for its simplicity, robustness and high efficiency inherited from classic K-means clustering. However, the existing works along this line are still scattered and fragmented. The general theoretic framework of utility functions suitable for K-means based consensus clustering (KCC) is still not available.

To fulfill this crucial void, in this chapter, we provide a systematic study of utility functions for K-means based consensus clustering. The major contributions are summarized as the following. First, we propose a sufficient condition for utility functions which are suitable for KCC. Indeed, by this condition, we can easily derive a *KCC utility function* from a continuously differentiable convex function. This helps to establish a unified framework for K-means based consensus clustering. Second, we design the new computation process of utility functions (in consensus clustering) and distance functions (in K-means clustering), and successfully adapt KCC for handling inconsistent data.

Finally, we have conducted extensive experiments on real-world data to evaluate the performances of KCC using different utility functions. Experimental results demonstrate that: (1) KCC is an efficient and effective method, which generates clustering results comparable to the results by state-of-the-art graph-partitioning method;

(2) While diversified utility functions bring extra flexibility to KCC in different application domains, the utility functions based on the Shannon entropy often lead to better performances of KCC; (3) KCC is a very robust method, which can generate stable results on basic clusterings of varying qualities, even for severely inconsistent data; (4) The generation strategy of basic clusterings has a strong impact on KCC, and the Random Feature Selection (RFS) strategy is particularly effective in handling data with noisy features and finding the true number of clusters.

The remainder of this chapter is organized as follows. In Sect. 7.2, we give some preliminaries and provide the problem definition. Section 7.3 introduces the sufficient condition for utility functions that can be used for K-means based consensus clustering. In Sect. 7.4, we study how to exploit KCC for inconsistent data. Section 7.5 presents the experimental results. In Sect. 7.6, we describe the related work. Finally, Sect. 7.7 concludes this work.

7.2 Problem Definition

Here, we briefly introduce the basic concepts of consensus clustering, and formulate the problem to be studied in this chapter.

7.2.1 Consensus Clustering

Let $\mathscr{X} = \{x_1, x_2, \ldots, x_n\}$ denote a set of data objects. A partitioning of $\mathscr{X}$ into K crisp clusters can be represented as a collection of K sets of objects: $\mathscr{C} = \{C_k | k = 1, \ldots, K\}$, with $C_k \bigcap C_{k'} = \emptyset, \forall k \neq k'$ and $\bigcup_{k=1}^{K} C_k = \mathscr{X}$, or as a label vector: $\pi = \langle L_\pi(x_1), \ldots, L_\pi(x_n) \rangle$, where $L_\pi(x_i)$ maps x_i to one of the K labels: $1, 2, \ldots, K$.

The problem of consensus clustering is typically formulated as follows. Given a set of r basic partitionings: $\Pi = \{\pi_1, \pi_2, \ldots, \pi_r\}$ of $\mathscr{X}$, the goal is to find a consensus partitioning π such that

$$\Gamma(\pi, \Pi) = \sum_{i=1}^{r} w_i U(\pi, \pi_i) \qquad (7.1)$$

is maximized, where $\Gamma : \mathbb{N}^n \times \mathbb{N}^{nr} \mapsto \mathbb{R}$ is a *consensus function*, $U : \mathbb{N}^n \times \mathbb{N}^n \mapsto \mathbb{R}$ is a *utility function*, and $w_i \in \mathbb{R}_{++}$ is the user-specified weight for π_i, $i = 1, 2, \ldots, r$.

The choice of the utility function is critical for the success of a consensus clustering. In the literature, many external measures originally proposed for cluster validity have been adopted as the utility functions for consensus clustering, such as Normalized Mutual Information [16], Category Utility Function [13], Quadratic Mutual Information [18], and Rand Index [11]. These utility functions, together with the consensus function, largely determine the quality of consensus clustering.

Table 7.1 The contingency matrix

		π_i				
		$C_1^{(i)}$	$C_2^{(i)}$	$\cdots$	$C_{K_i}^{(i)}$	$\sum$
	C_1	$n_{11}^{(i)}$	$n_{12}^{(i)}$	$\cdots$	$n_{1K_i}^{(i)}$	n_{1+}
π	C_2	$n_{21}^{(i)}$	$n_{22}^{(i)}$	$\cdots$	$n_{2K_i}^{(i)}$	n_{2+}
	$\cdot$	$\cdot$	$\cdot$	$\cdots$	$\cdot$	$\cdot$
	C_K	$n_{K1}^{(i)}$	$n_{K2}^{(i)}$	$\cdots$	$n_{KK_i}^{(i)}$	n_{K+}
	$\sum$	$n_{+1}^{(i)}$	$n_{+2}^{(i)}$	$\cdots$	$n_{+K_i}^{(i)}$	n

7.2.2 *K-means Based Consensus Clustering*

The computation issue is always the critical concern of consensus clustering, since maximizing Eq. (7.1) is an NP-complete problem. Many algorithms have been proposed to address this challenge [5, 11, 16, 18]. Among these approaches, a K-means based method is of particular interests, which was firstly studied by [13], and later introduced to consensus clustering by [18]. Here, we call it the *K-means based Consensus Clustering* (KCC) problem, and revisit it briefly from a contingency-matrix perspective as follows.

A *contingency matrix* is often employed for computing the difference of two partitionings. As can be seen from Table 7.1, $n_{kj}^{(i)}$ denotes the number of data objects contained by both cluster $C_j^{(i)}$ in π_i and cluster C_k in π, $n_{k+} = \sum_{j=1}^{K_i} n_{kj}^{(i)}$, and $n_{+j}^{(i)} = \sum_{k=1}^{K} n_{kj}^{(i)}$, $1 \le k \le K$, $1 \le j \le K_i$. Let $p_{kj}^{(i)} = n_{kj}^{(i)}/n$, $p_{k+} = n_{k+}/n$, and $p_{+j}^{(i)} = n_{+j}^{(i)}/n$, we then have the *normalized contingency matrix* for utility computation. For instance, the well-known Category Utility Function for KCC [13] can be computed as follows:

$$U_c(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+} \sum_{j=1}^{K_i} (p_{kj}^{(i)}/p_{k+})^2 - \sum_{j=1}^{K_i} (p_{+j}^{(i)})^2. \tag{7.2}$$

We then construct a *binary* data set $\mathscr{X}^{(b)} = \{x_l^{(b)} | 1 \le l \le n\}$ according to Π as follows:

$$x_l^{(b)} = \langle x_{l,1}^{(b)}, \ldots, x_{l,i}^{(b)}, \ldots, x_{l,r}^{(b)} \rangle, \tag{7.3}$$

$$x_{l,i}^{(b)} = \langle x_{l,i1}^{(b)}, \ldots, x_{l,ij}^{(b)}, \ldots, x_{l,iKi}^{(b)} \rangle, \tag{7.4}$$

$$x_{l,ij}^{(b)} = \begin{cases} 1, \text{ if } L_{\pi_i}(x_l) = j \\ 0, \text{ otherwise} \end{cases}, \tag{7.5}$$

where L_{π_i} maps x_l to one of the K_i labels in π_i. This implies that $\mathscr{X}^{(b)}$ is an $n \times \sum_{i=1}^{r} K_i$ binary data matrix with $\sum_{j=1}^{K_i} x_{l,ij}^{(b)} = 1, \forall l, i$. In [13], $\mathscr{X}^{(b)}$ is further centralized to $\mathscr{X}^{(c)} = \{x_l^{(c)} | 1 \le l \le n\}$ where $x_{l,ij}^{(c)} = x_{l,ij}^{(b)} - \sum_{l=1}^{n} x_{l,ij}^{(b)}/n$, which leads to:

$$\max_{\pi} \sum_{i=1}^{r} U_c(\pi, \pi_i) \Longleftrightarrow \min_{\mathscr{C}} \sum_{k=1}^{K} \sum_{x_l^{(c)} \in C_k} \|x_l^{(c)} - m_k\|^2, \tag{7.6}$$

where m_k is the arithmetic mean of all $x_l^{(c)} \in C_k$. Eq. (7.6) indicates that, given $w_i = 1$ $\forall i$ and $U \doteq U_c$, the maximization of Γ in Eq. (7.1) is equivalent to the minimization of the sum of inner-cluster variances of $\mathscr{X}^{(c)}$, which can be solved quickly by the centroid-based alternating optimization process of the K-means clustering [12].

7.2.3 *Problem Definition*

While its properties have not been fully understood in the literature, KCC is expected to have appealing merits inherited from the K-means algorithm (i.e. simple, efficient, and robust). In this chapter, we focus on studying the general framework of K-means based consensus clustering. Specifically, we aim to answer the following two questions:

- What kind of utility functions can be used for K-means based consensus clustering?
- How well do these utility functions perform for K-means based consensus clustering on real-world data sets?

The answers to these questions can not only establish a theoretic framework for KCC, but also provide a better guidance for the practical use of KCC.

7.3 Utility Functions for K-means Based Consensus Clustering

In this section, we give a sufficient condition for utility functions that can be used for K-means based consensus clustering (KCC). First, given the symbols in Sect. 7.2, we have the definition of utility functions for KCC as follows:

Definition 7.1 (**KCC Utility Function**) A utility function U is called a KCC utility function, if there exists a distance function f such that

$$\max_{\pi} \sum_{i=1}^{r} w_i U(\pi, \pi_i) \Longleftrightarrow \min_{\mathscr{C}} \sum_{k=1}^{K} \sum_{x_l^{(b)} \in C_k} f(x_l^{(b)}, m_k). \tag{7.7}$$

Definition 7.1 implies that, by using KCC utility functions, the search for the consensus clustering π is equivalent to the K-means clustering of the binary data set $\mathscr{X}^{(b)}$. Thus, we can reduce an NP-complete problem to a roughly linear one. As a result, in what follows, we focus on finding KCC utility functions.

7.3.1 The Distance Functions for K-means

To understand U in Eq. (7.7), we should first understand f, the distance function of K-means. It is interesting to note that the choice of distance functions is highly correlated with the choice of centroid types, given that the convergence of the K-means algorithm must be guaranteed [17]. For instance, if the well-known squared Euclidean distance is used, the centroids must be the arithmetic means of cluster members. However, if the city-block is used instead, the centroids must be the medians. By taking account of the computational efficiency and the mathematical property, we limit K-means in Eq. (7.7) to the one with the centroid type of the arithmetic mean, i.e., the classic K-means.

Recall the point-to-centroid distance defined in Definition 3.2 of Chap. 3. It has been pointed out in Corollary 3.4 that, the point-to-centroid distance is the only distance function suitable for the classic K-means clustering. This indicates that f in Eq. (7.7) must be a point-to-centroid distance taking the form as follows:

$$f(x, y) = \phi(x) - \phi(y) - (x - y)^T \nabla\phi(y), \tag{7.8}$$

where ϕ is a continuously differentiable convex function defined on an open convex set. As different ϕ can lead to different instances, the point-to-centroid distance is actually a family of multiple distance functions. Some popular point-to-centroid distances widely used for K-means clustering, e.g. the squared Euclidean distance (d^2), the KL-divergence (D), the cosine distance (f_{cos}), and the l_p distance (f_{l_p}), can be found in Table 3.2 of Chap. 3.

7.3.2 A Sufficient Condition for KCC Utility Functions

Given the math symbols in Sect. 7.2, we have a sufficient condition for being a KCC utility function as follows:

Theorem 7.1 *A utility function U is a KCC utility function, if there exists a continuously differentiable convex function ϕ such that*

$$\sum_{i=1}^{r} w_i U(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+}\phi(m_k), \tag{7.9}$$

where $\Pi = \{\pi_1, \ldots, \pi_r\}$ and π are arbitrary partitionings of $\mathcal{X}$, $p_{k+} = |C_k|/|\mathcal{X}|$ is the size ratio of cluster C_k in π, and m_k is the centroid of C_k when applying π to $\mathcal{X}^{(b)}$.

Proof For a binary data set $\mathcal{X}^{(b)}$, the objective function of K-means is

$$\min_{\pi} \sum_{k=1}^{K} \sum_{x_l^{(b)} \in C_k} f(x_l^{(b)}, m_k), \tag{7.10}$$

where f is a point-to-centroid distance derived by some continuously differentiable convex function ϕ. Also, according to Eq. (7.8), we have

$$\begin{aligned} &\sum_{k=1}^{K} \sum_{x_l^{(b)} \in C_k} f(x_l^{(b)}, m_k) \\ \overset{(\alpha)}{=} &\sum_{x_l^{(b)} \in \mathcal{X}} \phi(x_l^{(b)}) - n \sum_{k=1}^{K} p_{k+} \phi(m_k) \\ \overset{(\beta)}{=} &\sum_{x_l^{(b)} \in \mathcal{X}} \phi(x_l^{(b)}) - n \sum_{i=1}^{r} w_i U(\pi, \pi_i). \end{aligned}$$

Note that (α) holds since $\sum_{x_l^{(b)} \in C_k} x_l^{(b)} - |C_k| m_k = 0$, and (β) holds according to Eq. (7.9). Since both $\sum_{x_l^{(b)} \in \mathcal{X}} \phi(x_l^{(b)})$ and n are constants, we finally have Eq. (7.7). Therefore, by Definition 7.1, U is a KCC utility function. □

Remark Theorem 7.1 provides a criterion for recognizing KCC utility functions; that is, if we can transform the consensus function $\Gamma(\pi, \Pi) = \sum_{i=1}^{r} w_i U(\pi, \pi_i)$ into the form: $\sum_{k=1}^{K} p_{k+} \phi(m_k)$, then U is a KCC utility function. In other words, a KCC utility function relates the contingency-matrix-based utility computation to the K-means clustering of a binary data set $\mathcal{X}^{(b)}$.

In what follows, we continue to explore the properties of ϕ in Eq. (7.9). To this end, we first revisit the binary data set $\mathcal{X}^{(b)}$ generated from Π. As noted in Sect. 7.2.2, $\mathcal{X}^{(b)} = [x_{l,ij}^{(b)}]_{n \times \sum_{i=1}^{r} K_i}$ consists of n records, each of which contains $\sum_{i=1}^{r} K_i$ binary attributes with values in $\{0, 1\}$. As a result, the k-th centroid m_k of partitioning π in Eq. (7.9) is also a $\sum_{i=1}^{r} K_i$-dimensional vector as follows:

$$m_k = \langle m_{k,1}, \ldots, m_{k,i}, \ldots, m_{k,r} \rangle, \tag{7.11}$$

$$m_{k,i} = \langle m_{k,i1}, \ldots, m_{k,ij}, \ldots, m_{k,iK_i} \rangle, \tag{7.12}$$

where $1 \le k \le K$, $1 \le i \le r$, and $1 \le j \le K_i$. Also, according to the contingency matrix in Table 7.1, we have

$$m_{k,ij} = \frac{\sum_{x_l^{(b)} \in C_k} x_{l,ij}^{(b)}}{n_{k+}} = \frac{n_{kj}^{(i)}}{n_{k+}} = \frac{p_{kj}^{(i)}}{p_{k+}}. \tag{7.13}$$

Then, we have a theorem about ϕ as follows:

Theorem 7.2 *If U is a KCC utility function satisfying Eq. (7.9), then there exists a continuously differentiable convex function φ such that*

$$\phi(m_k) = \sum_{i=1}^{r} w_i \varphi(m_{k,i}),\ 1 \le k \le K. \tag{7.14}$$

Proof Denote $\phi(m_{k,i})$ by $\varphi(m_{k,i})$. Since Eq. (7.9) holds for individual π_i and a same π, for each $1 \le i \le r$ we have

$$U(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+} \varphi(m_{k,i}). \tag{7.15}$$

Accordingly, we have

$$\sum_{i=1}^{r} w_i U(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+} \sum_{i=1}^{r} w_i \varphi(m_{k,i}). \tag{7.16}$$

By comparing Eq. (7.16) with Eq. (7.9), and taking the arbitrariness of π into consideration, we have

$$\phi(m_k) = \sum_{i=1}^{r} w_i \varphi(m_{k,i}),\ 1 \le k \le K, \tag{7.17}$$

which indicates that the theorem holds. □

Remark From the application viewpoint, Theorem 7.2 is an important supplement to Theorem 7.1. It reveals that ϕ can be derived by simply summarizing the weighted φ on $m_{k,i}$, $i = 1, \ldots, r$. In other words, given φ and the weights, we can define the K-means clustering in correspondence with the consensus clustering.

Based on Theorem 7.2, we can further explore the properties of KCC utility functions satisfying Eq. (7.9). We have the following corollary:

Corollary 7.1 If U is a KCC utility function satisfying Eq. (7.9), then there exists a continuously differentiable convex function φ such that $\forall i$

Table 7.2 Some examples of KCC utility functions

	$\varphi(m_{k,i})$	$U(\pi,\pi_i)$	$f(x_l^{(b)},m_k)$
U_c	$\sum_{j=1}^{K_i} m_{k,ij}^2 - \sum_{j=1}^{K_i}(p_{+j}^{(i)})^2$	$\sum_{k=1}^{K} p_{k+}\sum_{j=1}^{K_i}(p_{kj}^{(i)}/p_{k+})^2 - \sum_{j=1}^{K_i}(p_{+j}^{(i)})^2$	$\sum_{i=1}^{r} w_i\Vert x_{l,i}^{(b)} - m_{k,i}\Vert^2$
U_H	$\sum_{j=1}^{K_i} m_{k,ij}\log m_{k,ij} - \sum_{j=1}^{K_i} p_{+j}^{(i)}\log p_{+j}^{(i)}$	$MI(\mathscr{C},\mathscr{C}^{(i)})$	$\sum_{i=1}^{r} w_i D(x_{l,i}^{(b)}\Vert m_{k,i})$
$U_{\cos}$	$\Vert m_{k,i}\Vert - \Vert\langle p_{+1}^{(i)},\cdots,p_{+K_i}^{(i)}\rangle\Vert$	$\sum_{k=1}^{K} p_{k+}\sqrt{\sum_{j=1}^{K_i}(p_{kj}^{(i)}/p_{k+})^2} - \sqrt{\sum_{j=1}^{K_i}(p_{+j}^{(i)})^2}$	$\sum_{i=1}^{r} w_i(1-\cos(x_{l,i}^{(b)},m_{k,i}))$
U_{L_p}	$\Vert m_{k,i}\Vert_p - \Vert\langle p_{+1}^{(i)},\cdots,p_{+K_i}^{(i)}\rangle\Vert_p$	$\sum_{k=1}^{K} p_{k+}\sqrt[p]{\sum_{j=1}^{K_i}(p_{kj}^{(i)}/p_{k+})^p} - \sqrt[p]{\sum_{j=1}^{K_i}(p_{+j}^{(i)})^p}$	$\sum_{i=1}^{r} w_i(1-\frac{\sum_{j=1}^{K_i} x_{l,ij}^{(b)}(m_{k,ij})^{p-1}}{(\Vert m_{k,i}\Vert_p)^{p-1}})$

Notes: (1) MI mutual information; (2) D relative entropy or KL-divergence; (3) log the logarithm of base 2; (4) $U_{\cos}$ is the same as U_{L_2}.

$$U(\pi,\pi_i) = \sum_{k=1}^{K} p_{k+}\varphi(\langle p_{k1}^{(i)}/p_{k+},\cdots,p_{kK_i}^{(i)}/p_{k+}\rangle). \tag{7.18}$$

Remark Compared with Theorem 7.1, Corollary 7.1 provides a more practical guidance for the recognition of KCC utility functions. To understand this, let us recall the normalized contingency matrix in Sect. 7.2.2, from which we can learn that $U(\pi,\pi_i)$ in Eq. (7.18) is nothing more than the weighted average of φ on normalized elements in each row. As a result, given a φ, we can derive a KCC utility function. Hereinafter, we denote the KCC utility functions derived by Eq. (7.18) as U_φ for simplicity.

In summary, a continuously differentiable function φ serves as the bridge between the consensus clustering and the K-means clustering. Table 7.2 shows some examples of KCC utility functions (U) derived from various convex functions (φ), and their corresponding point-to-centroid distance (f). Therefore, for the K-means based consensus clustering, we have an algorithmic framework as follows: (1) Select an appropriate U_φ; (2) Decompose U_φ to get φ; (3) Derive ϕ and f from φ; (4) Apply K-means on binary data $\mathscr{X}^{(b)}$ using f as the distance function. Then, the resultant partitioning is returned as the consensus clustering π we want.

7.3.3 *The Non-Unique Correspondence and the Forms of KCC Utility Functions*

Here, we discuss the correspondence between KCC utility functions and K-means clustering. We have two forms of U_φ as follows.

The standard form of $\mathbf{U}_{\boldsymbol{\varphi}}$. It is interesting to note that U_φ and the point-to-centroid distance have a many-to-one correspondence. In other words, different utility functions may correspond to a same point-to-centroid distance, and therefore lead to a same consensus partitioning π. To illustrate this, suppose we have a utility function U_φ derived from φ. If we let

$$\varphi_s(m_{k,i}) = \varphi(m_{k,i}) - \underbrace{\varphi(\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle)}_{(\alpha)}, \tag{7.19}$$

then by Eq. (7.18), we can construct a new utility function as follows:

$$U_{\varphi_s}(\pi, \pi_i) = U_\varphi(\pi, \pi_i) - \varphi(\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle). \tag{7.20}$$

Since (α) is a constant in Eq. (7.19) given π_i, it is easy to show that U_{φ_s} and U_φ correspond to a same point-to-centroid distance. Moreover, since $U_{\varphi_s} \geq 0$ for the convexity of φ (using the Jensen's inequality), U_{φ_s} can be regarded as the *utility gain* after calibrating U_φ to the benchmark: $\varphi(\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle)$. Here, we define utility gain as the standard form of a KCC utility function. As a result, all the utility functions listed in Table 7.2 are in the standard form.

The normalized form of $\mathbf{U}_{\boldsymbol{\varphi}}$. It is natural to take a further step from the standard form U_{φ_s} to the *normalized* form U_{φ_n}. Let

$$\varphi_n(m_{k,i}) = \varphi_s(m_{k,i}) / |\varphi(\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle)|. \tag{7.21}$$

Since $\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle$ is a constant vector given π_i, it is easy to note that φ_n is also a convex function from which a KCC utility function U_{φ_n} can be derived. That is,

$$U_{\varphi_n}(\pi, \pi_i) = \frac{U_\varphi(\pi, \pi_i) - \varphi(\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle)}{|\varphi(\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle)|}. \tag{7.22}$$

From Eq. (7.22), $U_{\varphi_n} \geq 0$ can be viewed as the *utility gain ratio* to the constant $\varphi(\langle p_{+1}^{(i)}, \cdots, p_{+K_i}^{(i)} \rangle)$. As a result, the corresponding point-to-centroid distance of U_{φ_n} will be the same as U_φ, if we let $w_i \leftarrow w_i / |\varphi(\langle p_{+1}^{(i)}, \ldots, p_{+K_i}^{(i)} \rangle)|, i = 1, \ldots, r$.

Since they have clear physical meanings, the standard form and the normalized form become the two major forms of U_φ we will use in the following experimental section.

7.3.4 Discussions

It is noteworthy that Theorem 7.1 and Corollary 7.1 only provide a sufficient condition for a utility function to be a KCC utility function. However, the necessary condition has not been fully understood yet.

Recall Definition 7.1, which defines the concept of KCC utility functions. Similar to the proof of Theorem 7.1, since f is a point-to-centroid distance, a utility function is a KCC utility function if and only if

$$\max_{\pi} \sum_{i=1}^{r} w_i U(\pi, \pi_i) \Longleftrightarrow \max_{\pi} \sum_{k=1}^{K} p_{k+} \phi(m_k). \tag{7.23}$$

where ϕ is a continuously differentiable function.

Equation (7.23) is the necessary and sufficient condition for a KCC utility function. However, it has only the theoretical significance, since it is hard to link ϕ to U given only the equivalence relation. A more practical way is to replace equivalence by equality, as we have done in Theorem 7.1, which gives a sufficient condition. In fact, we can have a more general sufficient condition as follows:

Theorem 7.3 *A utility function U is a KCC utility function, if there exist a continuously differentiable function ϕ and a positive monotonic transformation g such that*

$$g\left(\sum_{i=1}^{r} w_i U(\pi, \pi_i)\right) = \sum_{k=1}^{K} p_{k+} \phi(m_k). \tag{7.24}$$

Proof Since g is a positive monotonic transformation, or more directly, a strictly increasing function, we have

$$\max_{\pi} g\left(\sum_{i=1}^{r} w_i U(\pi, \pi_i)\right) \Longleftrightarrow \max_{\pi} \sum_{i=1}^{r} w_i U(\pi, \pi_i). \tag{7.25}$$

Then, according to Eq. (7.24), we have Eq. (7.23), which indicates that U is a KCC utility function. □

Remark Although Theorem 7.3 is still a sufficient condition, it extends our knowledge about KCC utility functions greatly by introducing a positive monotonic transformation g. Indeed, compared with Theorem 7.1, this sufficient condition is much closer to the necessary condition given in Eq. (7.23), with a stronger assumption that $\forall \Pi$, $\sum_{i=1}^{r} w_i U(\pi, \pi_i)$ and $\sum_{k=1}^{K} p_{k+} \phi(m_k)$ must have the same ranking to all π. More importantly, Eq. (7.24) can help to find the correspondence between ϕ and U, which enables the practical use of KCC. Some typical instances of g are as follows:

- $g(\Gamma(\pi, \Pi)) = a\Gamma(\pi, \Pi) + b$, where $a > 0$, b is a constant.

Table 7.3 The adjusted contingency matrix

		π_i				
		$C_1^{(i)}$	$C_2^{(i)}$	$\cdots$	$C_{K_i}^{(i)}$	$\sum$
π	C_1	$n_{11}^{(i)}$	$n_{12}^{(i)}$	$\cdots$	$n_{1K_i}^{(i)}$	$n_{1+}^{(i)}$
	C_2	$n_{21}^{(i)}$	$n_{22}^{(i)}$	$\cdots$	$n_{2K_i}^{(i)}$	$n_{2+}^{(i)}$
	$\cdot$	$\cdot$	$\cdot$	$\cdots$	$\cdot$	$\cdot$
	C_K	$n_{K1}^{(i)}$	$n_{K2}^{(i)}$	$\cdots$	$n_{KK_i}^{(i)}$	$n_{K+}^{(i)}$
	$\sum$	$n_{+1}^{(i)}$	$n_{+2}^{(i)}$	$\cdots$	$n_{+K_i}^{(i)}$	$n^{(i)}$

- $g(\Gamma(\pi, \Pi)) = \Gamma(\pi, \Pi)^q$, where q is a positive odd number.
- $g(\Gamma(\pi, \Pi)) = \ln \Gamma(\pi, \Pi)$.

While there are more choices about KCC utility functions after introducing g, from the application viewpoint, it is still a good way to take the simple affine transformation with $a = 1$ and $b = 0$, which results in Eq. (7.9) and the subsequent Eq. (7.18).

7.4 Handling Inconsistent Data

Here, we address the issue when applying K-means based consensus clustering to basic partitionings on inconsistent data.

Let $\mathscr{X} = \{x_1, x_2, \ldots, x_n\}$ denote a set of data objects. A basic partitioning π_i is obtained from the clustering of a data subset $\mathscr{X}_i \subseteq \mathscr{X}$, $1 \le i \le r$, with the constraint that $\bigcup_{i=1}^r \mathscr{X}_i = \mathscr{X}$. Here, the problem is, given r basic paritionings $\Pi = \{\pi_1, \ldots, \pi_r\}$ obtained from r inconsistent data subsets, how to adapt KCC for the clustering of $\mathscr{X}$ into K crisp clusters? Indeed, this problem is worthy of research from a practical viewpoint, since some data instances are often unavailable in a basic partitioning due to the reasons such as the geographical distribution or time delay of data.

We first adjust the way for utility computation with the presence of inconsistent data. We still have maximizing Eq. (7.1) as the objective of consensus clustering, but the contingency table for the computation of $U(\pi, \pi_i)$ should be modified carefully, as shown in Table 7.3.

In Table 7.3, $n_{k+}^{(i)}$ is the number of instances in $\mathscr{X}_i$ from cluster C_k, $1 \le k \le K$, and $n^{(i)}$ is the total number of instances in $\mathscr{X}_i$, i.e., $n^{(i)} = |\mathscr{X}_i|$, $1 \le i \le r$. Let $p_{kj}^{(i)} = n_{kj}^{(i)}/n^{(i)}$, $p_{k+}^{(i)} = n_{k+}^{(i)}/n^{(i)}$, $p_{+j}^{(i)} = n_{+j}^{(i)}/n^{(i)}$, and $p^{(i)} = n^{(i)}/n$.

We then adjust the K-means clustering on the binary data set $\mathscr{X}^{(b)}$ with the presence of inconsistent data. Assume the distance

$$f(x_l^{(b)}, m_k) = \sum_{i=1}^{r} f_i(x_{l,i}^{(b)}, m_{k,i}), \tag{7.26}$$

we obtain a new objective for K-means clustering as follows:

$$\min_{\mathscr{C}} L = \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_l^{(b)} \in \mathscr{C}_k \bigcap \mathscr{X}_i} f_i(x_{l,i}^{(b)}, m_{k,i}), \tag{7.27}$$

where the centroid

$$m_{k,ij} = \frac{\sum_{x_l^{(b)} \in \mathscr{C}_k \bigcap \mathscr{X}_i} x_{l,ij}^{(b)}}{|\mathscr{C}_k \bigcap \mathscr{X}_i|} = \frac{n_{kj}^{(i)}}{n_{k+}^{(i)}} = \frac{p_{kj}^{(i)}}{p_{k+}^{(i)}}, \quad 1 \le j \le K_i. \tag{7.28}$$

The two-phase iteration process of K-means turns into: (1) Assign $x_l^{(b)}$ to the cluster with the smallest distance f computed by Eq. (7.26); (2) Update the centroid of cluster C_k by Eq. (7.28). It is easy to note that the assigning phase will decrease L in Eq. (7.27) definitely. Moreover, since f_i is a point-to-centroid distance derived by a continuously differentiable convex function ϕ_i, we have $\forall y_k \neq m_k, L(y_k) - L(m_k) = \sum_{i=1}^{r} \sum_{k=1}^{K} n_{k+}^{(i)} f_i(m_{k,i}, y_{k,i}) \geq 0$, which indicates that the centroid-updating phase will also decrease L definitely. As a result, we guarantee that the solution of K-means will converge to a local minimum or a saddle point within finite numbers of iterations.

Based on the above adjustments, we now extend the K-means based consensus clustering to the inconsistent-data case. We have a theorem as follows:

Theorem 7.4 *A utility function U is a KCC utility function, if there exists a continuously differentiable convex function φ such that $\forall i$*

$$U(\pi, \pi_i) = p^{(i)} \sum_{k=1}^{K} p_{k+}^{(i)} \varphi(\langle p_{k1}^{(i)}/p_{k+}^{(i)}, \ldots, p_{kK_i}^{(i)}/p_{k+}^{(i)} \rangle). \tag{7.29}$$

Proof Let $m_{k,ij} = p_{kj}^{(i)}/p_{k+}^{(i)}$, and $\phi_i \doteq w_i \varphi$. On one hand, we have

$$\Gamma(\pi, \Pi) = \sum_{i=1}^{r} w_i U(\pi, \pi_i) = \sum_{i=1}^{r} p^{(i)} \sum_{k=1}^{K} p_{k+}^{(i)} \phi_i(m_{k,i}). \tag{7.30}$$

On the other hand, using ϕ_i, f_i in Eq. (7.27) can be:

$$f_i(x_{l,i}^{(b)}, m_{k,i}) = \phi_i(x_{l,i}^{(b)}) - \phi_i(m_{k,i}) - (x_{l,i}^{(b)} - m_{k,i})^T \nabla \phi_i(m_{k,i}). \tag{7.31}$$

Accordingly, we have

Table 7.4 Some characteristics of real-world data sets

Data	Source	#Objects	#Attributes	#Classes	MinClassSize	MaxClassSize	CV
breast_w	UCI	699	9	2	241	458	0.439
ecoli	UCI	336	7	8	2	143	1.160
iris	UCI	150	4	3	50	50	0.000
pendigits	UCI	10992	16	10	1055	1144	0.042
satimage	UCI	4435	36	6	415	1072	0.425
wine	UCI	178	13	3	48	71	0.194
sports	TREC	8580	126373	7	122	3412	1.022
tr45	TREC	690	8261	10	18	160	0.669

$$L = \underbrace{\sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{x_l^{(b)}\in\mathscr{C}_k\bigcap\mathscr{X}_i}\phi_i(x_{l,i}^{(b)})}_{(\alpha)} - n\sum_{i=1}^{r}p^{(i)}\sum_{k=1}^{K}p_{k+}^{(i)}\phi_i(m_{k,i}). \tag{7.32}$$

Since (α) and n are constants, according to Eq. (7.30), we have

$$\min_{\mathscr{C}} L \Leftrightarrow \max_{\mathscr{C}}\sum_{i=1}^{r}p^{(i)}\sum_{k=1}^{K}p_{k+}^{(i)}\phi_i(m_{k,i}) \Leftrightarrow \max_{\pi}\Gamma(\pi,\Pi). \tag{7.33}$$

The above implies that the maximization of the consensus function is equivalent to the clustering of the binary data, so U is a KCC utility function. □

Remark Eq. (7.29) is very similar to Eq. (7.18) except for the parameter $p^{(i)}$. This parameter implies that the basic partitioning with more data instances should have more impact on the consensus clustering, which is considered reasonable. Note that, when the inconsistent case reduces to the consistent one, Eq. (7.29) reduces to Eq. (7.18) naturally. This implies that the inconsistent case is a more general scenario. Finally, the KCC procedure for the inconsistent case is very similar to the consistent one; that is, given U, we first obtain φ, and then ϕ_i, f_i and f, and finally perform K-means clustering on the binary data to get π.

7.5 Experimental Results

In this section, we present experimental results of K-means based consensus clustering using various utility functions. Specifically, we demonstrate: (1) The convergence of KCC; (2) The cluster validity of KCC; (3) The impact of the generation strategies of basic partitionings on KCC; (4) The effectiveness of KCC in handling inconsistent data.

7.5.1 Experimental Setup

We first introduce the experimental setup, including the data information, the clustering tools, the validation measure, and the experimental environment.

Experimental data. In the experiments, we use a number of real-world data sets. Table 7.4 shows some important characteristics of these data sets, where CV is the distribution of class sizes measured by coefficient of variation [1].

Clustering tools. Two types of consensus clustering algorithms, namely the K-means based consensus clustering algorithm (KCC) and the graph partitioning algorithm (GP), have been tested in the experiments. GP is actually a general concept of three well-known benchmark algorithms: CSPA, HGPA and MCLA [16], which were implemented in MATLAB provided by Strehl.[1] We also implemented KCC in MATLAB, which includes ten utility functions, namely U_c, U_H, $U_{\cos}$, U_{L_5} and U_{L_8} and their normalized versions (denoted as NU).

To generate basic partitionings, we used the `kmeans` function[2] of MATLAB with squared Euclidean distance for UCI data sets, and the `vcluster` tool of CLUTO[3] with cosine similarity for text data. Three generation strategies [11, 16], i.e. Random Parameter Selection (RPS), Random Feature Selection (RFS), and Multiple Clustering Algorithms (MCA), are used. In RPS, we randomized the number of clusters for K-means in different basic clusterings. In RFS, we randomly selected partial features for clustering. Finally, in MCA, we used different clustering algorithms to produce diversified basic clusterings.

The default setting is as follows. The number of clusters K for KCC or GP is set to the number of true classes, and the weights of all basic partitionings are exactly the same, i.e. $w_i = 1, \ \forall\, i$. For UCI data, the RPS strategy is used, with the number of basic partitionings $r = 100$, and the number of clusters $K_i \in [K, \lceil\sqrt{n}\rceil], \ \forall\, i$. For text data, however, the MCA strategy is used instead, with 25 basic partitionings generated by the combination of five clustering methods (i.e. rb, rbr, direct, agglo and bagglo) and five objective functions (i.e. i2, i1, e1, g1, g1p) in CLUTO, and $K_i \in [K-2, K+2], \forall\, i$. We run KCC or GP 10 times for each data set to get the average result. Each time KCC calls K-means routine 10 times for the best result, and GP calls all the three algorithms and returns the best result.

Validation measure. Since we have class labels for all the real-world data sets, we use the external validation measures for cluster validity. In the literature, it has been recognized that the normalized Rand index (R_n) has merits in evaluating the K-means clustering [20]. The details of R_n can be found in [9]. Note that R_n is a positive measure with a wide value range.

Experimental environment. All the experiments were run on a Microsoft Windows 7 platform with SP2 32-bit edition. The PC has an Intel Core2 Duo T7250 2.0GHz×2 CPU of a 2MB cache, and a 3GB DDR2 332.5MHz RAM.

---

[1] http://www.strehl.com.

[2] http://www.mathworks.cn/help/toolbox/stats/kmeans.html.

[3] http://glaros.dtc.umn.edu/gkhome/views/cluto/.

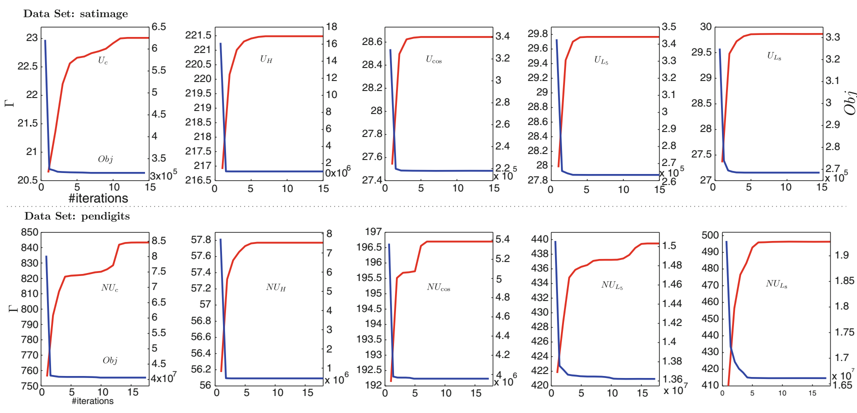


Fig. 7.1 The convergence of KCC using different utility functions

7.5.2 The Convergence of KCC

We first study the convergence property of KCC by observing the changing tendency of the consensus-function value during the clustering processes.

To this end, we employ KCC on the data sets in Table 7.4 using the default setting. Figure 7.1 shows some randomly selected runs of KCC on data sets `satimage` and `pendigits`, respectively. As can be seen, no matter what utility function is used, the value of the consensus function (Γ) keeps going up while the value of the objective function of K-means (obj) keeps going down, until a solution is found. Indeed, we find this trend ubiquitous for all $8 \times 10 \times 10 = 800$ runs of KCC. This implies that any U_φ derived by a continuously differentiable convex φ can be used for K-means based consensus clustering.

If we take a closer look at the convergence processes in Fig. 7.1, we can find that the convergence speed of KCC is satisfactory. Indeed, KCC usually achieves the largest portion of utility increase within 10 iterations. Furthermore, while having no theoretical evidence, we still find that U_H and NU_H often lead to faster convergence than other utility functions, in 6 out of 8 data sets for our case.

We also applied GP for the same groups of basic partitionings of all data sets. Figure 7.2 shows the comparative results of KCC (using U_c) and GP with respect to the average execution time. As can be seen, for seven data sets, KCC consumes obviously less time than GP. Note that (1) we do not have the result of GP on `pendigits` due to the out-of-memory failure caused by high K_i values, and (2) for `satimage` and `sports`, GP only calls two algorithms due to the large data scales. These imply that GP may not be a very good choice for large-scale, multiple-class data when a RPS setting is used.

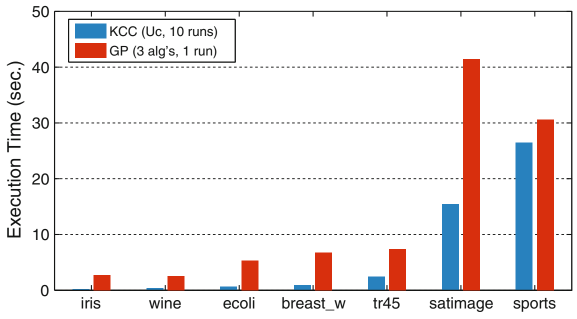


Fig. 7.2 KCC versus GP: the execution time

7.5.3 *The Cluster Validity of KCC*

In this section, we analyze the cluster validity of KCC in terms of both the quality, diversity and robustness.

Table 7.5 shows the clustering results of KCC and GP on the same groups of basic partitionings. As can be seen from Table 7.5, KCC using different utility functions shows higher clustering quality than GP in six out of eight data sets (indicated by the numbers in bold). For the two data sets, namely `breast_w` and `satimage`, the advantage of KCC is particularly evident. Nonetheless, it can be argued that the six winning cases of KCC should contribute to five utility functions. We can explain this from two aspects. On one hand, if we compare the clustering quality of GP and KCC with U_H, NU_H, or $U_{\cos}$, we can find that KCC still has at least five winning cases. This implies that KCC using a proper utility function can outperform GP. On the other hand, this result well illustrate that diversified utility functions are very important for the success of KCC. For instance, KCC with U_H produces a much better clustering result for `breast_w`, whereas KCC with NU_{L_8} is obviously a better choice for `sports`. Indeed, by incorporating different utility functions into a unified framework, KCC has more flexibilities in clustering data sets from different application domains.

Furthermore, the clustering result produced by KCC is very robust. To illustrate this, we plot the 10-run clustering results of KCC (with U_H) on the six UCI data sets, and the best basic-clustering results (with the least volatility among 100 partitionings in the 10 runs) are also plotted as references. Figure 7.3 shows the results. As can be seen from the figure, for each data set, KCC has a much narrower R_n range than the basic clustering, and also has a much higher average R_n value, except for the `wine` data set (we will address this in the next subsection). This indicates that although the results of basic clusterings may be drastically varying, KCC can still generate stable results. This robustness is particularly important for the real-world applications when external information of data is not available.

Finally, while it is hard to know which utility function is the best for a given data set, we can still provide some guidance for the practical use of KCC. We here attempt

Table 7.5 The clustering results of real-world data sets. (measured by R_n)

Data set	U_c	U_H	$U_{\cos}$	U_{L_5}	U_{L_8}	NU_c	NU_H	$NU_{\cos}$	NU_{L_5}	NU_{L_8}	GP
breast_w	0.390	**0.872**	0.505	0.135	0.135	0.065	0.854	0.131	0.124	0.124	0.497
ecoli	0.356	**0.377**	0.358	0.364	0.346	0.360	0.357	0.367	0.353	0.358	0.351
iris	0.746	0.735	0.746	0.746	0.746	0.702	0.737	0.746	0.746	0.746	**0.915**
pendigits	0.545	0.554	**0.591**	0.590	0.565	0.498	0.580	0.576	0.567	0.569	N/A[a]
satimage	0.338	0.490	0.494	0.484	0.482	0.292	**0.498**	0.454	0.432	0.385	0.385
wine[b]	0.144	0.140	0.144	0.137	0.137	0.146	0.138	0.145	0.145	0.143	**0.147**
sports	0.461	0.499	0.464	0.458	0.481	0.480	0.478	0.495	0.502	**0.510**	0.465
tr45	0.669	0.629	0.671	0.684	0.670	0.656	0.658	**0.688**	0.652	0.664	0.642
score	6.855	**7.758**	7.392	6.921	6.852	6.294	7.734	6.974	6.831	6.772	-

[a]Out of memory
[b]The last attribute is normalized by a scaling factor 100

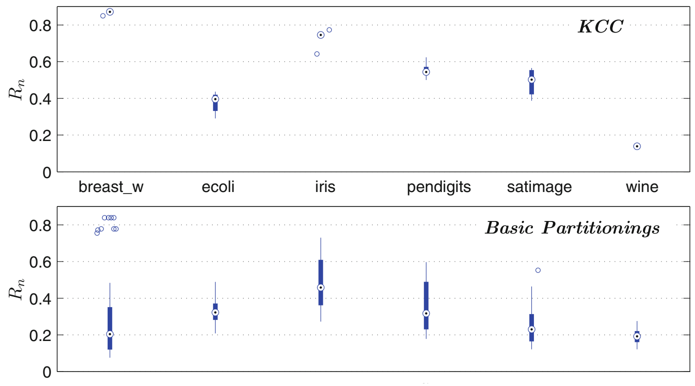


Fig. 7.3 The robustness of KCC

to rank the utility functions according to their performance on the eight data sets. We score a utility function U_i by $score(U_i) = \sum_j \frac{R_n(U_i, D_j)}{\max_i R_n(U_i, D_j)}$, where $R_n(U_i, D_j)$ is the R_n score of the clustering result generated by KCC using U_i on data set D_j. Table 7.5 lists the scores of all utility functions in the bottom row. Accordingly, the ranking of the ten utility functions is as follows: $U_H \succ NU_H \succ U_{\cos} \succ NU_{\cos} \succ U_{L_5} \succ U_c \succ U_{L_8} \succ NU_{L_5} \succ NU_{L_8} \succ NU_c$. Although this ranking is a bit coarse, we can still learn from the eight data sets that (1) U_H, NU_H, or $U_{\cos}$ is usually a good choice for KCC on unknown data, and (2) although it is the first utility function proposed in the literature, U_c and the normalized NU_c often work poorly in real-world applications.

7.5.4 The Comparison of the Generation Strategies of Basic Clusterings

In the previous section, we mainly used the RPS and MCA strategies to generate basic partitionings for consensus clustering. The RFS strategy, often considered as a limited alternative for using only partial data, has not been fully investigated. Therefore, in this subsection, we reveal the special effects of RFS for KCC.

First, RFS may help KCC to avoid the negative impact of noise or irrelevant attributes. Let us take the `wine` data set for illustration. As can be seen from Table 7.5 and Fig. 7.3, by using the RPS strategy, KCC produces clustering results even worse than the ones by the basic clustering. We now use RFS instead, with $r = 100$, $K_i = K$, and the number of randomly selected features $numF = 2$. Figure 7.4 shows the comparative results evaluated by R_n. As can be seen from the figure, by employing RFS, KCC with all utility functions improves the clustering quality dramatically! To explore the reasons, we first cluster `wine` by K-means and obtain a clustering quality $R_n = 0.134$. Then, we use the χ^2 method to rank the features, delete the five features with extremely low χ^2 values, and cluster the new data by K-means again. Now, we get a clustering accuracy $R_n = 0.631$. This result indicates that there exist some noise or irrelevant attributes in `wine` which seriously degrade the performances of KCC if RPS is used. Instead, RFS only selects partial attributes for one basic clustering, and thus can produce some high-quality basic partitionings, which may guide KCC to find a good consensus partitioning eventually.

Second, RFS can help KCC to find the true number of clusters K. Recall the normalized utility function $NU_H(\pi, \pi_i) = \frac{MI(\mathscr{C},\mathscr{C}')}{H(\mathscr{C}')}$. Let $\Gamma' = \sum_{i=1}^{r} w_i NU_H(\pi, \pi_i) \sqrt{\frac{H(\mathscr{C}')}{H(\mathscr{C})}}$. We use KCC with NU_H for consensus clustering, and observe the change of Γ' with respect to K, and search for the best ones. Figure 7.5 shows the comparative result using RPS and RFS, respectively. As can be seen, for data sets `breast_w` and `wine`, KCC using RFS obtains the highest Γ' values when K equals the true cluster numbers 2 and 3, respectively. For the `iris` data set, KCC using RFS obtains just a slightly smaller Γ' value when $K = 3$. In contrast, KCC using RPS obtains increasing Γ' values as K increases for all the three data sets, and therefore cannot identify the best K. Note that, for each K value, we repeat running KCC 10 times and return the average Γ' value for robustness.

7.5.5 The Effectiveness of KCC in Handling Inconsistent Data

In this subsection, we demonstrate the effectiveness of KCC in clustering inconsistent data. To illustrate this, we select two data sets with different sizes, `breast_w` and `satimage`, to simulate two cases with inconsistent data as follows.

The first is the "missing samples" case, which means that although we are given the basic parititionings $\Pi = \{\pi_1, \ldots, \pi_r\}$, part of the labels in π_i $(1 \leq i \leq r)$ are

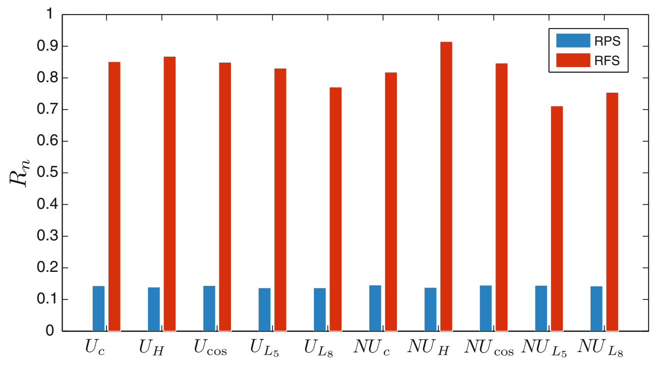


Fig. 7.4 A comparison of RFS and RPS on the `wine` data set

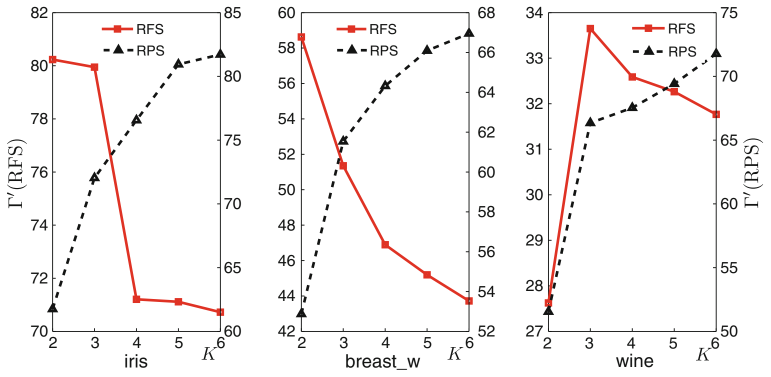


Fig. 7.5 A comparison of RFS and RPS in finding K

missing due to some unknown reasons. To simulate this, we do sampling on π_i with different ratios, and the unsampled labels are discarded as missing samples. The second is the "distributed samples" case, which means that the data is distributed so that each basic clustering is performed on different data subsets. To simulate this, we sample the original data, and perform basic clustering on the sampled subsets to obtain Π. For the two cases, the sampling ratio is set from 0.1 to 0.9, and KCC with U_H is run 10 times for average scores.

Figure 7.6 illustrates the clustering result. One observation is that, although the sampling ratio keeps going down from 0.9 to 1.0, KCC provides clustering results of surprisingly stable quality. This implies that KCC has the ability to organize "cluster fragments" into a complete picture. Another observation is that, KCC shows roughly the same clustering quality for two cases with inconsistent data. This is interesting, as we had thought that the "distributed samples" case would be much harder to deal with. This indicates that KCC can be employed for real-world distributed data clustering.

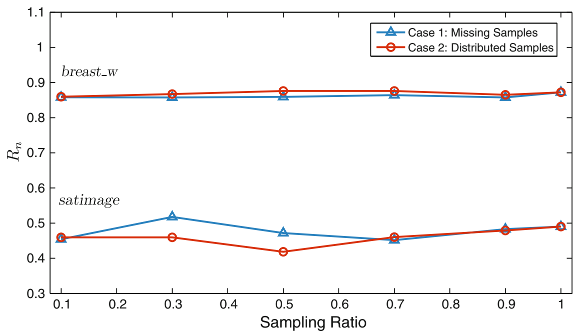


Fig. 7.6 The clustering result of KCC on inconsistent data

7.6 Related Work

Consensus clustering is often formulated as an NP-complete optimization problem. In the literature, many optimization objectives (explicit or implicit) and solutions have been proposed to do consensus clustering. In the pioneer work, [16] developed three graph-based algorithms, namely CSPA, HGPA and MCLA, for consensus clustering using Normalized Mutual Information (NMI). Another class of solutions is based on the similarity matrix. For instance, [6] summarized the results of basic clusterings in a co-association matrix, based on which the agglomerative hierarchical clustering [3, 17] was used to find the final clustering. [7] also proposed the Furthest Consensus algorithm to repeatedly find the furthest centers based on the similarity matrix. In the inspiring work, [18] proposed to use K-means for consensus clustering with Quadratic Mutual Information, and this elegant idea can be traced back to the work by Mirkin on the Category Utility Function [13]. They further extended their work to using the Expectation Maximization (EM) algorithm with a finite mixture of multinomial distributions for consensus clustering [19]. Other approaches for consensus clustering including relabeling and voting [2, 4], iterative voting [15], simulated annealing [11], weighting [10], etc. [8] provided a good comparison of various heuristic algorithms for consensus clustering. Different from the existing research efforts, in this chapter, we establish a general theoretic framework for K-means based consensus clustering using multiple utility functions derived from continuously differentiable convex functions.

7.7 Concluding Remarks

In this chapter, we studied the generalization issues of utility functions for K-means based consensus clustering (KCC). Specifically, we identified a sufficient condition, which indicates that utility functions that fit KCC can be derived from continuously differentiable convex functions. We therefore established a unified framework for KCC on both consistent and inconsistent data. Experiments on real-world data demonstrated the efficiency and effectiveness of KCC compared with the state-of-the-art method. In particular, KCC exhibits robustness on basic clusterings of varying qualities, and works surprisingly well for highly inconsistent data.

References

1. DeGroot, M., Schervish, M.: Probability and Statistics, 3rd ed. Addison Wesley, Reading (2001)
2. Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. Bioinformatics **19**(9), 1090–1099 (2003)
3. Fern, X., Brodley, C.: Random projection for high dimensional data clustering: A cluster ensemble approach. In: Proceedings of the 20th International Conference on Machine Learning, pp. 186–193. Washington, DC, USA (2003)
4. Fischer, R., Buhmann, J.: Path-based clustering for grouping of smooth curves and texture segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **25**(4), 513–518 (2003)
5. Fred, A., Jain, A.: Data clustering using evidence accumulation. In: Proceedings of the 16th International Conference on Pattern Recognition, vol. 4, pp. 276–280. Quebec City, Quebec, Canada (2002)
6. Fred, A., Jain, A.: Combining multiple clusterings using evidence accumulation. IEEE Trans. Pattern Anal. Mach. Intell. **27**(6), 835–850 (2005)
7. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. ACM Trans. Knowl. Discov. Data **1**(1), 1–30 (2007)
8. Goder, A., Filkov, V.: Consensus clustering algorithms: Comparison and refinement. In: Proceedings of the 9th SIAM Workshop on Algorithm Engineering and Experiments. San Francisco, USA (2008)
9. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
10. Li, T., Ding, C.: Weighted consensus clustering. In: Proceedings of the 8th SIAM International Conference on Data Mining, pp. 798–809. Atlanta, Georgia, USA (2008)
11. Lu, Z., Peng, Y., Xiao, J.: From comparing clusterings to combining clusterings. In: Fox, D., Gomes, C. (eds.) Proceedings of the 23rd AAAI Conference on Artificial Intelligence, pp. 361–370. AAAI Press, Chicago, Illinois, USA (2008)
12. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Cam L.L., Neyman J. (eds.) Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, Statistics. University of California Press (1967)
13. Mirkin, B.: Reinterpreting the category utility function. Mach. Learn. **45**(2), 219–228 (2001)
14. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. Mach. Learn. **52**(1–2), 91–118 (2003)
15. Nguyen, N., Caruana, R.: Consensus clusterings. In: Proceedings of the 7th IEEE International Conference on Data Mining, pp. 607–612. Washington, DC, USA (2007)
16. Strehl, A., Ghosh, J.: Cluster ensembles-a knowledge reuse framework for combining partitions. J. Mach. Learn. Res. **3**, 583–617 (2002)
17. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Boston (2005)
18. Topchy, A., Jain, A., Punch, W.: Combining multiple weak clusterings. In: Proceedings of the 3th IEEE International Conference on Data Mining, pp. 331–338. Melbourne, Florida, USA (2003)
19. Topchy, A., Jain, A., Punch, W.: A mixture model for clustering ensembles. In: Proceedings of the 4th SIAM International Conference on Data Mining, pp. 379–390. Florida, USA (2004)
20. Wu, J., Xiong, H., Chen, J.: Adapting the right measures for k-means clustering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 877–886. Paris, France (2009)

Glossary

Basic clustering The clustering performed for the purpose of generating a single data partition for the subsequent consensus clustering. See Chap. 7 and the term “consensus clustering”.

Class imbalance problem The challenge to data mining tasks where the data have multiple classes (or true clusters) in varying sizes. See Chaps. 2, 5, and 6.

Cluster analysis The data analysis task that attempts to partition data objects into multiple clusters (or groups) without using external information, such that objects in a cluster are more similar to each other than to objects in different clusters. See Chap. 1.

Cluster validity Using external or internal validation measures to evaluate clustering results in a quantitative and objective way. See Chap. 5 and the term “external validation measure”.

Consensus clustering Also known as cluster ensemble or clustering aggregation, an NP-complete combinatorial optimization problem that aims to find a single clustering from multi-source basic clusterings such that this single clustering matches all the basic clusterings as much as possible. See Chap. 7.

Data smoothing A technique that adds a small positive real number to all data objects such that the sparsity of high-dimensional data, such as text corpora, can be eliminated. See Chap. 4.

External validation measure A category of cluster validity measures that evaluates clustering results by comparing them to the true cluster structures defined by external information such as class labels. See Chap. 5.

Fuzzy c-means A type of prototype-based fuzzy clustering algorithms that acts like K-means clustering but allows a data object to belong to two or more clusters with a membership grade between zero and one. See Chap. 3 and the term “K-means”.

Information-theoretic K-means The K-means algorithm using the Kullback-Leibler divergence (KL-divergence) as the distance function. See Chap. 4 and the term "K-means".

K-means A type of prototype-based clustering algorithms that assigns data objects to closest clusters by computing the distances between the data objects and the centroids of the clusters. It can be also viewed as a special case of fuzzy c-means when the fuzzy factor tends to one. See Chap. 1 and the term "fuzz c-means".

Local clustering A data decomposition technique that performs clustering on a subset of data, e.g. the major class of data. See Chap. 6.

Measure normalization The issue that attempts to normalize the cluster validity measures into a small value range such as [0,1] or [−1,1], for the purpose of comparing clustering quality. See Chap. 5.

Point-to-centroid distance The only family of distance functions that fits directly K-means clustering with centroids of arithmetic means. See Chap. 3 and the term "K-means".

Rare class analysis The task of classification analysis on highly imbalanced data with the emphasis on identifying positive instances of rare classes. It plays a vital role in many important real-life applications, such as network intrusion detection, credit-card fraud detection, and facility fault detection. See Chap. 6.

Resampling A technique that draws randomly with or without replacement from the available data for generating a smaller (under-sampling) or a larger (over-sampling) subset of that data. See Chap. 6.

Spherical K-means The K-means algorithm using the cosine similarity as the proximity function. See Chap. 4 and the term "K-means".

Uniform effect The effect of K-means that tends to partition data objects into clusters in uniform sizes. This is a negative effect when applying K-means for class imbalance data. See Chap. 2 and the term "K-means".

Variable neighborhood search An optimization meta-heuristic which exploits systematically the idea of neighborhood change, both in the descent to local minima and in the escape from the valleys that contain them. See Chap. 4.

Zero-value dilemma The problem in computing KL-divergence between data objects and centroids when there exist zero-value features in the high-dimensional feature space. See Chap. 4.