Robust models and novel similarity measures for high-dimensional data clustering

Nguyen, Duc Thang

2012

Nguyen, D. T. (2012). Robust models and novel similarity measures for high-dimensional data clustering. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/48657>

<https://doi.org/10.32657/10356/48657>

**ROBUST MODELS AND
NOVEL SIMILARITY MEASURES FOR
HIGH-DIMENSIONAL DATA CLUSTERING**

NGUYEN DUC THANG

School of Electrical & Electronic Engineering

A thesis submitted to Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2012

Acknowledgments

First and foremost, I wish to express my deep gratitude to the Division of Information Engineering, School of Electrical and Electronic Engineering, Nanyang Technological University, who has made my Ph.D. journey feasible in the first place. I am grateful to have been granted the research scholarship by the school.

I am very thankful to my supervisors, Dr. Chen Lihui and Dr. Chan Chee Keong, for all the time and effort they have been giving me during my entire Ph.D. journey. Their opinions, ideas and numerous useful insights have been so valuable. Dr. Chen and Dr. Chan have provided great help to enrich my knowledge and improve the quality of my research. All the meetings with them have been very enjoyable, interesting and beneficial. I hope they will continue to give me their many advices and supports in the future.

Special thanks to Mrs. Leow-How and Christina in Software Engineering Lab for being so helpful to create a very nice research environment in the lab. I would like to thank Mei Jianping and Yan Yang for their friendship and the useful discussions we have had.

I would like to reserve my final appreciation to the most precious person in my life, my beloved wife Rose. She has been my motivator since day one, continuously giving me supports and encouragements. She has always been there with me, during my happy moments as well as in my toughest time. Her care, love and companionship have been incredibly important to me. No word can describe my love for her.

Contents

Acknowledgments	i
Contents	vi
Summary	viii
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Overview	1
1.2 Motivation	2
1.3 Contributions	4
1.4 Thesis Outline	5
2 Research Background	7
2.1 Overview	7
2.2 Recent Developments in Clustering	8
2.2.1 k -means and Extensions	8
2.2.2 Self-Organizing Feature Mapping	11
2.2.3 Fuzzy Clustering	13
2.2.4 Non-negative Matrix Factorization	14
2.2.5 Spectral Clustering	15
2.2.6 Search-based Clustering	17
2.2.7 Mixture Model-based Clustering	17
2.3 Existing Problems and Potential Solution Approaches	21
2.3.1 The Curse of Dimensionality	21
2.3.2 The Number of Clusters	23
2.3.3 Initialization Problem	24
2.3.4 Outlier Detection	25

2.4	Text Document Clustering	26
2.4.1	Applications to Web Mining & Information Retrieval . .	27
2.4.2	Text Document Representations	28
2.5	Document Datasets	32
2.6	Evaluation Metrics	36
3	Mixture Model-based Approach: Analysis & Efficient Techniques	39
3.1	Overview	39
3.2	Mixture Models of Probabilistic Distributions	42
3.2.1	Mixture of Gaussian Distributions	42
3.2.2	Mixture of Multinomial Distributions	43
3.2.3	Mixture of von Mises-Fisher Distributions	43
3.3	Comparisons of Clustering Algorithms	44
3.3.1	Algorithms for Comparison	44
3.3.2	Experimental Results	45
3.4	The Impacts of High Dimensionality	47
3.4.1	On Model Selection	47
3.4.2	On Soft-Assignment Characteristic	51
3.4.3	On Initialization Problem	52
3.5	MMDD Feature Reduction	54
3.5.1	The Proposed Technique	54
3.5.2	Experimental Results	55
3.6	Enhanced EM Initialization for Gaussian Model-based Clustering	58
3.6.1	DA Approach for Model-based Clustering	58
3.6.2	The Proposed EM Algorithm	60
3.6.3	Experimental Results	61
3.7	Conclusions	65
4	Robust Mixture Model-based Clustering with Genetic Algorithm Approach	68
4.1	Overview	68
4.2	M2C and Outliers	70
4.2.1	Classical M2C	70
4.2.2	Toward Robustness in M2C	72
4.3	GA-based Partial M2C	74
4.4	Empirical Study	78
4.4.1	Parameter Setting	78

4.4.2	Continue Experiment 4.2.1	79
4.4.3	Mixture of Five Bivariate Gaussians with Outliers	81
4.4.4	Simulated Data in Higher Dimensions	84
4.4.5	Bushfire Data	86
4.4.6	Classification of Breast Cancer Data	87
4.4.7	Running Time	89
4.5	Conclusions	89
5	Multi-Viewpoint based Similarity Measure and Clustering	
	Criterion Functions	91
5.1	Overview	91
5.2	Related Work	93
5.3	Multi-Viewpoint based Similarity	97
5.3.1	Our Novel Similarity Measure	97
5.3.2	Analysis and Practical Examples of MVS	98
5.4	Multi-Viewpoint based Clustering	102
5.4.1	Two Clustering Criterion Functions I_R and I_V	102
5.4.2	Optimization Algorithm and Complexity	107
5.5	Performance Evaluation of MVSC	108
5.5.1	Experimental Setup and Evaluation	109
5.5.2	Experimental Results	110
5.5.3	Effect of α on MVSC- I_R 's performance	113
5.6	MVSC as Refinement for k -means	115
5.6.1	Introduction	115
5.6.2	Experimental Setup	115
5.6.3	Experimental Results	116
5.7	Conclusions	119
6	Applications	120
6.1	Collecting Meaningful English Tweets	120
6.1.1	Introduction to Sentiment Analysis	120
6.1.2	Applying GA-PM2C to Differentiate English from Non-English Tweets	122
6.2	Web Search Result Clustering with MVSC	125
6.2.1	Overview of Web Search Result Clustering	125
6.2.2	Integration of MVSC into Carrot ² Search Result Clustering Engine	129

7	Conclusions	136
7.1	Summary of Research	136
7.2	Future Work	138
	 Author's Publications	 140
	 Bibliography	 155

Summary

In this thesis, we present our research works on some of the fundamental issues encountered in high-dimensional data clustering. We examine how statistics, machine learning and meta-heuristics techniques can be used to improve existing models or develop novel methods for the unsupervised learning of high-dimensional data. Our objective is to achieve multiple key performance characteristics in the methods that we propose: reflecting the natural properties of high-dimensional data, robust to outliers and less sensitive to initialization, effective and efficient methods which are simple, fast, highly applicable and, on the other hand, produce good quality clustering results.

Mixture Model-based Clustering, or M2C, is a clustering approach that has very strong foundation in probability and statistics. Among all the possible models, Gaussian mixture is the most widely used. However, when applied for very high-dimensional data such as text documents, it expresses a few disadvantages that do not exist in low-dimensional space. To explore and understand this matter thoroughly, an analysis of the impacts of high dimensionality to various aspects related to Gaussian M2C has been conducted. We propose an enhanced Expectation Maximization algorithm to help the Gaussian M2C go through the initialization stage more properly. Other than that, von Mises-Fisher is a kind of distribution coming from Directional Statistics and has recently been known as a suitable model for document data. Our application of the von Mises-Fisher distribution mixture as a Feature Reduction method shows interesting results in the document clustering problem. Experiments carried out on benchmarked document datasets confirm the performance improvements offered by the proposed methods.

With the thesis, we also propose and present a novel clustering framework and the related algorithm to address the issue of clustering data with noise and outliers. The framework is called Partial Mixture Model-based Clustering, or PM2C. While the classical M2C framework does not take noisy data and outliers into consideration, the new framework is aware of the existence of these elements,

and provides a solution to address the issue. In a particular implementation designed following this framework, we propose the GA-PM2C algorithm. By incorporating the robust searching capability of Genetic Algorithm (GA) into the original M2C, we enable the new model to handle noise and outliers in data. The algorithm is capable of accurately differentiating clustered data from noise and outliers, and hence producing quality clustering results. Through our experiments and analysis on simulated and real datasets, the advantages of PA-PM2C compared with the classical M2C approach are demonstrated. We also showcase an application scenario in real-life social media data mining problem, in which PA-PM2C helps to fulfill the clustering task properly.

In clustering methodology, discriminative approach is the other side of the coin compared with the generative approach discussed above. Without assuming any underlying probabilistic distributions, discriminative methods are built by optimizing some objective functions of either error measures or quality measures. To formulate these clustering criterion functions, they often define certain similarity or dissimilarity measures among data objects. There is an implicit assumption that the data’s intrinsic structure can be approximated by these predefined measures.

However, in the current data clustering field, there is still a need for more appropriate and accurate similarity measures. In an effort to address this issue, we propose MVS- a Multi-Viewpoint based Similarity measure for text document data. As its name reflects, the novelty of our proposal is the concept of measuring similarity from multiple different viewpoints, rather than from just one origin point like in the case of cosine measure. Subsequently, we apply MVS to formulate two new criterion functions, called I_R and I_V , and introduce MVS-based Clustering, or MVSC. The major advantages of our algorithms are that they can be easily applicable like k -means or similar algorithms, but at the same time provide better clustering quality. Extensive experiments on a large number of document collections are presented to support these claims. Furthermore, we also implement MVSC into an actual, real-world web search and clustering system. The demonstration shows how effective and efficient MVSC is for practical clustering applications.

List of Figures

2.1	A snapshot of search engine WebClust	27
3.1	Fitting an overlapping Gaussian mixture	49
3.2	An example of bad initialization	53
3.3	Clustering results of dataset <i>reuters10</i>	56
3.4	Clustering results of dataset <i>fbis</i>	56
3.5	Clustering results of dataset <i>tr45</i>	57
3.6	Clustering results of dataset <i>webkb4</i>	57
3.7	Enhanced EM for spherical Gaussian model-based clustering . .	62
3.8	Clustering results in <i>Purity</i>	63
3.9	Clustering results in <i>NMI</i> on datasets <i>tr23</i> and <i>tr45</i>	65
4.1	Classical Gaussian M2C on normal dataset and contaminated dataset	71
4.2	Partial mixture model-based clustering.	73
4.3	Algorithm: GA-PM2C	76
4.4	Procedure: Guided Mutation	77
4.5	GA-PM2C fits with ϵ at: 0.15, 0.25, 0.35 & 0.45	80
4.6	GA-PM2C and FAST-TLE fits with ϵ at 0.03 & 0.04	83
4.7	An example of Recombination in GA-PM2C.	84
4.8	Classification performance at different trimming rates	88
4.9	Running time on datasets A and B	90
5.1	Procedure: Build MVS similarity matrix.	100
5.2	Procedure: Get validity score.	101
5.3	Characteristics of <i>reuters7</i> and <i>k1b</i> datasets.	102
5.4	Validity test on <i>reuters10</i> and <i>k1b</i>	102
5.5	Validity test on <i>tr31</i> and <i>reviews</i>	103
5.6	Validity test on <i>la12</i> and <i>sports</i>	103
5.7	Validity test on <i>tr12</i> and <i>tr23</i>	104
5.8	Algorithm: Incremental clustering.	108

5.9	Clustering results in <i>Accuracy</i>	110
5.10	MVSC- I_R 's performance with respect to α	114
5.11	<i>Accuracies</i> on the 50 test sets	119
6.1	Twitter Sentiment from a Stanford academic project.	121
6.2	Twitter sentiment analysis.	122
6.3	A snapshot of tweet clustering result by GA-PM2C algorithm.	124
6.4	Examples of tweets classified differently by GA-PM2C & <i>Spkmeans</i>	126
6.5	Web search and clustering.	127
6.6	A screenshot of Carrot ² 's GUI.	129
6.7	Clusters with topic labels recommended for query "apple".	132
6.8	Clusters with representative snippets.	133
6.9	MVSC2's clusters visualized by Carrot ²	134

List of Tables

2.1	Document datasets I	34
2.2	Document datasets II	35
2.3	Document datasets III	35
3.1	Clustering result comparison I	45
3.2	Clustering result comparison II (based on NMI values)	47
3.3	Characteristics of <i>Iris</i> and <i>classic3</i> data	49
3.4	Values for <i>Iris</i> and <i>classic3</i> data	50
3.5	The highest posterior probabilities of the first few objects in ascending order and clustering purities	51
3.6	Changes in posterior probabilities of a randomly selected document object in <i>5Newsgroups</i> during EM	53
3.7	Comparison between clustering results with and without M2FR technique	58
3.8	<i>NMI</i> results & clustering time by 3 Gaussian models	63
3.9	<i>NMI</i> results: Gaussian models compared with CLUTO and other probabilistic models	64
4.1	Confusion matrices resulted from classical Gaussian M2C	72
4.2	Log-likelihood and success rates over 100 repetitions with $ P = 4$	79
4.3	Confusion matrices resulted from GA-PM2C with $\epsilon = 0.35$	79
4.4	5-component Gaussian mixture with outliers	81
4.5	Success rates over 100 repetitions for dataset in Table 4.4	82
4.6	Datasets A and B	85
4.7	Success rates over 100 Monte Carlo samples for datasets A and B	85
4.8	Cluster assignments with $k=3$ for Bushfire data	86
4.9	Classification error rate (%) for Wisconsin data	87
5.1	Notations	93
5.2	Clustering results in <i>FScore</i>	111
5.3	Clustering results in <i>NMI</i>	112

5.4	Statistical significance of comparisons based on paired t -tests with 5% significance level	113
5.5	Clustering results on <i>TDT2</i>	117
5.6	Clustering results on <i>Reuters-21578</i>	118
6.1	Clustering time (in second).	134

Chapter 1

Introduction

1.1 Overview

Organizing information into meaningful groupings is one of the most fundamental activities that we can encounter in daily life. For examples, you may split emails in your company email folder according to discussion topics; you may separate and label the documents on your desk based on the projects they are created for; you may also categorize the entries in your online blogs, through tagging, by the contents that you have written, and so on. Data clustering, or Cluster analysis, is a field of research which focuses on concepts and methodologies used for grouping (i.e. clustering) data objects. The purpose of data clustering process is to discover natural and intrinsic groupings of similar objects. Clustering does not use category labels to learn data. Clustering algorithms will categorize a collection of data objects into clusters, or groups, without any prior label information, such that the objects in the same cluster are most similar to each other, while at the same time are also dissimilar to those in other clusters. This type of knowledge discovery is called unsupervised learning, and different from supervised learning such as classification, which does involve label information to train the classifier.

Nowadays, data clustering techniques have been applied everywhere, whether in scientific research projects or in practical industrial applications. They are usually used as part of a decision making process that involves analyzing multivariate data. Some popular application scenarios of clustering are: image segmentation- an important topic in computer vision with many useful applications such as medical image examination, hand-writing recognition or satellite image analysis; information retrieval- text documents can be categorized into groups of topics, organized and summarized for pre-query as well as post-

query; market segmentation and analysis- products or customers are clustered for strategic decision making according to their characteristics and transaction data; bioinformatics- clustering techniques are applied on microarray gene data to discover new protein structures or functionally related groups of genes. These few examples show the huge benefits that clustering can potentially offer.

Recently, there is a new version of Moore's law proposed by Annalee Newitz from AlterNet¹: "The amount of information in the world is always expanding faster than the data storage systems available to capture it." And this is exactly true. You may have found out by your own experience that the larger the capacity of your personal laptop's hard disk becomes, the even more you want it to be to store all your data. And just how fast and big data storage systems are today? In a study appearing on February 10 in the journal *Science Express*, the researchers announced that humankind could store, in both digital memory and analog devices, at least 295 exabytes of information.

We are truly living in the age of information. If talking about the amount of information that is available on online websites alone, we can estimate that the total number of web pages out there is in the order of tens of billion, although there is currently not any official figure. Besides traditional web pages, we also have emails, books, Twitter, Facebook and so on. Most of these sources are in the form of unstructured text. Document clustering- or text clustering, a specific area within the data clustering field that we have introduced above- is the tool for us to categorize and organize this information automatically and efficiently. One characteristic of document data is that they are often of very high dimension (the number of words are huge), and also very sparse (each document only contains a very small portion of the total word vocabulary). Other types of data, such as microarray genes, also express this property at different levels. The main theme of this thesis is about novel concepts and techniques that are applied for clustering this kind of sparse and high-dimensional data.

1.2 Motivation

Despite the fact that data clustering has been an ongoing research field for a few decades, it still remains an interesting and challenging task to develop a good clustering algorithm due to many facets of the problem. There are so many unsolved issues exist in data clustering that it motivates us to focus our research work on this field. Some of them are short-listed below:

¹<http://www.alternet.org>

- Discriminative and Generative: These are two different approaches to clustering. The generative approach assumes data are generated from some class conditional probability density models. On the other hand, in discriminative approach, clustering methods are formulated without such assumption of specific probabilistic distributions, but with some functions of error measure or similarity measure to optimize. Researchers have been divided into two groups, each of which favors one approach over the other. We are interested in looking into both directions, and hopeful to find a good combination of the two approaches which benefits from all of their strong features.
- Stability of clustering solution: Many clustering algorithms are very sensitive to their initialization state. Their solutions can become greatly different with different initialized values. In practical applications, this is not desirable since we would prefer a stable system. Is there a method to reduce the sensitiveness to initialization of the existing algorithms? Or can we design an algorithm that performs consistently in many cases?
- Measure of Similarity: Any clustering algorithm holds a certain perception of similarity between two data objects. It is because the very definition of clustering is to divide a set of objects into groups of “similar” objects. In low-dimension space, it can be reliably done by measures such as Euclidean distance. However, in a high-dimensional space, the curse of dimensionality makes it difficult to have a proper measure. The true intrinsic structure of data becomes more tricky to study. In the case of document clustering, cosine of feature vectors has been used widely. Nevertheless, there is still a need to find better, more robust and satisfying measures.
- Effects of outliers and noisy data: Impurity in data is a undesirable but unavoidable fact. In the presence of noise and outliers, an algorithm that does not take these elements into consideration may produce inaccurate clustering results. Many existing clustering algorithms have overlooked the outlier effects. The basic question is whether we have to label every single object in a given dataset? Is it sensible, for example, to have a clustering framework that includes these elements as a separate set of data, apart from the normal data that need to be clustered?
- Scalability and speed: We could imagine for now that there will never be a limit on the volume of data that we have to handle. In general, we

would prefer clustering algorithms to be as less computational demanding as possible, and to be scalable with the size and dimension of input data. Scalability and speed are often related factors, although there can be different performance requirements for a clustering algorithm depending on its particular use. In the case of web search result clustering, for example, maybe only a portion of the search results need to be processed but the return time has to be as fast as possible, perhaps less than a fraction of a second. Our objective is also to develop an algorithm that is both effective in quality and efficient in computation.

- Usability: Clustering algorithms are often used as one part of a bigger process in an entire system. The ability to easily adapt and integrate an algorithm into various application scenarios is an important advantage. As we have known, k -means is a very fundamental clustering algorithm, but it is one of the most widely used algorithms for so many years since it was introduced. The reason is its simplicity and generalness. Many other existing algorithms are formulated in a way specific to some particular domains, and hence difficult to be implemented in other situations.

There are obviously many other challenges to achieve a good clustering algorithm that we have not mentioned here. In spite of that, the potential benefits of data clustering techniques are huge. With our research, we hope to be able to shorten the gap between these difficult challenges and the useful, practical applications of clustering.

1.3 Contributions

Through our research work, we have a few contributions to the data clustering field. They are declared briefly as follows:

- A comprehensive review of various related clustering methods proposed and developed recently. Besides discussing the distinctive features of these methods, we also point out some important issues existing in clustering field that need to be paid attention to.
- A critical analysis and experimental study of probabilistic mixture model-based clustering approaches. We give a few clear examples and explain why some algorithms fail on high-dimensional data.

- Two techniques for improving the performance of mixture model-based algorithms. One technique uses an enhanced-Expectation Maximization (EM) algorithm to reduce the sensitiveness of Gaussian mixture model to initialization, while the other applies a mixture of the directional distribution von Mises-Fisher to perform Feature Reduction.
- A general framework, known as Partial Mixture Model-based Clustering (PM2C), for data clustering in the presence of outliers. From this framework, we propose a novel algorithm called GA-PM2C, which is a combination of Genetic Algorithm (GA), with a new concept and customized operation, and the probabilistic mixture model-based clustering.
- A novel concept of similarity measure between sparse and high-dimensional feature vectors, called Multi-Viewpoint based Similarity (MVS). Based on this proposed measure, we formulate two new clustering criterion functions (I_R & I_V) and their algorithms (MVSC- I_R & MVSC- I_V), which then shows good performance in document clustering problems.
- A study of clustering's application to ordinary web mining problems. Two use cases of our proposed algorithms are demonstrated. In one case, one of our algorithms is applied on Twitter data, and helps to differentiate English and non-English tweets. In another case, one of our algorithms is used to cluster web search results retrieved from popular web search engines. The algorithm helps to categorize web pages and organize them into meaningful topics.

1.4 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2, we give the literature review of the data clustering field. It includes recent clustering algorithms and current problems often encountered in the field. Chapter 3 contains an analytical study of probabilistic mixture model-based clustering. We give a critical review, examine and compare different algorithms. The impacts of high dimensionality of data is discussed. Two methods: the Feature Reduction method using mixture of von Mises Fisher distributions, and the enhanced-EM algorithm for reducing Gaussian mixture's sensitiveness to initialization, are also presented in this chapter. Next, in Chapter 4, the Partial M2C framework is introduced together with the algorithm GA-PM2C whose objective is to cluster data in the presence of outliers. It is followed by Chapter 5 in which the MVS measure is

proposed, and so are its resulted clustering algorithms $MVSC-I_R$ and $MVSC-I_V$. Chapter 6 is a chapter of applications, showing how the proposed clustering algorithms are used in real-life problems. Finally, we conclude and summarize the thesis by Chapter 7.

Chapter 2

Research Background

2.1 Overview

In this chapter, we review some important background knowledge in the field of data clustering. A summary of various clustering algorithms which have been recently proposed is presented in Section 2.2. Depending on the techniques used or the characteristics of the data partition resulted from the clustering process, the algorithms can be divided into different categories. Subsequently, in Section 2.3, we identify some critical problems that researchers have found encountered when working on high-dimensional data clustering. The issues described in this section are the inspiration and the basis from which the research works in this thesis have been developed. Then, with Section 2.4, we highlight a few important applications of clustering in various domains, including web search, information retrieval, genetic microarray data analysis and image segmentation. Another aspect that we would like to review is the presentation of data before they are fed into a clustering algorithm. Domain-specific applications and specialized algorithms require the input data to be preprocessed and presented in proper ways in order to achieve desirable results. This topic is also part of Section 2.4. From next chapter onwards, there are a series of experiments that we carry out and present in this thesis to evaluate the clustering algorithms considered in our study. Majority of the experiments are on text document data. Therefore, we make use of Section 2.5 to summarize the document datasets that are used mainly across our entire study. They are also popular datasets in the clustering literature. Besides test data, evaluation metrics are the important tools to measure the performance of clustering methods. Section 2.6 presents a group of well-known metrics that are employed throughout the experiments.

2.2 Recent Developments in Clustering

There have been plenty of review and survey papers on the topic of data clustering algorithms. Some of the typical examples in the field are [1–4]. In this section, we focus on recent developments in the area of high-dimensional data clustering. Among them is the mixture model-based approach that we have been studying extensively.

2.2.1 k -means and Extensions

Perhaps it would not be inappropriate to say that k -means is the most well-known algorithm not only in data clustering but the whole Data Mining field. The algorithm is simple, fast and yet powerful [5]. More than half a century has passed since its introduction, until today k -means is still regarded as one of the top 10 data mining algorithms [6]. Therefore, we feel obliged to review it here. Basically, the idea is to find a set of vectors $\mathbf{c}_m$ ($m = 1, \dots, k$) that minimize the sum of squared error (SSE) objective function:

$$e^2 = \sum_{m=1}^k \sum_{i=1}^N \delta_{im} \|\mathbf{x}_i - \mathbf{c}_m\|^2 \quad (2.1)$$

where k is the number of data clusters and must be predefined. Vector $\mathbf{c}_m$ represents the center of cluster m . If data object x_i ($i = 1, \dots, N$) is assigned to cluster m , it will be $\delta_{im} = 1$. Otherwise, $\delta_{im} = 0$. Despite its popularity, original k -means performs poorly on text data. The reason is that it uses distance measure, such as Euclidean-norm in function 2.1, which is ineffective in high-dimensional space. This will be mentioned more in section 2.3, when various opening problems and existing challenges in text clustering research are discussed.

Various extended versions of k -means had been proposed to overcome this problem. In [7], Dhillon and Modha introduced Spherical k -means algorithm. Its framework is the same as the original k -means, but instead of Euclidean distance, cosine similarity is used. And instead of minimizing, we maximize the objective function:

$$f = \sum_{m=1}^k \sum_{i=1}^N \delta_{im} \mathbf{x}_i^T \mathbf{c}_m \quad (2.2)$$

in which all the document vectors $\mathbf{x}_i$ and “concept vectors” $\mathbf{c}_m$, as named by the authors, are normalized to unit length. They argue that when the dimension is

too high, direction is more important than distance. And thereby cosine similarity is more effective than Euclidean distance. The algorithm can be summarized as following:

1. *Initialization*: All the document vectors are normalized to unit length, and randomly partitioned into k groups. Given a group of vectors, it can be proven that the group's mean itself has the maximum sum of cosine similarities to all the elements in the group. Hence, concept vector of cluster m can be determined as:

$$\mathbf{c}_m = \frac{\sum_{i=1}^N \delta_{im} \mathbf{x}_i}{\left\| \sum_{i=1}^N \delta_{im} \mathbf{x}_i \right\|}, \forall m = 1, \dots, k \quad (2.3)$$

2. *Re-assignment*: Calculate cosine similarities of each document vector $\mathbf{x}_i$ to all the concept vectors, then re-locate it to the cluster with the closest concept vector, i.e.:

$$\delta_{im} = 1 \iff \mathbf{x}_i^T \mathbf{c}_m \geq \mathbf{x}_i^T \mathbf{c}_l, \forall l \neq m, 1 \leq l \leq k$$

with the constraint that $\sum_m \delta_{im} = 1, \forall i = 1, \dots, N$. So if a document happens to be closest to more than one cluster, it can be assigned to any one, and only one, of these.

3. *Re-defining concept vector*: Based on the new partitions, concept vectors are re-calculated according to equation 2.3.
4. The steps of re-assigning document vectors into clusters, and redefining concept vectors are repeated until no further changes are made.

Another variant of k -means, which has shown to perform well as text clustering algorithm, is Bisecting k -means [8]. It is in a way similar to divisive hierarchical clustering, starting from one partition containing the entire data, then subsequently dividing it into the desired number of clusters, as following:

1. Select one of the partition based on some criterion for splitting.
2. For a predefined number of times, split the selected partition into two sub-groups, using a clustering algorithm such as Spherical k -means. Among the results, select the pair of sub-groups with the highest similarity measure.
3. Repeat the steps above until the number of clusters reaches k .

The criterion for selecting which partition to split is subjective and application-dependent. One possible choice is to choose cluster with the largest population.

More recently, a version of k -means incorporating feature weighting was proposed in [9]. It is named Feature Weighting k -Means, or FW-KMeans. The technique can be categorized into subspace clustering, an approach to clustering high-dimensional data where different classes are considered under different sub-groups of dimensional space. Subspace clustering will be explored in more details in a later part of the report, when we address alternative solutions to text classification problem. In FW-KMeans, features of a data vector are weighted based on its importance toward the cluster that data object is supposed to belong to. Basically, the main idea is to minimize the objective function:

$$f(W, Z, \Lambda) = \sum_{m=1}^k \sum_{i=1}^n \sum_{j=1}^d w_{im} \lambda_{jm}^\beta D(z_{mj}, x_{ij}) \quad (2.4)$$

In the equation, k is number of classes, n number of data objects and d number of features. $D(z_{mj}, x_{ij})$ is some dissimilarity measure between cluster center z_m and object x_i with respect to feature j . In [9], the authors used Euclidean distance. Binary values w_{im} represents belongingness of object i to cluster m like in standard k -means. Additionally, variable λ_{jm} is introduced as weighting factor of feature j to cluster m , whereas β is some given constant greater than 1. The optimization procedure is carried out with matrices $W = \{w_{im}\}$ and $Z = [z_1 \dots z_k]$ updated just like in standard k -means algorithm, and one extra step to update parameter λ :

$$\lambda_{jm} = \frac{1}{\sum_{t=1}^d \left[\frac{\sum_{i=1}^n \hat{w}_{im} D(\hat{z}_{mj}, x_{it})}{\sum_{i=1}^n \hat{w}_{im} D(\hat{z}_{mt}, x_{it})} \right]^{1/(\beta-1)}} \quad (2.5)$$

where $\hat{w}_{im}$ and $\hat{z}_{mj}$ are updated values from previous iterations. Unfortunately, this model is not designed to appropriately handle sparsity, as happened in the case of text data. Hence, in [10], the authors improved the FW-KMeans algorithm to perform document clustering. It is done simply by adding a constant parameter σ into the dissimilarity measure as following:

$$f(W, Z, \Lambda) = \sum_{m=1}^k \sum_{i=1}^n \sum_{j=1}^d w_{im} \lambda_{jm}^\beta [D(z_{mj}, x_{ij}) + \sigma] \quad (2.6)$$

Consequently, the updating equation of λ in (2.5) becomes:

$$\lambda_{jm} = \frac{1}{\sum_{t=1}^d \left[\frac{\sum_{i=1}^n \hat{w}_{im} [D(\hat{z}_{mj}, x_{ij}) + \sigma]}{\sum_{i=1}^n \hat{w}_{im} [D(\hat{z}_{mt}, x_{it}) + \sigma]} \right]^{1/(\beta-1)}} \quad (2.7)$$

When a word j does not exist in any documents of a cluster, or if it is present with the same frequency in all the documents, it is potential that $D(z_{mj}, x_{ij})$ tends zero, thereby λ_{jm} becomes infinity. The constant σ help preventing this situation. Empirical results in [10] shows effectiveness of FW-KMeans compared to standard and Bisecting k -means on text clustering.

A recently developed extension of Spherical k -means, which aims to speed up clustering for real-time applications, is the online spherical k -means [11]. Unlike Spherical k -means or the other algorithms, which process the entire dataset in batch mode, this is an online competitive learning scheme in which document objects are streamed into the data collection continuously. As they are added, the objects are assigned to their closest cluster, and the cluster that gets assignment adjusts its centroid vector according to a learning rate η . Given $\mathbf{x}_i$ be assigned to cluster m , i.e. $\delta_{im} = 1$, the update equation is:

$$\mathbf{c}_m^{new} = \frac{\mathbf{c}_m + \eta \mathbf{x}_i}{\|\mathbf{c}_m + \eta \mathbf{x}_i\|} \quad (2.8)$$

The learning rate η is an annealing factor that decreases gradually over time with respect to the function:

$$\eta_t = \eta_0 \left(\frac{\eta_f}{\eta_0} \right)^{\frac{t}{NM}} \quad (2.9)$$

In the above function, N is the number of document objects, M is the number of batch iteration and η_f is the desired learning rate that the algorithm should finally arrive at. Compared with the original Spherical k -means, the online spherical k -means was shown to improve clustering performance in terms of both quality and speed.

2.2.2 Self-Organizing Feature Mapping

This famous neural networks-based technique has a functionality that makes it more special than other techniques in a way. It not only groups data into clusters, but also visualizes them. It provides a two-dimensional lattice structure, where lattice units are called neurons. High-dimensional data vectors are

then projected onto this lattice and displayed as points surrounding their related neurons. Interested readers can refer to Kohonen's publications [12] and [13] for further study. Generally, a Self-Organizing Feature Mapping (SOFM) process is carried out by these steps:

1. *Initialization*: Topology of the SOFM network is defined. The number of neurons k is determined, and each of them is associated with a randomly initialized prototype vector $\mathbf{w}_m$. $\mathbf{w}_m, m = 1, \dots, k$ are have d dimensions as data vectors $\mathbf{X} = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$.
2. *Winner selection*: One data vector $\mathbf{x}$ is drawn randomly from $\mathbf{X}$ to input into the network. The winning node, denoted as c , is chosen based on Euclidean distance between its prototype vector and the input vector:

$$c = \arg_m \min \|\mathbf{x} - \mathbf{w}_m\| \quad (2.10)$$

3. *Adaptation*: The winner node and its neighbors are adjusted to fit the current input. The learning rule proposed by Kohonen is:

$$\mathbf{w}_m(t+1) = \mathbf{w}_m(t) + h_{cm}(t) \times [\mathbf{x} - \mathbf{w}_m] \quad (2.11)$$

The neighborhood function $h_{cm}(t)$ is decreasing over time, and often defined by:

$$h_{cm}(t) = \alpha(t) \exp\left\{-\frac{\|\mathbf{r}_c - \mathbf{r}_m\|^2}{2\sigma^2(t)}\right\} \quad (2.12)$$

where $\alpha(t)$ and $\sigma(t)$ are monotonically decreasing learning rate and kernel width function, and $\|\mathbf{r}_c - \mathbf{r}_m\|$ represents the distance between the winner neuron c and a neuron m .

4. The Winner selection and Adaptation steps above are iterated until no change in the neuron lattice is observed significantly.

In [14], and more recently in [15], the authors presents a SOFM-based method called WEBSOM to organize a massive collection of about 7 millions patent abstracts onto two-dimensional display. It provides an interesting way of browsing and exploring information. Besides, a few variants of SOFM have been proposed targeting the design of network topology and improvement of computational speed. One example of such work, which is aimed for document clustering, is reported in [16].

2.2.3 Fuzzy Clustering

Fuzzy clustering approach makes use of fuzzy set theory when partitioning data. Different from other methods such as k -means, which assign one object to one cluster, fuzzy-based techniques allow a data object to belong to all the clusters. There are certain degrees of membership to represent how strongly the object is related to the clusters. Probably the most well-known and generic fuzzy clustering algorithm is Fuzzy C-Means (FCM) [17]. Given a set of data objects $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \dots, n$, FCM aims to group the data into c fuzzy clusters, by minimizing the objective function:

$$f(\mathbf{U}, \mathbf{M}) = \sum_{m=1}^c \sum_{i=1}^n (u_{mi})^\beta D(\mathbf{x}_i, \boldsymbol{\mu}_m) \quad (2.13)$$

$$\text{s.t. } \sum_{m=1}^c u_{mi} = 1, u_{mi} \geq 0 \quad \forall i \quad (2.14)$$

where $\mathbf{U} = [u_{mi}]$ is the $c \times n$ fuzzy partition matrix, whose element u_{mi} is the membership degree of object i to cluster m . Matrix $\mathbf{M} = [\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_c]$ is prototype matrix, with column $\boldsymbol{\mu}_m$ representing clustering m . Parameter β controls the fuzziness of the data sets, and is normally set to 2. $D_{mi} = D(\mathbf{x}_i, \boldsymbol{\mu}_m)$ is some distance measure between the two vectors. The approximation process to solve the optimization problem of FCM is described below:

1. *Initialization*: Number of cluster c is defined, and column vectors $\boldsymbol{\mu}_m, m = 1, \dots, c$ in matrix $\mathbf{M}$ are randomly assigned.
2. *Membership update*: Membership degrees are updated as

$$u_{mi} = 1 / \left(\sum_{l=1}^c (D_{li}/D_{mi})^{1/(1-\beta)} \right), \forall m = 1, \dots, c \text{ and } i = 1, \dots, n \quad (2.15)$$

3. *Prototype update*: Following the previous step, the prototype vectors are adjusted by

$$\boldsymbol{\mu}_m = \frac{\sum_{i=1}^n (u_{mi})^\beta \mathbf{x}_i}{\sum_{i=1}^n (u_{mi})^\beta}, \forall m = 1, \dots, c \quad (2.16)$$

4. Repeatedly, membership degrees and prototype vectors are updated until convergence under some predefined threshold.

One disadvantage of FCM is that it is sensitive to noise and outliers. To overcome this problem, Possibilistic C-Means (PCM) was proposed [18]. Basically,

PCM relaxes the constraint (2.14) to become $u_{mi} > 0, \forall m, i$. It means membership degrees of an object to all the clusters must not sum to 1. However, PCM has its own drawback that it tends to produce overlapping clusters. An improved version of fuzzy-based clustering, called Possibilistic Fuzzy C-Means (PFCM), was introduced in [19]. The authors combine two techniques into one in order to take advantage of each, and solve the problems of both. Three methods above serve as the basic background for fuzzy-based clustering approach. Nevertheless, they are still far from being efficient for document categorization. The intensive research work on this direction over the past decades has led to numerous variants of fuzzy-based clustering algorithm. Some of them are specifically designed for text clustering, such as Fuzzy Co-clustering of Documents and Keywords (Fuzzy CoDoK) in [20], Fuzzy Simultaneous KeyWord Identification and Clustering (FSKWIC) in [21], and Possibilistic Fuzzy Co-Clustering (PFCC) in [22].

2.2.4 Non-negative Matrix Factorization

The birth of LSA technique, mentioned in section 2.4.2.4, and its application to text analysis has been stimulating other methods. Generally speaking, LSA can be considered as a matrix factorization technique, where the term-document matrix is divided into sub-matrices representing terms and documents in a latent semantic space. More recently, one approach to document clustering called Non-negative Matrix Factorization (NMF) has been developed [23]. Its name describes all its basic idea of how to cluster data. Its main difference from LSA is that sub-matrices decomposed from the original term-document matrix are non-negative, not containing any negative values like in the case of LSA. Besides, LSA makes use of SVD for factorizing matrix, whereas NMF directly solves a minimization problem by iterative approximation process. More precisely, given a document corpus of k topics, with d words, n documents, and represented by $X \in \Re^{d \times n_+}$, NMF aims to minimize the objective function:

$$f = \frac{1}{2} \|X - UV^T\| \quad \text{s.t. } U \in \Re_+^{d \times k}, V \in \Re_+^{n \times k} \quad (2.17)$$

So, X is approximated by two non-negative matrices $U = [u_{jm}]$ and $V = [v_{im}]$ ($j = 1, \dots, d; m = 1, \dots, k; i = 1, \dots, n$). This constrained optimization problem can be explicitly solved by general approach: taking derivatives with La-

grange multiplier. It results in the following updating formulas for U and V :

$$u_{jm}(t+1) = u_{jm}(t) \frac{(XV)_{jm}}{(UV^TV)_{jm}} \quad (2.18)$$

$$v_{im}(t+1) = v_{im}(t) \frac{(X^TU)_{im}}{(VU^TU)_{im}} \quad (2.19)$$

Once the updating iteration is converged, matrix V itself is considered as the clustering result. Each row i of V stands for a document $\mathbf{v}_i$ projected into k -dimensional latent semantic space, i.e. $\mathbf{v} = [v_{i1} \dots v_{ik}]$. Document i is assigned to cluster c if $c = \arg \max_m v_{im}$ for $(m=1, \dots, k)$. Its simple way to identify a document's class is claimed to be more favorable than LSA's.

After the original NMF, a number of its variants has been proposed, such as convex and semi-NMF [24]. These NMF algorithms are different from each other by how the objective function is constructed, and non-negativity constraints on the factorization matrices. A study on various NMFs is reported in [25]. Among those, a method called Orthogonal Nonnegative Matrix Tri-factorization shows an attractive performance [26]. It simultaneously do clustering in document space and word space- a methodology called "co-clustering" which is used specifically on high-dimensional data like text documents. Co-clustering is examined in section 2.3. In [27], the authors proposed a new method called Nonnegative Double Singular Value Decomposition (NDSVD) to enhance the initialization stage of NMFs. Various NMF-based algorithms and their applications in text mining field are studied in [28].

2.2.5 Spectral Clustering

The methods in this category apply graph theory to model the clustering problems. The basis of spectral clustering techniques is to represent data by an undirected graph $G(V, E, A)$, where V is a set of vertices whose elements correspond to data objects, E is a set of edges representing associations among the objects and A is an affinity matrix. An edge e_{ij} is assigned an element a_{ij} from A , which is often a measure of proximity or similarity of objects i and j . An example is $a_{ij} = \mathbf{x}_i^T \mathbf{x}_j$, the cosine similarity between document vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. Clustering solution is then achieved by finding the best cut to divide G into sub-graphs and optimize certain predefined objective function.

Let V_i denote a vertex subset of V corresponding to cluster i and $W(V_i, V_j)$ the sum of similarities between vertices in V_i and those in V_j . Depending on the objective function, different spectral clustering methods have been proposed.

The Ratio Cut (RC) [29] aims to minimize the inter-cluster similarity normalized by cluster size:

$$f_{RC} = \sum_{j=1}^k \frac{W(V_j, V - V_j)}{|V_j|} \quad (2.20)$$

Similarly, the Normalized Cut (NC) [30] also aims to minimize the inter-cluster similarity, but normalizes it with a measure of compactness of the data:

$$f_{NC} = \sum_{j=1}^k \frac{W(V_j, V - V_j)}{W(V_j, V)} \quad (2.21)$$

Another method called the Min-Max Cut (MMC) [31] has the objective to simultaneously minimize the inter-cluster similarity and maximize the intra-cluster similarity at the same time:

$$f_{MMC} = \sum_{j=1}^k \frac{W(V_j, V - V_j)}{W(V_j, V_j)} \quad (2.22)$$

With some matrix transformation and applying the Rayleigh Quotient Theorem [30], all the graph cutting optimization problems above can be solved by finding the set of k smallest or largest eigenvectors and eigenvalues.

There is a clustering package named CLUTO which has been developed and made freely available by the researchers at University of Minnesota [32]. CLUTO implements many different hierarchical and partitional clustering methods. It also has a min-cut nearest-neighbor graph partitioning algorithm that utilizes various types of similarity measure, as well as pruning, coarsening and uncoarsening techniques. CLUTO has become very popular for document clustering and microarray gene analysis. One disadvantage of graph-based spectral clustering is that the pairwise similarity of the vertices has to be explicitly defined, and the affinity matrix has to be pre-computed, leading to both memory and computational difficulties when working with large and high-dimensional data. Some other recent developments in this area include bipartite graph for co-clustering [33], spectral clustering with discriminant analysis [34], or with projection to low-dimension semantic space and new correlation similarity [35]. Algorithms that incorporate parallel computing technologies have also been proposed to overcome the memory and computational demands mentioned previously [36].

2.2.6 Search-based Clustering

With the recent advance in metaheuristic techniques that are originally used in optimizations for exploring large search, a branch of clustering research field starts to focus on applying these search methods to find the optimal partition for data. These metaheuristics include Genetic Algorithm (GA), Simulated Annealing (SA), Taboo Search (TS) and Particle Swarm Optimization (PSO).

Most popular among this group is GA. In GA, the idea is to represent a candidate solution with a chromosome, which is encoded, for example, as a binary bit string or as a matrix of k prototype vectors of a valid partition of the data. The algorithm is initialized with a population consisting of a number of chromosomes. Over multiple iterations, genetic operations such as crossover and mutation are applied on the chromosomes producing new instances. The best individuals from the group of old and new chromosomes are selected according to some objective function (here called fitness function) and then carried on to the next generations. GA algorithms differ from one another in chromosome encoding methods, fitness function definition and the way genetic operations are constructed. Some examples of GA-based clustering algorithms are [37–39]. GA techniques are also integrated into other algorithms, e.g. EM [40], to empower the searching and learning capabilities of these algorithms. Besides, due to its population-based and parallel characteristics in nature, GA is a very suitable tool for clustering problems involving multiple objectives [41] and parallel or distributed computing [42], which are of practical importance in real-life applications.

2.2.7 Mixture Model-based Clustering

Finite mixture model is a mathematical approach to modeling of data with strong statistical foundation. It has been widely applied to a variety of heterogeneous kinds of data, especially in the field of cluster analysis [43]. In this approach, data are assumed to be generated from a mixture of probability distributions. The clustering task then becomes a process of finding parameters of the mixture components. Each component corresponds to a cluster. At the end, any data points found to be generated by the same component will belong to the same cluster.

Let $\mathbf{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_n\}$ be a random sample of size n , where each of $\mathbf{X}_i, i = 1, \dots, n$ is a d -dimensional random vector, and follows a probability density function $f(\mathbf{x})$. We use lower-case letters $\mathbf{x}_1, \dots, \mathbf{x}_n$ to denote the observed

random sample of $\mathbf{X}$ given in a particular context, in which $\mathbf{x}_i$ is the realized value of random variable $\mathbf{X}_i$. We say $\mathbf{X}$ follows a k -component finite mixture distribution if its probability density function can be written in the form:

$$f(\mathbf{x}|\Theta) = \sum_{m=1}^k \alpha_m f_m(\mathbf{x}|\theta_m) \quad (2.23)$$

where each f_m is a probability density function, and is considered as a component of the mixture. Non-negative quantities $\alpha_1, \dots, \alpha_k$ are called mixing probabilities ($\alpha_m \geq 0, \sum_{m=1}^k \alpha_m = 1$). θ_m denotes a set of parameters defining the m^{th} component, and $\Theta = \{\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k\}$ denotes the complete set of parameters needed to define the mixture. It is normally assumed that all the components f_m have the same functional form.

Under this model, the problem of identifying k clusters transforms into problem of determining the set of parameters Θ . The most well-known approach to fitting data into mixture of models is Maximum Likelihood (ML) [44]. The likelihood function of the entire data set is its probability of being generated from the given mixture distributions. If $\mathbf{x}_1, \dots, \mathbf{x}_n$ are independent and identically distributed, the likelihood to the k -component mixture will be:

$$L(\mathbf{X}|\Theta) = \prod_{i=1}^n f(\mathbf{x}_i|\Theta) \quad (2.24)$$

and its logarithm form is:

$$\log L(\mathbf{X}|\Theta) = \sum_{i=1}^n \log \sum_{m=1}^k \alpha_m f_m(\mathbf{x}_i|\theta_m) \quad (2.25)$$

The log-likelihood is used as an objective function of the optimization process. The aim of ML is to estimate the set of parameters Θ so as to maximize this function.

$$\hat{\Theta}_{ML} = \arg \max_{\Theta} \{\log L(\mathbf{X}|\Theta)\} \quad (2.26)$$

A well-known technique for solving this optimization problem is Expectation-Maximization (EM) [45]. It is an iterative procedure that helps finding the local maximum of ML. This algorithm interprets $\mathbf{X}$ as “incomplete data”. What “missing” is a set of n vectors $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$ corresponding to n elements of $\mathbf{X}$. Each vector has k binary values, i.e. $\mathbf{z}_i = [z_{i1}, \dots, z_{ik}]$. An object $\mathbf{x}_i \in \mathbf{X}$ belongs to the m^{th} component if $z_{im} = 1$, otherwise $z_{im} = 0$. Then, the

“complete” log-likelihood is:

$$\log L_c(\mathbf{X}, \mathbf{Z}|\Theta) = \sum_{i=1}^n \sum_{m=1}^k z_{im} \log[\alpha_m f_m(\mathbf{x}_i|\theta_m)] \quad (2.27)$$

There are two steps in EM algorithm: E-step and M-step. In the first step, algorithm starts with the given data set $\mathbf{X}$ and initialized value $\hat{\Theta}(t=0)$. The conditional expectation of the complete log-likelihood is estimated. The result is a function Q of Θ :

$$Q(\Theta; \hat{\Theta}(t)) \equiv E[\log L_c(\mathbf{X}, \mathbf{Z}|\Theta) | \mathbf{X}, \hat{\Theta}(t)] \quad (2.28)$$

The M-step updates the parameter set Θ by maximizing function Q :

$$\hat{\Theta}(t+1) = \arg \max_{\Theta} \{Q(\Theta, \hat{\Theta}(t))\} \quad (2.29)$$

These two steps are repeated until no further significant changes in the likelihood value. It has been proven that the likelihood value under EM updates is monotonically non-decreasing. At convergence, clusters are determined based on the estimated values in $\mathbf{Z}$. Object i is assigned to cluster c if $c = \arg \max_m z_{im}, \forall m = 1, \dots, k$. If referring back to section 2.2.3, we can see that parameters z_{im} are similar to the degrees of membership u_{mi} in fuzzy. Hence, M2C is also considered as soft assignment like fuzzy clustering in this sense. However, different from fuzzy clustering concept, M2C is generative approach in which data are assumed to follow certain probability distributions. Under this model, it follows that cluster memberships also represent the true probabilities that data are generated from the corresponding mixture components.

This is the general framework for every M2C method. Depending on what family of probabilistic distributions is used, we have different type of mixture models, such as mixture of Gaussians or mixture of multinomials. They are different from one another by their parameter sets, so the parameter updates in the M-step should also be different. However, the E-step is basically identical in all the cases. From equations (2.27) and (2.28), it can be observed that given $\mathbf{X}$ and the current estimate $\hat{\Theta}(t)$, the expectation of the complete log-likelihood is determined by the expectation of $\mathbf{Z}$. Besides, in (2.27), $\log L_c(\mathbf{X}, \mathbf{Z}|\Theta)$ is linear w.r.t. z_{im} ($i = 1, \dots, n; m = 1, \dots, k$). Hence, calculating the expectation of $\log L_c(\mathbf{X}, \mathbf{Z}|\Theta)$ is equivalent to calculating expectation of each z_{im} , denoted by ω_{im} :

$$\omega_{im} = E[z_{im} | \mathbf{X}, \Theta(t)] = \text{Prob}[z_{im} = 1 | \mathbf{X}, \Theta(t)] \quad (2.30)$$

Applying Bayes law yields:

$$\omega_{im} = \frac{\hat{\alpha}_m(t) \cdot f_m(\mathbf{x}_i | \hat{\theta}_m(t))}{\sum_{j=1}^k \hat{\alpha}_j(t) \cdot f_j(\mathbf{x}_i | \hat{\theta}_j(t))} \quad (2.31)$$

So ω_{im} is the posterior probability which represents the likelihood that object i belongs to component m . As a result, the function Q in (2.28) becomes:

$$Q(\Theta; \hat{\Theta}(t)) = \sum_{i=1}^n \sum_{m=1}^k \omega_{im} [\log \alpha_m + \log \{f_m(\mathbf{x}_i | \theta_m)\}] \quad (2.32)$$

In the M-step, by taking partial derivatives of function Q in (2.28) w.r.t. different parameter variables, the following updating formula is obtained for the mixing probabilities:

$$\hat{\alpha}_m = \frac{1}{n} \sum_{i=1}^n \omega_{im} \quad (2.33)$$

Depending on the particular type of probabilistic distribution that is used for the mixture model, other model parameters also need to be updated. Following the above framework, in the next chapter, we analyze different types of mixture model that have been known as good solutions to the data clustering problem.

Recently, different variations and enhanced versions of EM-based clustering algorithm have been proposed. These algorithms have EM nicely incorporated with other techniques such as Minimum Message Length, GA, split-and-merge and so on. They are aimed to address the drawbacks encountered in the original EM framework, and are discussed in the next Section.

It should be highlighted that although ML-EM algorithm is very popular, it is not the only approach to learning mixture model for clustering purpose. In the context of Gaussian mixture model-based clustering, researchers have proposed alternative techniques to estimate components of a Gaussian mixture. An example is Dasgupta's algorithm presented in [46]. The algorithm does not employ ML-EM, but instead consists of four steps. Firstly, data are projected to a lower dimensional space by a random projection. Secondly, a density-based technique is applied to cluster the data points and find the centers in the projected space. Then, the high-dimensional estimates of the cluster centers are reconstructed from the low-dimensional ones that have just been found. Finally, the overall clustering is achieved by assigning data points to the closest center estimate in high-dimensional space. A major advantage of this algorithm is that it has high probability of finding the true centers of the Gaussians to within the

precision defined by users.

Another representative example of algorithms for clustering data through fitting mixtures of Gaussians is Variational Bayes (VB). This approach of learning Gaussian model has often been studied in conjunction or in comparison with EM, since it could be considered as extension of EM. Usually, VB approach also leads to some iterative procedure for estimating the mixture’s component parameters. However, unlike the original EM, whose singularity problem does not facilitate the inference of the number of mixture components well, VB methods impose priors on the component parameters, and has an criterion optimization process that allows simultaneous estimation of the parameters and the number of components, i.e. the number of clusters. Some typical examples of research work done in this direction are [47–49].

2.3 Existing Problems and Potential Solution Approaches

2.3.1 The Curse of Dimensionality

Text documents are regarded as high-dimensional data. But how high is “high”? Data with more than 16 attributes are considered high-dimensional, according to Berkhin in [50]. One text document, on the other hand, normally has a few thousands of words, each of which is counted as a feature. All the documents in a certain collection then add up to tens or hundreds of thousands of features. Hence, the meaning of “high dimensionality” in text clustering domain is pushed to the most extreme level. Because of this characteristic, it is when working with text documents that the problems caused by high dimensionality critically arises. Most of the features of a document vector in VSM model are irrelevant, or even create noisy information. Only a small part of the features actually carries some meanings toward the document’s topic. In this ill-informative feature space, dissimilarity measures based on distance such as Euclidean fail to perform effectively on text. Consequently, clustering performance can be seriously affected.

Many approaches have been proposed for clustering algorithms to overcome this curse of dimensionality. These approaches mainly focus on dealing with the feature aspect of data. They provides techniques that are either added on as pre-processing steps before the clustering algorithms, or embedded into the algorithms to proceed in parallel. It is impossible to list out all of the numerous

amount of methods and their variations. We summarize a few important ideas below:

1. *Feature selection (FS)*: Generally, FS methods base on some particular criterion to calculate a score value for each word. This value represents the quality, or importance, of a word in the collection. They then rank the words in descending or ascending order according to the values, and select a suitable number of words of highest ranks. Conventional FS methods, such as Document Frequency (DF) [51], Term Contribution (TC) [52] and Mutual Information (MI) [53], are simple but have shown to be efficient. New methods continue to be developed over the years, such as the work reported in [54] which is based on Best Individual Features selection scheme, or in [55] which is a supervised method using χ^2 statistic.
2. *Feature reduction (FR)*: Feature Reduction techniques, on the other hand, seek to actually transform the original word space into a completely different sub-space. It is often called latent sub-space, since it is more compact, in much lower dimension, and promises to intrinsically represent the data better. It is usually established by a linear, and sometimes non-linear, transformation of the original word space. Let $\mathbf{X}_0$ be a d -by- n matrix representing the initial corpus in VSM model, with d words and n documents. A FR method will find a d -by- r matrix $\mathbf{A}$ such that:

$$\mathbf{X} = \mathbf{A}^T \mathbf{X}_0. \quad (2.34)$$

The new matrix $\mathbf{X}$ has dimension r -by- n , i.e. each of n documents now has only r features, where $r \ll d$. Matrix $\mathbf{A}$ is sometimes called projection matrix, since it projects the data from a d -dimensional feature space into a r -dimensional one. The popular Latent Semantic Analysis (LSA) method, [56], was initially proposed for indexing and information retrieval, but has been shown to produce great clustering or classification result when used as a FR technique [57–59]. Another, very popular, technique for reducing the feature dimension of data is Principal Component Analysis (PCA). It allows projection of data into a subspace that captures the most variation in the data. Application of PCA in fitting high-dimensional Gaussian model using EM has been well-studied, for instance [60]. Another approach is random projection. The basis of this kind of technique is to project data into a randomly chosen r -dimensional subspace. While PCA should not be used to reduce the dimensionality of a mixture of k Gaussians to below

$\Omega(k)$, random projection is said to allow effective projection to $O(\log(k))$ -dimensional subspace. Representative work on random projection and its comparison with PCA can be found in [46, 61].

3. *Sub-space clustering*: It is similar to FS, in the way that some criterion function must be utilized to select informative features, and omit irrelevant ones. There is a major difference between the two though. FS is a global approach, where after the selection phase, all the documents have a same set of features. In sub-space clustering, it is believed that clusters can be recognized and distinguished when we look into different sub-spaces of the original feature space. Hence, FS in sub-space clustering is locally-oriented. It means documents belonging to different clusters will have different sub-set of features. The selection criteria in sub-space clustering must, therefore, be more robust to be able to detect potential sub-spaces. A good survey on sub-space clustering for high-dimensional data is reported in [62].
4. *Co-clustering*: It is also called bi-clustering. So named because it is an approach where both objects and objects' features are clustered simultaneously. Feature selection itself is treated as a clustering process. Clustering in the feature direction is carried out dynamically and in parallel with clustering in the object direction. At the end, the result shows not only group of objects in a cluster, but also groups of features that best represent that cluster. This adds an advantage in clustering result description and interpretation. [63], [64] are two examples of Gaussian mixture model-based co-clustering, while [34] is another one but based on multinomial distribution. Besides, some of the fuzzy clustering methods mentioned earlier, such as Fuzzy CoDoK [20], FSKWIC [21] and Nonnegative Matrix Factorization methods such as Tri-NMF [26] also provide co-clustering capability.

2.3.2 The Number of Clusters

Any of the clustering algorithms mentioned above has one initial assumption: the number of classes of the dataset it is applied on is known a priori. In variants of k -means, for example, the value of k is predefined. So is the number of components in M2C methods, where this problem is also regarded as “model selection problem”. This can be considered as some kind of domain knowledge, something we already know about the data. However, this is not always the case

in practice. If we have a totally new set of data, we will not know how many categories there are in that dataset.

Over the years, many algorithms have been developed to address this issue. Most of them follow a deterministic approach, where the algorithms normally run through a range of values for k to generate a set of candidates, then select the most suitable model, according to:

$$k = \arg \min_r C(\hat{\Theta}(r), r), \quad r = \{r_{min}, \dots, r_{max}\} \quad (2.35)$$

where $C(\hat{\Theta}(r), r)$ is some criterion function w.r.t. r , and $\hat{\Theta}(r)$ is estimate of the model's parameter set corresponding r . Typical examples of such criteria are the Bayesian Inference Criterion (BIC) [65], [44] and the Minimum Message Length (MML) [44], [66]. The drawback of all the methods that follow this framework is that they have to run back and forth several times, with different values of k , in order to select the most suitable one. Recently, researchers have been trying to improve this model by integrating the model selection criterion into the clustering algorithm, so that there is no need to re-run the whole clustering process with different k values. One successful example is the work of Figueiredo and Jain reported in [67], where a MML-based criterion is derived to fit Gaussian mixture model. However, by our analysis in Chapter 3, we show that their method can hardly work on text documents. Other related approaches for model selection are genetic-based EM [40], or component splitting [68], [47]. To our knowledge, no methods for model selection have shown to succeed or perform satisfactorily in text clustering problem.

2.3.3 Initialization Problem

The problem of initialization is about an algorithm's sensitiveness to its initial state. M2C methods, like all others that utilize EM in general, encounter this kind of problem. Given a bad initialization, they may converge to a not-so-good local optimum, leading to a not-so-good clustering result. This problem exists even with low-dimensional data, let alone high-dimensional ones such as documents. There have been a few different initialization schemes developed throughout the years:

- For each component, a data object is selected randomly from the dataset to be used as its mean vector. This scheme can work well only if each true class has at least one representative selected.

- Sample mean of the data can be calculated, and assigned to mean vector of each component with some small random perturbation.
- Otherwise, an algorithm can be initialized by labeling each object by one of the components randomly. Then, the parameters of a component is determined based on the objects that have been assigned to that component.

Nevertheless, there is not yet an absolute solution to the problem. The effect of those initialization schemes is rather context-dependent. It is understandable, because how to escape local optimum is not only the obstacle of EM-based algorithms, but also of optimization community in general. A good initialization would only lead to a higher chance of heading to a good optimum.

Beside the above schemes, the standard k -means is also often used as another way of initialization for M2C. Mentioned in the previous section, the algorithm in [67] is claimed to be less sensitive to initialization than standard EM-based ones. In [69], the authors proposed a new algorithm called split-and-merge expectation-maximization (SMEM) to overcome the local maxima problem for mixture models. It was later further improved by other researchers in [70] and [71]. However, just like in the case of [67] for model selection problem, this SMEM also performs well on low-dimensional sample data, or in an image compression application as shown in the paper, but it fails to produce reasonable result when applied for document classification.

2.3.4 Outlier Detection

Dependent on the use of ML estimate, M2C methods are not robust to outliers. If we take a look back at the equations (3.2) and (3.3) of Gaussian mixture, for example, its mean and covariance estimates rely heavily on weighted values of the sample observations. If there exists a gross outlier in the data, at least one of these estimates will be altered magnificently. One method to detect outliers is to use an appropriate metric, such as Mahalanobis distance in [72] and [73], to measure the distance between a data object and a data cluster's location, with respect to its dispersion. An example of Mahalanobis distance at its simplest form is:

$$D_i(\boldsymbol{\mu}_m, \Sigma_m) = \sqrt{(\mathbf{x}_i - \boldsymbol{\mu}_m)^T \Sigma_m^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_m)}, \quad (2.36)$$

calculating the distance from object $\mathbf{x}_i$ to the mean estimate $\boldsymbol{\mu}_m$ of cluster m , taking into account its covariance Σ_m . However, outliers can affect a cluster's location estimate, i.e. the mean, where they attract the mean estimate toward

their location and far away from the true cluster’s location. Outliers can also inflate the covariance estimate in their direction. For those reasons, D_i value for an outlier may not necessarily be large, and that outlier will hardly be detectable. This is called the “masking” effect, as the presence of some outliers mask the appearance of another outlier. On the other hand, D_i value of certain non-outlying object may possibly become large, hence makes it misclassified as atypical if based on the criterion. This is called the “swamping” problem. Therefore, determining outliers based on such a criterion is either ineffective or inefficient.

There are some other ideas to deal with noise and outliers when modeling data with probabilistic mixture. In [74], the authors introduced an additional component- a uniform distribution- into the mixture of Gaussian distributions to account for the presence of noise in data. However, according to Hennig, while providing a certain gain of stability in cluster analysis, this approach does not prove a substantial robustness to the outlier detection problem [75]. Another approach is to employ Forward Search technique, such as the ones proposed in [76], [73] and [77]. A Forward Search-based method starts by fitting a mixture model to a subset of data, assumed to be outlier-free. The rest of the data are then ordered based on some metric, e.g. the Mahalanobis distance in (2.36), with regarding to fitted model. Next, the subset is updated by adding into it the “closest” sample. The search goes on by repeated fitting and updating until all the population are included. Although this approach have shown the ability of detecting multiple outliers in multivariate data, one drawback is its heavy reliance on the choice of distance metric. As discussed earlier, measures like Euclidean or Mahalanobis distance perform poorly on high-dimensional data, especially text with sparsity characteristic.

2.4 Text Document Clustering

Image segmentation, microarray gene analysis and automatic document categorization are the typical examples of application areas where high-dimensional data clustering is found useful. Among these fields, we are most interested in the one involved text documents. Recent research developments of clustering methods for gene data analysis can be observed through some of the works such as [78–81]. The clustering toolkit CLUTO that we have mentioned earlier also works on microarray data, and a web-based application built on top of this engine has been developed [82]. The use of clustering methods for image segmentation

Fig. 2.1. A snapshot of search engine WebClust

have also become standard. Some good examples of this field are [30,83,84]. In the next paragraphs, we focus more on our area of interest which is document clustering. We explain the potential benefits from clustering of documents. It is also necessary to describe how text documents are transformed and represented in the feature space.

2.4.1 Applications to Web Mining & Information Retrieval

The World Wide Web is a tremendous resource of information. Nowadays, almost any of us know how to use a web search engine to look for information, and some of us probably do that several times a day. That comes from our needs of information, but it also shows how much all these technologies has become an integrated part of our daily life. As we all know, Google is the powerhouse and the leading company in the web search industry. However, there are other parties trying hard to do better than the search engine inventor, and one of the technologies they rely on to achieve that is clustering. Examples are the new search engines such as WebClust and Yippy Search. Fig. 2.1 is a snapshot

of results returned to the keywords “web mining” from WebClust. Apart from returning the relevant web pages, the engine also uses clustering techniques to group them into different topic categories, which are displayed on the left. Apparently, web pages containing the words “web mining” can be about Usage Mining, Content Mining, Pattern Discovery and so on. By categorizing the information before presenting to users, it hopes to help them manage the data and find what they are looking for more easily.

Not only at the interface and presentation level, clustering technology can help search engines at the lower level, where data indexing and retrieval are carried out. For example, there are circumstances where we have to search for relevant documents from a very large collection by calculating the similarity of the query to every document. It could be more efficient if we already have the entire collection grouped into clusters; we only need to find the clusters closest to the query and limit our search to the documents from clusters. Because the number of clusters are normally a lot fewer than the number of documents, the retrieval time can be much faster. Clustering is definitely one of the useful tools that helps companies like Google build such powerful information retrieval systems.

Before any computation is done, document data must be represented in some appropriate form to be processed by the engines. The core contents of text documents are paragraphs, sentences and words that present some meaningful topics. There are also other more complex resources, e.g. pictures and charts, but we consider only textual information here. Depending on whether other information, such as grammar, syntax or semantic meanings of words is taken into account, we have different levels of representation. The typical representation models of document data are described as follows.

2.4.2 Text Document Representations

2.4.2.1 Vector Space Model

Vector Space Model (VSM) can be reckoned the simplest level of document representation in clustering [85]. Given a document collection, any word present in the collection is count as a dimension. If there are totally d separate words, each document is treated as a d -dimensional vector, whose coordinate values are the frequencies of appearance of the words in that document. Consequently, this vector is very high dimensional but extremely sparse, because a collection normally contains so many documents that only a tiny portion of the words

actually belongs to an individual document.

This representation model treats words as independent entities, completely ignoring the structural information inside documents, such as syntax and meaningful relationship between words or between sentences. Recently, many efforts have been made to find a better way of representing text document. As mentioned, sparsity is a problem of VSM. A document vector has so many unrelated dimensions that may hide its actual meaning. Researchers have tried to make use of semantic relatedness of words, or to find some sort of concepts, instead of words, to represent documents. These kinds of model will be described in the next sections. However, such semantic relatedness or concepts is hard to obtain accurately. Despite its simplicity, VSM still offers the best performance until now. Its simplicity facilitates fast computation, at the same time provides sufficient numerical and statistical information. Hence, it is the common model used in most of the clustering algorithms nowadays.

2.4.2.2 Multi-word Term and Character N -gram Representation

Multi-word term is a slightly modified version of the VSM model above. Documents are still represented as vectors, but their entities are now groups of words, or noun phrases, instead of single words [86]. The purpose is to increase semantic information, because in natural language, words are often combined orderly into terms or phrases to express an idea, object or event. Therefore, additional steps such as natural language processing and lexicon analysis must be carried out. Another advantage of this model is that dimensionality of document vectors is reduced compared to traditional word representation. However, while semantic quality might be increased, statistical quality can be inferior because group of words are obviously hard to repeat than word alone. Besides, to identify semantic relationship of words accurately is very difficult. It still remains a challenging task nowadays. Perhaps due to this reason that, although this representation sounds naturally more convincing, its experimental results are not always better than single-word VSM [87].

Character N -gram is another VSM-based representation. It is even less language-dependent than traditional word model. N -gram entities are sequences of N characters, extracted from document collection by moving a window of width N across the documents in a character-by-character manner [88]. This technique pays no regard to linguistic rules whatsoever. It simply forms sequences of characters. Depending on chosen value of N , document vectors under this representation can theoretically have up to $|A|^N$ dimensions, where $|A|$ is

size of the alphabet. However, in practice, the dimension is much lower, since not all the possible combinations are present in a given document collection. Besides, as in [87], it has been empirically shown that 3 to 4 are appropriate values for N . In [89], the authors show results where N-gram model outperforms both word and multi-word term representation.

2.4.2.3 Word-Cluster Model

When trying to reduce the dimension of vector model, and interpret more relational information of words for better representing documents, researchers have applied clustering algorithms on the words themselves [90], [91]. This means that not documents, but words become the objectives of a clustering process. It is hoped that related words would be gathered into same sub-group, which is equivalent to a concept or topic they are all intended to express. With this technique, document vector's dimension is greatly reduced, because a group of semantically related words can be replaced by its center or centroid, determined by some criterion or numerical measure. Besides, given a text collection, clustering algorithms when applied will group together words that are bearing similar meaning in their immediate context. Therefore, word-cluster model can be said to offer contextual adaptivity. A question to be answered is which algorithm should be well-suited for clustering words. In chapter 3, we propose a novel feature reduction technique based on this kind of model. It not only provides a very low-dimensional and compact document representation, but also helps improving text clustering results.

2.4.2.4 Latent Semantic Analysis

Originally, Latent Semantic Analysis or LSA was proposed by Deerwester and his colleagues for automatic indexing in Information Retrieval (IR) [56]. Thus, it is also referred to as Latent Semantic Indexing, or LSI. Let X denote a given document matrix, whose columns correspond to documents represented in VSM. LSA makes use of Singular Value Decomposition (SVD) technique to break down X into three matrices:

$$\mathbf{X}_{t \times d} = \mathbf{U}_{t \times m} \mathbf{\Sigma}_{m \times m} \mathbf{V}_{d \times m}^T \quad (2.37)$$

where X : $t \times d$ document matrix

U : $t \times m$ column-orthonormal matrix

Σ : $m \times m$ diagonal matrix

V : $d \times m$ column-orthonormal matrix

t equals number of words
 d equals number of documents
 m is the rank of matrix X

Columns of U are the left singular vectors of X , and correspond to the words (or terms), whereas V 's columns are the right singular vectors of X , and represent the documents in the collection. On the other hand, Σ has singular values of X as its diagonal elements. These values are sorted in non-increasing order from top-left to bottom-right of the matrix. It is suggested that the smallest singular values at the bottom are corresponding to noisy information. Supposed that we decide to retain only the first r largest, the remaining smaller ones can be set to zero. According to equation 2.37, it is equivalent to keeping the first r columns in matrices U and V while omitting the rest of the columns. As a result, when multiplying these modified matrices back, we will obtain an approximation $\hat{\mathbf{X}}$ of the original document matrix $\mathbf{X}$:

$$\mathbf{X}_{t \times d} = \mathbf{U}_{t \times r} \Sigma_{r \times r} \mathbf{V}_{d \times r}^T \quad (2.38)$$

$\hat{\mathbf{X}}$ is proved to be the rank- r matrix closest to $\mathbf{X}$ in term of least-square Frobenius norm. However, due to the changes in the SVD matrices described above, $\hat{\mathbf{X}}$ would never be exactly the same as $\mathbf{X}$. Its deviation from $\mathbf{X}$ is wanted, since it means some noise in $\mathbf{X}$ has been removed.

Normally, r is chosen much smaller than t . Thus, through LSA, the original document corpus is projected into a new and very lower-dimensional space. This is called the latent semantic space, in which not only documents but words are also represented as vectors (or data points). Each of these vectors has r feature values. It may happen in this new dimensional space that a word which is not physically present in a document appears to be located near that document. It is because this word may have a relationship with other ones in that document by the means of polysemy or synonymy. Hence, LSA is said to be capable of recognizing semantic meaning of words, and there comes the term "latent semantic space". As shown in [57] and [58], using LSA for document representation promises to give good improvement in IR and text clustering applications.

2.4.2.5 Knowledge-based Representation

Recently, the research community related to text and language is increasing its interest and attention to knowledge-based model. In the previous representation

schemes, either words (or terms) which are present in a document are utilized to represent that document, or some information-based transformations are carried out to create a sort of new latent space, wherein the word-based document vectors are transformed into new-coordinated compact vectors. In knowledge-based model, however, documents are represented not by their original words, but by explicit concepts. The term “explicit” here means these concepts have already been pre-defined by a separate process, e.g. using NLP and domain knowledge, and stored in a pool of knowledge, or often called an ontology system. Concepts in an ontology system are categorized into specific domains, such as artificial intelligence, bio-informatics and so on. The document processing step then uses the help of this knowledge database to replace words in documents by their related concepts. Hence, documents are no longer represented as vectors of words, but vectors of concepts instead.

Wordnet [92] can be considered as an example of a simple ontology system. It is a lexical database of English language. Related nouns, verbs, adjectives and so on are grouped together into sets to describe their semantic and lexical relations. In [93], the authors used the Wordnet ontology to create structured document vector space with low dimensionality, hence allowing usual clustering algorithms to perform well. Other examples of using ontology systems for text representation and clustering are [94] and [95]. One important property of ontology systems is the existence of relationship among entities. When documents are represented by simple word counts, distance measure (e.g. Euclidean) or cosine similarity can be used to determine relatedness among them. When they are represented by concepts, there arises the issue of how to measure correctly the relationship between these concepts. The effectiveness of this model is highly dependent on the accuracy of identifying concepts and measuring relationship among concepts, the two tasks which are still far from perfection until now. Therefore, although this approach is very promising, the VSM is still a favorite choice for text classification problem at the moment.

2.5 Document Datasets

No clustering method can be claimed to be the best in every application. They often perform differently in different domains and on different datasets. Hence, in order to have thorough and intensive examinations of the clustering methods, we utilized a large set of document collections for our experiments. All of them are popular and benchmarked datasets, which had been used extensively for

testing text classification and clustering systems in previous works, for examples [23, 96–99]. Their characteristics are described in Tables 2.1, 2.2 and 2.3. The document collections vary in content, number of topics, size, vocabulary and so on, creating a very diverse set of data on which clustering task is performed. They can be used once or repeatedly over different experiments. For the ease of the readers, we introduce all of them here. In addition, there are also simulated datasets and non-text real datasets that are used across our experiments. They are introduced and described clearly in the respective experimental sections.

Dataset *20news-18828* is a cleaned version of the well-known document corpus 20-Newsgroups¹ (originally with 19997 documents). Duplicated documents have been removed, so the number is now reduced to 18828. Newsgroup-identifying information in the documents’ text body has also been removed, but “From” and “Subject” are still kept. Dataset *classic*² is one of the most popular datasets for testing information retrieval systems. It contains the abstracts collected from computer systems papers CACM, information retrieval papers CISI, medical journal MEDLINE and aeronautical systems papers CRANFIELD. Each set of the abstracts is considered as one of the four topic classes. Datasets *classic3* and *classic300* are subsets of *classic*; *classic3* is formed by excluding CACM and using only documents from the last three topics, whereas *classic300* is created by selecting 100 documents randomly from each of the 3 topics.

Dataset *reuters10* is a subset of the famous collection Reuters-21578³, Distribution 1.0, which is one of the most widely used test collections for text categorization. We selected a sub-group of 10 categories from the collection (acq, corn, crude, earn, grain, interest, money-fx, ship, trade and wheat). Similarly, *reuters7* is another subset of Reuters-21578, containing 2,500 documents from the 7 largest categories (acq, crude, interest, earn, money-fx, ship and trade). Some of the documents may appear in more than one category. Dataset *webkb4* is a subset of WebKB, a collection of 7 groups of web pages collected from computer science department of various universities. *webkb4* covers only 4 classes of topic: student, faculty, course and project.

The rest of the datasets in Table 2.1, from *cranmed* to *tr45*, have been collected and preprocessed by the authors of the clustering toolkit CLUTO, and are made freely available on their website [32]. Dataset *cranmed* is yet another subset of *classic* and contains only the 2 groups of abstract CRANFIELD and MED-

¹<http://people.csail.mit.edu/jrennie/20Newsgroups>

²<ftp://ftp.cs.cornell.edu/pub/smart/>

³<http://daviddlewis.com/resources/testcollections/reuters21578>

Table 2.1
Document datasets I

Dataset	Source	# of topics	# of documents	# of words
<i>20news-18828</i>	20-Newsgroups	20	18,828	11,464
<i>classic</i>	CACM/CISI/ CRAN/MED	4	7,089	12,009
<i>classic3</i>	CISI/CRAN/MED	3	3,891	4,936
<i>classic300</i>	CISI/CRAN/MED	3	300	1,736
<i>reuters10</i>	Reuters	10	2,775	7,906
<i>reuters7</i>	Reuters	7	2,500	4,977
<i>webkb4</i>	WebKB	4	4,199	10,921
<i>cranmed</i>	CRAN/MED	2	2,431	5,703
<i>fbis</i>	TREC	17	2,463	2,000
<i>hitech</i>	TREC	6	2,301	13,170
<i>k1a</i>	WebACE	20	2,340	13,859
<i>k1b</i>	WebACE	6	2,340	13,859
<i>la1</i>	TREC	6	3,204	17,273
<i>la2</i>	TREC	6	3,075	15,211
<i>re0</i>	Reuters	13	1,504	2,886
<i>re1</i>	Reuters	25	1,657	3,758
<i>tr31</i>	TREC	7	927	10,127
<i>reviews</i>	TREC	5	4,069	23,220
<i>wap</i>	WebACE	20	1,560	8,440
<i>la12</i>	TREC	6	6,279	21,604
<i>new3</i>	TREC	44	9,558	36,306
<i>sports</i>	TREC	7	8,580	18,324
<i>tr11</i>	TREC	9	414	6,424
<i>tr12</i>	TREC	8	313	5,799
<i>tr23</i>	TREC	6	204	5,831
<i>tr41</i>	TREC	10	878	7,453
<i>tr45</i>	TREC	10	690	8,260

Table 2.2
Document datasets II

Dataset	Categories	# of documents
<i>A2</i>	alt.atheism	100
	comp.graphics	100
<i>A4</i>	comp.graphics	100
	rec.sport.baseball	100
	sci.space	100
	talk.politics.mideast	100
<i>B2</i>	talk.politics.mideast	100
	talk.politics.misc	100
<i>B4</i>	comp.graphics	100
	comp.os.ms-windows.misc	100
	rec.autos	100
	sci.electronics	100

Table 2.3
Document datasets III

	<i>TDT2</i>	<i>Reuters-21578</i>
Total number of documents	10,021	8,213
Total number of classes	56	41
Largest class size	1,844	3,713
Smallest class size	10	10

LINE. Dataset *fbis* is received from the Foreign Broadcast Information Service data of TREC-5⁴. Similarly, *hitech*, *la1*, *la2*, *tr31*, *reviews*, *la12*, *new3*, *sports*, *tr11*, *tr12*, *tr23*, *tr41* and *tr45* all are derived from various TREC collections. The topics that they contain are very diverse; for example, *hitech* documents are about computer, electronics, health, medical, research and technology, whilst *reviews* documents are about food, movies, music, radio and restaurants. Datasets *k1a*, *k1b* and *wap* contains web pages from the Yahoo! subject hierarchy and were created from a past study in information retrieval called WebACE [100]. Datasets *re0* and *re1* are also from Reuters-21578 collection, but unlike *reuters7* and *reuters10*, every each one of their documents has only a single label.

Table 2.2 shows the second set of text datasets which are derived from 4 subsets of the 20-Newsgroups collection. They were previously used for evaluating

⁴<http://trec.nist.gov/data.html>

EWKM method [99]. Among the four, A2 and A4 contains highly dissimilar themes, whereas B2 and B4 consist of documents from more closely related topics.

Lastly, Table 2.3 describes another two document sets that are used in our experiments: *TDT2* and *Reuters-21578*. The original *TDT2* corpus⁵, which consists of 11,201 documents in 96 topics, has been one of the most standard sets for document clustering purpose. We used a sub-collection of this corpus which contains 10,021 documents in the largest 56 topics. The *Reuters-21578* Distribution 1.0 has already been mentioned earlier. The original corpus consists of 21,578 documents in 135 topics. We used a sub-collection having 8,213 documents from the largest 41 topics. These two document collections had been used in the same way in previous works on the NMF methods [23].

All the datasets were preprocessed by standard procedures, including removing headers of the 20-Newsgroups documents, stopword removal, stemming and removal of too rare as well as too frequent words. They were done with the toolkits MC⁶ and Bow [101]. Empty documents after preprocessing were removed. Finally, the documents went through TF-IDF weighting and L_2 -normalization to unit vectors.

2.6 Evaluation Metrics

In order to assess quality of clustering results produced by an algorithm, we utilized a few different evaluation metrics to measure the clustering quality from different aspects. The metrics are *Entropy*, *Purity*, *Accuracy*, *FScore* and Normalized Mutual Information *NMI*. Here is a review of their formulation and meaning. Let c denotes the number of true classes, k the specified number of clusters (normally $k = c$), n_i the number of objects in class i , n_j the number of objects assigned to cluster j , and $n_{i,j}$ the number of objects shared by class i and cluster j . *Entropy* is defined by:

$$Entropy = \sum_{j=1}^k \frac{n_j}{n} E(S_j) \quad (2.39)$$

$$\text{where } E(S_j) = -\frac{1}{\log c} \sum_{i=1}^c \frac{n_{i,j}}{n_j} \log \frac{n_{i,j}}{n_j}$$

⁵<http://nist.gov/speech/tests/tdt/tdt98>

⁶<http://cs.utexas.edu/users/dml/software/mc>

Entropy of a cluster reflects how the various classes of objects are spread in that cluster; the overall entropy is a weighted sum across all the clusters. A perfect result would be that each cluster contains objects from only a single class. The second metric, *Purity*, is determined by:

$$Purity = \sum_{j=1}^k \frac{n_j}{n} P(S_j) \quad (2.40)$$

where $P(S_j) = \frac{1}{n_r} \max_i n_{i,j}$

For each cluster, purity means the percentage of the cluster size corresponding to the largest class of objects assigned to that cluster. Hence, two clusters can be considered as representing the same class. The overall purity is defined as a weighted sum of the cluster purities. *Accuracy* is very similar to *Purity*. In many occasions, they can have the same value. However, when identifying the fraction of documents that are correctly labeled, with *Accuracy* we assume an one-to-one correspondence between true classes and assigned clusters. Let q denote any possible permutation of index set $\{1, \dots, k\}$, *Accuracy* is defined by:

$$Accuracy = \frac{1}{n} \max_q \sum_{i=1}^k n_{i,q(i)} \quad (2.41)$$

The best mapping q to determine *Accuracy* could be found by the Hungarian algorithm⁷. *FScore* is an equally weighted combination of the “precision” (P) and “recall” (R) values used in information retrieval. It is determined as:

$$FScore = \sum_{i=1}^k \frac{n_i}{n} \max_j (F_{i,j}) \quad (2.42)$$

where $F_{i,j} = \frac{2 \times P_{i,j} \times R_{i,j}}{P_{i,j} + R_{i,j}}$; $P_{i,j} = \frac{n_{i,j}}{n_j}$, $R_{i,j} = \frac{n_{i,j}}{n_i}$

NMI measures the information the true class partition and the cluster assignment share. It tells how much knowing about the clusters helps us know about the classes:

$$NMI = \frac{\sum_{i=1}^c \sum_{j=1}^k n_{i,j} \log \left(\frac{n \cdot n_{i,j}}{n_i n_j} \right)}{\sqrt{\left(\sum_{i=1}^c n_i \log \frac{n_i}{n} \right) \left(\sum_{j=1}^k n_j \log \frac{n_j}{n} \right)}} \quad (2.43)$$

⁷http://en.wikipedia.org/wiki/Hungarian_algorithm

For all the evaluation metrics, their range of values is from 0 to 1. With respect to *Entropy*, as it reflects the randomness in assignments, the smaller its value, the better a clustering solution is. On the contrary, for all the other measures, greater values indicate better clustering solutions.

Chapter 3

Mixture Model-based Approach: Analysis & Efficient Techniques

3.1 Overview

There has been argument that the assumption given by probabilistic mixture model-based methods are not always practical enough, that not every realistic data are independently and identically distributed by some distribution functions. While not strongly denying this argument, we would like to show, by empirical experiments, that mixture model-based clustering (M2C) methods indeed perform very well on most of the real-life benchmark datasets. Besides, we also examine the effects of high dimensionality having on M2C algorithms. We point out the cases where some algorithms perform excellent on low-dimensional data, but fail dramatically when applied on text. The purpose is to understand clearly the disadvantages and problems that M2C methods often come across.

Furthermore, in the major parts of this chapter, we propose efficient techniques related to the M2C framework that eventually help to improve performance of the M2C methods. In the previous chapter, Section 2.3, we have highlighted a few existing problems that the data clustering community have been facing when dealing with high-dimensional domains. Two of the main problems are: the “curse of dimensionality” and the sensitiveness to initialization. In this chapter, we tackle these issues by proposing a novel feature reduction technique and an effective EM initialization enhancement.

Text data normally have thousands, or even tens of thousands, of features. This causes the well-known “curse of dimensionality” in text clustering. Feature reduction methods have been proposed to address this problem by transforming the text data into much lower dimension, which may eventually facilitate

clustering task and also improve clustering quality. On the other hand, due to the high-dimensional characteristic of text, cosine similarity has been proven to be more suitable than Euclidean distance metric. This suggests modeling text as directional data. The first part of this chapter presents a feature reduction technique which is derived from Mixture Model of Directional Distributions (MMDD). Empirical results on various benchmarked datasets show that our Feature Reduction technique performs comparably with Latent Semantic Analysis (LSA), and much better than standard methods such as Document Frequency (DF) and Term Contribution (TC).

The second issue to be discussed in this chapter is the initialization problem which is often encountered in probabilistic model-based clustering methods. Gaussian mixture model-based clustering is one of the most popular data clustering approaches. However, for very high-dimensional data such as text documents, it was suggested that Gaussian model is not very efficient [102]. Additional analysis is usually needed for Gaussian data. Applying Principal Component Analysis (PCA) to transform Gaussian data into lower-dimensional space is an example [60]. On the other hand, other probabilistic models (multivariate Bernoulli, multinomial and von Mises-Fisher distributions) have been deemed to be more appropriate for document clustering [96, 103]. Basically, in high-dimensional domains, the *expectation-maximization* (EM) algorithm [104], which is often used to learn the Gaussian models, faces a problem in which its cluster membership assignment is very unreliable in the initial stage. This can potentially lead to poor local optimum. In general, mixture model-based clustering offers “soft” cluster assignments, thanks to the use of EM algorithm for learning the probabilistic mixtures. Soft assignment means that a data object can be assigned to all the clusters, each with a certain degree of membership or probability. Consequently, this characteristic allows smooth transition of cluster boundaries, i.e. the membership values change gently within 0 and 1 during the EM iterations. However, in very high-dimensional space, soft assignments do not exist anymore. It is observed that the probabilities deciding cluster memberships always get very close to either 1 or 0, even in the very first few cycles of EM [105]. In spite of that, at these early cycles, membership assignments are obviously not reliable. This is also one of the reasons why initialization has a strong effect on performance of mixture model-based clustering methods. With a bad initialization, EM can be quickly trapped in a nearby bad local optimum, resulting in a bad clustering.

Theoretically, the number of local optima for EM is usually large when data’s

dimensionality is high. Hence, it is important to keep the transition smooth, especially at the early EM cycles, to prevent the search from falling into wrong tracks easily and end prematurely. To achieve this, we introduce an annealing-like technique which improves the initial phase of Gaussian model-based clustering when applied on high-dimensional data. Specifically, the characteristics and advantages of the proposed EM technique are as follows:

- The proposed method is developed specifically for Gaussian model. The principle idea is to control the size of the Gaussian ellipsoids during early EM steps.
- The method helps improve Gaussian model’s performance in document clustering. It outperforms significantly the classical models with standard EM and DAEM. It brings Gaussian model’s performance closer to that of some of the latest generative clustering approaches and, in a few experimental cases, even better than them.
- Since it is only applied to the initial stage of the clustering process, our method is faster than the previous DA framework. Compared to standard EM, it requires only a small number of additional steps and, hence, a small amount of additional computation time.

Recently, Zhong and Ghosh [96, 103] presented a unified framework for model-based clustering. They compared different models and their variations, and showed that the incorporation of deterministic annealing (DA) improved performance of model-based algorithms for document clustering. In DA, a decreasing temperature parameter is used to smoothen the clustering process, consequently introducing more softness to the membership assignments. The DA approaches to clustering were actually proposed earlier with a purpose to avoid poor local optima [106]. Deterministic annealing EM (DAEM) algorithm, proposed and applied to learning Gaussian mixture models [107], is a classic example of such work and has received a lot of attentions. Nevertheless, one drawback of DA approach is its high computational cost. Experiments on a set of popular document collections confirm that our approach gives better clustering performance than the standard EM and its deterministic annealing variant. Moreover, it also requires lower computational cost than the deterministic annealing approach. Comparisons with other state-of-the-art generative model-based methods and a well-known discriminative approach based on graph partitioning- CLUTO [32], further demonstrate the clustering quality improvement achieved by our proposal.

The remaining of this chapter is organized as follows. In Section 3.2, we discuss different probabilistic mixture models. Section 3.3 reports a comparative study that we have performed among M2C algorithms and other clustering methods, and Section 3.4 is a detailed analysis of the impacts of high dimensionality to the M2C. Following the analysis, a Feature Reduction technique applying mixture of Directional distributions is described in Section 3.5, and an enhanced EM initialization strategy is proposed for Gaussian M2C in Section 3.6. Finally, the chapter's conclusions are given in Section 3.7.

3.2 Mixture Models of Probabilistic Distributions

3.2.1 Mixture of Gaussian Distributions

Gaussian distribution, or also called normal distribution, is probably the most important family of continuous probability distribution. It has been tremendously used to model various phenomena in many different fields, from natural, scientific to social study. Its extensive applicability is supported by the well-known Central Limit Theorem. Therefore, it is obvious that normal distribution is also a wise choice in M2C. A d -multivariate random variable $\mathbf{x} \in \mathbb{R}^d$ is said to follow Gaussian distribution, with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$, if its probability density function has the form:

$$f(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\} \quad (3.1)$$

A set of sample data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is considered to be generated from a mixture of Gaussian distributions if they have a probability density function as given in function (2.23), where each $f_m(\cdot)$ is a Gaussian defined by (3.1). There is another reason supporting the popularity of Gaussian distribution in mixture model. It is that its parameter updating formulas in EM algorithm can be easily derived in closed forms. According to the framework sketched out above, in the E-step, the posterior probabilities ω_{im} ($i = 1, \dots, n$; $m = 1, \dots, k$) are calculated by formula (2.31). In the M-step, by taking partial derivatives of function Q in (2.28) w.r.t. different parameter variables, the mean vectors and covariance matrices are updated as follows:

$$\hat{\boldsymbol{\mu}}_m = \frac{\sum_{i=1}^n \omega_{im} \mathbf{x}_i}{\|\sum_{i=1}^n \omega_{im} \mathbf{x}_i\|} \quad (3.2)$$

$$\hat{\Sigma}_m = \frac{\sum_{i=1}^n \omega_{im} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_m)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_m)^T}{\sum_{i=1}^n \omega_{im}} \quad (3.3)$$

Although it has been studied for many decades, Gaussian mixture model still plays an important role in data clustering nowadays. Some up-to-date research work continue to show its useful applications in text clustering [108], feature selection for high-dimensional data [109], gene microarray data clustering [110] or image segmentation [84].

3.2.2 Mixture of Multinomial Distributions

The multinomial model has been quite popular for document clustering [96, 111]. With $\mathbf{x}_i$ representing a high-dimensional vector of word counts of document i , its distribution according to mixture component m is a multinomial distribution of the words in the document (based on naïve Bayes assumption):

$$p(\mathbf{x}_i | \theta_m) = \prod_l P_m(w_l)^{c_{il}} \quad (3.4)$$

where c_{il} is the number of times the word w_l appears in document i , and $P_m(w_l)$'s represent the word distribution in cluster m , $\sum_l P_m(w_l) = 1$. The parameter estimation for multinomial model with Laplacian smoothing is:

$$P_m(w_l) = \frac{1 + \sum_i \omega_{im} c_{il}}{\sum_l (1 + \sum_i \omega_{im} c_{il})} \quad (3.5)$$

3.2.3 Mixture of von Mises-Fisher Distributions

When studying application of mixture model on text clustering in [112], Banerjee and colleagues suggested to model text as directional data. The fact that cosine measure leads to superior result than Euclidean distance when dealing with high-dimensional data supports this idea, because in cosine measure, the direction of vectors, not the magnitude, is of interest. Subsequently, directional distributions, such as von Mises-Fisher (vMF) distribution, were used as mixture components and shown to yield promising results [113], [103].

Let $\mathbf{x}$ be a d -dimensional unit random vector, i.e. $\|\mathbf{x}\| = 1$. It is said to follow a d -variate vMF distribution if its probability density function is:

$$f(\mathbf{x} | \boldsymbol{\mu}, \kappa) = c_d(\kappa) \exp\{\kappa \boldsymbol{\mu}^T \mathbf{x}\} \quad (3.6)$$

The mean $\boldsymbol{\mu}$ is also a d -dimensional unit vector, $\|\boldsymbol{\mu}\| = 1$. Parameter κ is called concentration parameter, since it represents the density of generated random

vectors around the mean vector. The normalizing constant has the following formula:

$$c_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} \mathbf{I}_{d/2-1}(\kappa)} \quad (3.7)$$

Function $\mathbf{I}_m(\cdot)$ stands for the modified Bessel function of first kind and order m . The readers can refer to [112], [113] and [114] for a complete literature on vMF distribution and directional statistics in general. A set of data is said to follow a mixture of vMF distributions if they have a probability density function in the form given in (2.23), where each function $f_m(\cdot)$ is an instance of vMF distribution. As reported in [113], estimates of the concentration parameters during the M-step of EM algorithm are determined by:

$$\hat{\kappa}_m = \frac{\bar{R}_m d - \bar{R}_m^3}{1 - \bar{R}_m^2} \quad (3.8)$$

where $\bar{R}_m = \frac{\|\sum_{i=1}^n \omega_{im} \mathbf{x}_i\|}{\sum_{i=1}^n \omega_{im}} \quad \forall m = 1, \dots, k$

The updating of the mixing probabilities α_m and the mean vectors $\boldsymbol{\mu}_m$ are the same as in Eq. (2.33) and (3.2).

3.3 Comparisons of Clustering Algorithms

3.3.1 Algorithms for Comparison

We compare performance of different clustering algorithms to show that M2C approach has a potential to result in good cluster quality. Two forms of M2C were implemented. One of them was mixture of Gaussian distributions on unit hypersphere. A unit Gaussian distribution has a probability density function as in (3.1), where both sample variable and mean are constrained to be unit vectors, i.e. $\|\mathbf{x}\| = \|\boldsymbol{\mu}\| = 1$. We named this method ‘‘Gaussian-M2C’’. Another form of M2C implemented was mixture of von Mises-Fisher distributions, the type of directional distribution introduced in section 3.2.3. We used the term ‘‘vMF-M2C’’ to indicate this method.

In the first experimental study, two other popular clustering algorithms were carried out to compare with the two M2C ones above. The first algorithm was the variant of k -means mentioned in section 2.2.1, Spherical k -means or ‘‘Sp k means’’, which is known to have been developed specifically for sparse and high-dimensional data like text. The other algorithm was one proposed more recently: the Non-negative Matrix Factorization, or ‘‘NMF’’, which has also been

Table 3.1
Clustering result comparison I

Datasets	Evaluation	SpK means	Gaussian-M2C	vMF-M2C	NMF
<i>20news-18828</i>	Purity	.641 $\pm$.014	.590 $\pm$.022	.666 $\pm$.021	.605 $\pm$.009
	NMI	.633 $\pm$.005	.594 $\pm$.012	.650 $\pm$.009	.593 $\pm$.007
<i>classic3</i>	Purity	.992 $\pm$ 2E-4	.990 $\pm$.001	.992 $\pm$.0	.887 $\pm$.099
	NMI	.953 $\pm$ 7E-4	.947 $\pm$.004	.956 $\pm$.0	.768 $\pm$.148
<i>k1a</i>	Purity	.621 $\pm$.013	.614 $\pm$.014	.692 $\pm$.012	.615 $\pm$.016
	NMI	.524 $\pm$.010	.516 $\pm$.005	.601 $\pm$.005	.520 $\pm$.012
<i>k1b</i>	Purity	.849 $\pm$.011	.845 $\pm$.016	.864 $\pm$.017	.845 $\pm$.013
	NMI	.599 $\pm$.018	.594 $\pm$.033	.645 $\pm$.022	.595 $\pm$.020
<i>la12</i>	Purity	.777 $\pm$.018	.705 $\pm$.032	.781 $\pm$.012	.716 $\pm$.021
	NMI	.569 $\pm$.017	.493 $\pm$.035	.576 $\pm$.019	.517 $\pm$.023
<i>ohscal</i>	Purity	.562 $\pm$.012	.541 $\pm$.019	.573 $\pm$.015	.549 $\pm$.006
	NMI	.453 $\pm$.009	.432 $\pm$.012	.462 $\pm$.008	.430 $\pm$.006
<i>re0</i>	Purity	.661 $\pm$.017	.658 $\pm$.006	.665 $\pm$.009	.672 $\pm$.012
	NMI	.412 $\pm$.013	.424 $\pm$.013	.420 $\pm$.011	.404 $\pm$.009
<i>re1</i>	Purity	.667 $\pm$.012	.652 $\pm$.008	.688 $\pm$.008	.659 $\pm$.005
	NMI	.549 $\pm$.009	.537 $\pm$.007	.582 $\pm$.006	.536 $\pm$.005
<i>tr11</i>	Purity	.739 $\pm$.025	.746 $\pm$.016	.791 $\pm$.012	.748 $\pm$.017
	NMI	.568 $\pm$.033	.581 $\pm$.021	.649 $\pm$.020	.586 $\pm$.023
<i>tr12</i>	Purity	.650 $\pm$.033	.688 $\pm$.030	.708 $\pm$.016	.689 $\pm$.028
	NMI	.512 $\pm$.041	.544 $\pm$.040	.604 $\pm$.022	.548 $\pm$.037

discussed in section 2.2.4.

In the second study, vMF-M2C was compared with subspace clustering algorithms. We selected one of the latest research publications, the Entropy Weighting k -Means (EWKM) [99]. Bisecting k -means [8] and another subspace clustering, the Feature Weighting k -Means (FWKM) [10], were also included in the comparison.

3.3.2 Experimental Results

Table 3.1 summarizes the results of the first experiment on the datasets *20news-18828*, *classic3*, *k1a*, *k1b*, *la12*, *ohscal*, *re0*, *re1*, *tr11* and *tr12* (refer to Section

2.5 for the details of these datasets). For each pair of dataset and clustering algorithm, a test was carried out consisting of 20 trials. Then, only the top 10 runs out of the 20, regarding Purity and NMI measures, were considered in order to restrain bad initialization effect. Their average values are shown in the Table together with their standard deviations. The best result among algorithms with respect to a dataset is display in bold font.

As one can observe from Table 3.1, vMF-M2C results in the best clustering quality, regarding both Purity and NMI metrics, in 9 out of 10 examined datasets. The only exceptional case is with *re0*, but its quality is second among the 4 algorithms and very close to the top. This shows that vMF-M2C dominates all the other algorithms in consideration. Spkmeans shares the top rank with vMF-M2C in the case of *classic3*. However, since *classic3* is a very well-balanced and well-separated dataset, it is often expected that clustering result on it should be as good as shown in the first three cases. NMF, though, produced the worst result for this dataset. Besides, if looking at the NMI and the standard errors, we can see that vMF-M2C is not only the best but also the most consistent classifier on *classic3*.

Spkmeans and Gaussian-M2C alternatively perform better than one another on different text collections, although the results are often quite close. One thing to take note about mixture of Gaussians is that its performance is also affected by the choice of constraints on covariance matrices. There can be totally free covariance matrix, or covariance as diagonal matrix, or further more diagonal matrix with identical diagonal elements. Moreover, it can be assumed all the mixture components have the same covariance matrix. In our case, we used different one-element diagonal matrices for different components. It is also interesting to see here that Purity and NMI sometimes give us diverse evaluation perspectives. For example, with *re0*, NMF gives the best Purity score, but Gaussian-M2C produces the best NMI measure. Generally, NMI provides a stricter assessment to clustering quality than Purity.

Besides, it was also observed during our experiment that M2C methods and SphKMeans were much faster than NMF. While the first three algorithms required less 50 iterations, most of the time around 30, of their own cycles to converge to the presented results, NMF needed more than 500 iterations of its own cycles. NMF is more computationally demanding than the other three, since it involves matrix decomposition, which is similar to the case of SVD.

Table 3.2 shows the second comparison study, among vMF-M2C, Bisecting *k*-means and the two subspace clustering algorithms FWKM and EWKM on

Table 3.2
Clustering result comparison II (based on NMI values)

Dataset	vMF-M2C	Bisecting k -means	FWKM	EWKM
$A2$	0.923	0.785	0.796	0.834
$A4$	0.831	0.808	0.755	0.769
$B2$	0.648	0.470	0.605	0.721
$B4$	0.501	0.382	0.646	0.689

4 datasets $A2$, $A4$, $B2$ and $B4$ (refer to Section 2.5 for the details of these datasets). When applied on $A2$ and $A4$, vMF-M2C produces the best clustering quality. The result on $A2$ is significantly high (at some tests, the NMI values received were 1.0). However, on $B2$ and $B4$, its performance drops. Similarly, Bisecting k -means also yields good clusters on $A2$ and $A4$, but greatly reduces its efficiency on the other two datasets. Overall, the FWKM and EWKM algorithms perform relatively well on all the datasets. EWKM is the top scorer with $B2$ and $B4$.

This result can be explained by the original creation of the datasets. $A2$ and $A4$ contain semantically well-separate categories, whereas $B2$ and $B4$ consist of semantically close documents. There are more overlapping words in the later two. This is where the subspace weighting techniques of FWKM and EWKM are brought in to help the clustering process.

To conclude, we have shown in this section that M2C methods, especially vMF-M2C, are very suitable for solving unsupervised document classification problems. The comparison between vMF-M2C and EWKM suggests that further improvement should be done with vMF-M2C to enhance its performance on highly overlapping data. A possible direction for future work, for example, is to develop a vMF-M2C with local feature selection capability. In the next section, we continue to explore and analyze various issues of working with sparse and high-dimensional data.

3.4 The Impacts of High Dimensionality

3.4.1 On Model Selection

We have stated in section 2.3 that determining the number of true classes is one of the existing problems of data clustering. We also pointed out one typical example among the attempts to solve this problem. That is the work of

Figueiredo and Jain [67]. Their algorithm is so successful that it has been cited by 128 other publications up to date. However, we show here that the effect of high dimensionality makes it hardly be able to perform on text clustering.

The authors followed the Gaussian M2C framework. The key novelty in their algorithm is that, in order to perform model selection, they used a newly developed Minimum Message Length (MML) criterion, instead of the classical Maximum Likelihood (ML). The message had two-part length: estimating and transmitting the parameter space $Length(\Theta)$; and encoding and transmitting the data $Length(\mathbf{X}|\Theta)$. Minimum encoding length theory states that model's parameter estimate is the one minimizing the total length:

$$Length(\mathbf{X}, \Theta) = Length(\Theta) + Length(\mathbf{X}|\Theta) \quad (3.9)$$

After some derivations, the objective function became:

$$Length(\mathbf{X}, \Theta) = \frac{T}{2} \sum_{m=1}^k \log\left(\frac{n\alpha_m}{12}\right) + \frac{k}{2} \log \frac{T}{2} + \frac{k(T+1)}{2} - \log L(\mathbf{X}|\Theta) \quad (3.10)$$

where T is the number of parameters specifying each component of the model, and $\log L(\mathbf{X}|\Theta)$ is defined in formula (2.25). When EM was applied to solve the optimization problem, as a result of the criterion (3.10), the updating formula of the mixing probabilities in (2.33) was changed into:

$$\hat{\alpha}_m = \frac{\max\{0, \sum_{i=1}^n \omega_{im} - \frac{T}{2}\}}{\sum_{j=1}^k \max\{0, \sum_{i=1}^n \omega_{ij} - \frac{T}{2}\}} \quad (3.11)$$

while the updating of means μ and covariance matrices Σ remained the same as in (3.2) and (3.3). According to (3.11), the mixing probability of a component could be reduced to zero during the updating process. Consequently, that component would be eliminated, and the value of k - the number of clusters- in (3.10) would be reduced by one. So, Figueiredo and Jain would start their algorithm with a large value of k , eventually decrease this value by annihilating zero-mixing probability components, and select the final model corresponding to the shortest length. Interested readers are encouraged to refer to [67] for more details.

Fig. 3.1 demonstrates the use of the MML-based algorithm in fitting a mixture of 4 Gaussian components, where there is overlapping among the components. The algorithm successfully detected the number of groups existing in the generated dataset. We also tested the method on a well-known *Iris* dataset¹.

¹<http://archive.ics.uci.edu/ml/datasets/Iris>

Fig. 3.1. Fitting an overlapping bivariate Gaussian mixture: (a) true mixture; (b) initialization with $k = 20$; (c), (d) and (e) three immediate estimates; (f) the final estimate ($k = 4$). (Figure taken from [67])

Table 3.3
Characteristics of *Iris* and *classic3* data

Dataset	k : # of classes	n : # of objects/docs	d : # of features/words
<i>Iris</i>	3	150	4
<i>classic3</i>	3	3891	7982

The result was impressive, as the program correctly determined the number of classes ($k = 3$), and yielded good cluster quality. However, when we used the algorithm on a typical document dataset, *classic3*², it would fail. Let us compare the characteristics of the two types of data recorded in Table 3.3. Although both datasets have 3 classes, their size and dimension are very much different. The number of objects in *classic3* is much bigger than that of *Iris* dataset. The divergence in dimension is even greater. But more importantly, it is the difference between the ratios of feature and object. *Iris* data has only 4 features for 150 objects, whereas *classic3* even has more features than objects. How is Figueiredo and Jain’s algorithm affected by this?

From the updating formula of mixing probabilities (3.11), it is easy to conclude that the necessary condition for a component m to survive, i.e. its mixing

²[ftp://ftp.cs.cornell.edu/pub/smart](http://ftp.cs.cornell.edu/pub/smart)

Table 3.4
Values for *Iris* and *classic3* data

Dataset	n	n/k	$T_1/2$	$T_2/2$	$T_3/2$
<i>Iris</i>	150	50	7	4	2.5
<i>classic3</i>	3891	1297	15934067.5	7982	3991.5

probability does not turn to zero, is:

$$\sum_{i=1}^n \omega_{im} > \frac{T}{2}. \quad (3.12)$$

For a d -multivariate Gaussian component, the number of parameters, including those of mean vector and covariance matrix, is $T_1 = d + d(d+1)/2$ for unconstrained covariance, $T_2 = 2d$ for diagonal covariance matrix, and $T_3 = d + 1$ for diagonal matrix with one common diagonal element. On the other hand, in the left side of (3.12), we have $0 \leq \omega_{im} \leq 1, \forall i, m$. If a dataset has k classes and a total of n objects, the average cluster size is $\frac{n}{k}$, where usually $k \geq 2$. Then, for a component m that represents a cluster of average size:

$$\text{Max}\left\{\sum_{i=1}^n \omega_{im}\right\} = \frac{n}{k}$$

For the two examples of data above, the values are summarized in Table 3.4. It can be observed from the Table that the condition (3.12) can be met easily by *Iris* dataset. On the contrary, in *classic3*, due to the high value of dimension, the condition can never be satisfied, even with the upper bound of $\sum \omega_{im}$. Consequently, mixing probability of the component are forced to zero quickly and hence eliminated, especially during the early iterations of the algorithm. This phenomenon happens to almost all the components in the mixture. As a result, the number of clusters can absolutely not be identified correctly.

We have just analyzed a case study where an algorithm performs very well in low-dimensional space, but fails because of the curse of high dimensionality. The problem persists with other similar model selection methods applied on text document collections. How can these methods, such as the MML-based one discussed above, be improved? Hence, although it is already brought into attention for decades, developing a model selection algorithm which is robust enough to work on high-dimensional data still remains a very challenging issue.

Table 3.5

The highest posterior probabilities of the first few objects in ascending order and clustering purities

Dataset	$\omega_1.$	$\omega_2.$	$\omega_3.$	$\omega_4.$	$\omega_5.$	$\omega_6. \dots$	Purity
<i>Iris</i>	0.590	0.733	0.781	0.818	0.822	0.827 ...	97.33%
<i>5Newsgroups</i>	0.994	0.998	0.999	1	1	1 ...	53.97%

3.4.2 On Soft-Assignment Characteristic

M2C methods, like fuzzy clustering, are well-known for its soft-assignment characteristic. The membership of an object $\mathbf{x}_i$ to a cluster m out of k clusters is determined by $\omega_{im} = P(m|\mathbf{x}_i)$, where probability rule has $\sum_{m=1}^k P(m|\mathbf{x}_i) = 1$. So theoretically, the object can belong to all the clusters, each with a certain degree of membership. This is different from hard assignment, in which an object can belong to one and only one of the clusters.

To access the soft-assignment characteristic of M2C and the effect of high dimensionality on it, experiments were carried out on the *Iris* data, and another text collection we temporarily call *5Newsgroups*, which is a subset of the popular *20Newsgroups* dataset. *5Newsgroups* consists of documents from 5 closely related topics: comp.graphics, comp.os.ms-windows-misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware and comp.windows.x. The set has 4881 documents, a total of 23430 words with 286668 non-zero counts. It can be expected from the class names that this collection contains very similar documents; hence, the “softness” should be high and the clustering is difficult. *Iris* data, on the other hand, is known for their perfect balance (50 objects per class) and high separateness. Information on *Iris* is described in section 3.4.1. Unconstrained Gaussian M2C method was used. One might expect a higher level of soft assignment in *5Newsgroups* than in *Iris*. But it shows in Table 3.5 that this is not the case. After clustering, an object in *Iris* data has 3 posterior probabilities (w.r.t. 3 clusters), and an object in *5Newsgroups* has 5 (w.r.t. 5 clusters). For an object i , the highest value $\omega_{ic} = \max\{\omega_{i1}, \dots, \omega_{ik}\}$ among these probabilities determines that object’s cluster. For each dataset, we ordered the objects based on their $\omega_c(s)$. The smallest $\omega_c(s)$ of the two datasets are recorded in Table 3.5.

The posterior probabilities in *Iris* data shows certain degree of “softness” in assigning the objects to the clusters. It is desirable, though, since some of them are corresponding to “Versicolor” items that have been misplaced into “Virginica” category. On the contrary, the result for *5Newsgroups* data always

indicates a hard assignment rather than soft one. The probability values are very close to 1, or in most of the cases can be considered exactly 1. Does it mean the clustering algorithm has a 100% confidence on its categorization? Unfortunately, its purity measure does not say so. In fact, it is actually because of the effect of extremely high dimensionality. In a d -dimensional feature domain, if d has very big value (e.g. $d = 286668$ in 5Newsgroups case), the volume of the space becomes exponentially large. The distance between objects, between object and cluster, or between cluster and cluster also becomes so large that there is only a tiny portion of the space where any ambiguity of assignment can occur. And the chances that any data objects fall into this zone are rare. Eventually, any object, once assigned to a cluster, is assigned with a high probability, usually close to 1. So the “softness” characteristic is no longer expressed clearly. Or speaking in another way, the ambiguity in the topics of the documents is hindered by the very large space of features.

3.4.3 On Initialization Problem

In Chapter 2, we have discussed about the initialization problem that many clustering methods encounter in general. In this part of the report, we go into further analysis and find out that this matter is even more critical when dealing with text data. Let us consider a mixture of three components in a high d dimension. Mean vectors μ_0 , μ_1 and μ_2 represent the three true clusters in Figure 3.2. $\mu_0(t = 0)$, $\mu_1(t = 0)$ and $\mu_2(t = 0)$ are initialized values of the means at the beginning of an algorithm. $\mu_0(t_i)$, $\mu_1(t_i)$ and $\mu_2(t_i)$ denotes these estimates at a certain time after zero. Suppose that the initialized set consists of one instance from cluster 0, two from cluster 2 but nothing from cluster 1. After some iterations of EM, the estimate $\mu_0(t_i)$ will move to somewhere in between the two true means μ_0 and μ_1 , while $\mu_1(t_i)$ and $\mu_2(t_i)$ will move near to μ_2 , and remain there because their distance to the other two means is so large. Hence, at convergence, only one estimate is assigned to the two true clusters, whereas the third true cluster is approximated by two estimates. This is an demonstration of bad initialization leading to bad clustering result. The sensitiveness is multiplied by the effect of high dimensionality.

EM-based methods are also known for their “smooth transition”, a phenomenon where probabilities vary their values smoothly between 0 and 1. However, in M2C for text, this is hardly the case. We have reported in Table 3.6 the changes in posterior probabilities of a document object, selected randomly, in 5Newsgroups during its EM updating process. It can be observed that, although

Fig. 3.2. An example of bad initialization

Table 3.6

Changes in posterior probabilities of a randomly selected document object in *5Newsgroups* during EM

Iteration	$\omega_{.0}$	$\omega_{.1}$	$\omega_{.2}$	$\omega_{.3}$	$\omega_{.4}$
1	1	3.96E-071	7.39E-130	1.13E-033	1.01E-166
2	4.04E-123	2.68E-094	0	1	0
3	0	4.91E-118	0	1	0
4	0	1	0	3.57E-028	0
5	0	1	0	1.78E-288	0
6	0	1	0	0	0
7...end	0	1	0	0	0

5Newsgroups contains poorly distinguishable clusters, the maximum probability rushes toward an absolute 1 just within the first few cycles, where the others are all zeros, showing a kind of crisp assignment. Even when the object is relocated to another cluster, showing by the changes in probability values, the transition is rough and sudden. Probabilities change their values from 0 to 1, and vice versa, after just one iteration. So, changeover is no longer smoothly, and the assignment of documents into clusters is not “soft”, but rather “hard” always.

Besides, what happen if the clustering falls under the case described above in Figure 3.2? After just the first iteration, the document is quickly assigned to a particular cluster (in this case, cluster 0) with such a high probability. This is, however, a wrong assignment, which is clearly shown by the changes in the next iterations. If the first iteration is a result of a bad initialization, and the distance between true clusters is large enough as demonstrated in Figure

3.2, this document may as well be stuck there, and never be re-assigned to its correct cluster. With the effect of very high dimensionality, bad initializations will mostly mislead into serious loss.

3.5 MMDD Feature Reduction

3.5.1 The Proposed Technique

Our philosophy is that if documents can be viewed as directional data, so can words in the current context. Attributes of a word data point will be its frequencies of appearance in the documents. We apply mixture model of vMF distributions for clustering on the word space. The result is a set of mean vectors, each of which potentially represents a group of words of the same topic. A projection matrix $\mathbf{A}$ is then formed, by calculating the cosine distance of each pair of word and mean vector. Hence, after the linear transformation, documents in the reduced-dimension space have the number of attributes equal to the number of potential topics in the document corpus. Our purpose is to find a projection matrix $\mathbf{A}$ to transform the text data into the new latent space. Given a document corpus, words appearing in the collection should form different small groups of sub-topics or semantic meanings. Therefore, if the documents can be represented in term of their contribution towards these sub-topics, the dimension can be significantly reduced to as low as the number of sub-topics there are.

Mixture of directional distributions, more particularly vMF distributions, has been proven to give good document clustering [103, 113]. In such circumstances, clustering is performed on documents, which are represented as unit vectors, and have words as their attributes. We now carry out clustering on words. Each word is expressed as a vector with its frequencies in the documents as its attributes. Term Frequency-Inverse Document Frequency technique [115] is applied before normalizing the vector to unit length. After the clustering process, words are assigned into different sub-groups. We assume that words belonging to the same group have a common semantic meaning in the current context, and can be represented by the mean vector of that group. How important a word is determined by its relationship with this mean vector. In our study, we use the cosine of two vectors to measure this relationship.

Let r be the number of sub-topics, which is known a priori. Our FR technique is summarized as following:

- Step 1: Considering words as unit random vectors $\mathbf{W} = \mathbf{w}_1, \dots, \mathbf{w}_d$, and using a r -component mixture model of vMF distributions, which have been discussed in Section 3.2.3, we divide the word space into r sub-groups, each of which has a mean vector $\boldsymbol{\mu}_j$ ($j= 1, \dots, r$).
- Step 2: Projection matrix $\mathbf{A}_{d \times r}$ is created. Element a_{ij} of matrix $\mathbf{A}$ is the weight of word i with respect to sub-topic j , determined by cosine distance between word $\mathbf{w}_i$ and mean vector $\boldsymbol{\mu}_j$, for $i= 1, \dots, d$ and $j= 1, \dots, r$:

$$a_{ij} = \mathbf{w}_i^T \boldsymbol{\mu}_j \quad (3.13)$$

- Step 3: After its creation, matrix $\mathbf{A}$ is used to project the documents into r -dimensional space. Let $\mathbf{X}$ denote the original word-document matrix, i.e. $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_n]$, where $\mathbf{x}_i = [\mathbf{x}_{i1} \mathbf{x}_{i2} \dots \mathbf{x}_{id}]^T$ represents a document in Vector Space Model. If using $\mathbf{Y}$ as the new attribute-document matrix in r -dimensional space, we can determine $\mathbf{Y}$ by:

$$\mathbf{Y} = \mathbf{A}^T \mathbf{X} \quad (3.14)$$

The result of this procedure is a r -by- n matrix $\mathbf{Y}$, whose columns correspond to new document vectors with only r attributes. The document vectors are then to be used as input into clustering systems to perform categorization.

3.5.2 Experimental Results

We compared our FR technique with DF, TC and LSA. They are first applied to the datasets, which are then clustered by the same algorithm. The feature reduction techniques are subsequently evaluated by their corresponding clustering results. The vMF mixture model-based algorithm is chosen for the clustering task. It has been known that mixture model-based clustering algorithms are sensitive to initialization. Hence, in order to reduce bad initialization effect, experiment is repeated 20 times on each dataset. The clustering results are then sorted, and average value of the top 10 results is calculated. Besides, it has been reported that FR techniques such as LSA normally produce their best results when dimension is reduced to around 100. Therefore, our experiment study is carried out in a range from 10 to 300 number of dimension, with interval of 10.

Beside comparison among FR techniques, we also compare them against clustering alone (without FR) to see how useful and robust they are.

Fig. 3.3. Clustering results of dataset *reuters10*

Fig. 3.4. Clustering results of dataset *fbis*

Figures 3.3 to 3.6 show the clustering results on 4 datasets *reuters10*, *fbis*, *tr45* and *webkb4* (refer to Section 2.5 for the details of these datasets) respectively. We temporarily use “M2FR” to denote our Mixture Model-based FR technique in the figures. It can be seen that DF and TC perform very poorly compared to M2FR and LSA. However, their clustering quality gradually improves as the dimension increases. It shows that in the DF, TC and FS techniques, dimensions cannot be reduced to too low a level without affecting clustering quality. For dataset *reuters10*, reducing dimension to below 220 by DF or 280 by TC leads to empty documents.

Fig. 3.5. Clustering results of dataset *tr45*

Fig. 3.6. Clustering results of dataset *webkb4*

For dataset *reuters10*, our technique out-performs LSA quite significantly. It is slightly better than LSA with *webkb4*. For *fbis* and *tr45*, its clustering quality is a bit worse than that of LSA, although the difference is not significant. Generally, with 40 features onwards for *reuters10*, or 20 features and above for the other cases, good clustering quality can be ensured with M2FR.

Furthermore, in Table 3.7, we compare the above results against the results of clustering without any FR techniques. Clustering without FR is also repeated 10 times at each dataset to produce an average NMI value. The first two columns of values record the best average NMIs, and the numbers of features at which

Table 3.7
Comparison between clustering results with and without M2FR technique

Datasets	With M2FR		Without FR	
	Dimension	NMI	Dimension	NMI
<i>reuters10</i>	250	0.674	7906	0.592
<i>fbis</i>	250	0.582	2000	0.586
<i>tr45</i>	20	0.727	8261	0.710
<i>webkb4</i>	120	0.428	10921	0.397

they are achieved with our FR method. They are compared with the average NMI values obtained with the original numbers of features. The table shows that except only *fbis* having a very small degradation, all other cases have better clustering results, especially *reuters10* with more than 13.8% improvement and more than 96.8% reduction in number of features. Therefore, M2FR can significantly reduce a dataset’s dimension, while excellent clustering quality is still guaranteed.

The algorithm presented above is considered as a FR technique for text documents. Mixture of directional distributions vMF are applied to transform the word dimension into a lower latent subspace based on grouping of the words. Subsequently, vMF mixture model are applied on the documents in the reduced-dimension subspace to achieve good document clustering. Hence, although we treat MMDD as a method of FR, groupings are applied on both word dimension and document dimension. From this perspective, one can relate our algorithm with co-clustering [7, 63, 64], in which words and documents are clustered simultaneously in one process. The difference is, here, words are clustered and transformed first in a complete and separate step. It is more of the same type with FR techniques like DF, TC and LSA.

3.6 Enhanced EM Initialization for Gaussian Model-based Clustering

3.6.1 DA Approach for Model-based Clustering

In the situation where no pre-processing techniques, such as the feature reduction proposed above, are carried out, and when Gaussian, instead of a more robust model, is used as the underlying probabilistic model for document clustering,

it is difficult to obtain the best possible quality results. As mentioned earlier, the cluster membership calculation is not reliable during the first few cycles of EM. The main goal of the DA-based approaches to extend EM in model-based clustering is to reduce the effect of posterior probabilities, calculated by Eq. (3.17), upon the estimation of model parameters by Eq. (3.18). In very high-dimensional domains, such as document clustering, this is even more crucial since soft assignments and smooth transition no longer exist. Hence, the key point is to prevent the data objects from either refusing or binding to any cluster completely (i.e. with probability 0 or 1) at early stage of learning the mixture. We will do just that by controlling the volume of the ellipsoids in Gaussian model.

Let us recall from Section 2.2.7 that the objective of model-based clustering is to maximize the log-likelihood function:

$$\log L(\mathbf{X}, \mathbf{Z}|\Theta) = \sum_{i=1}^n \sum_{j=1}^k P(z_i=j|\mathbf{x}_i) \log \{\alpha_j f(\mathbf{x}_i|\theta_j)\} \quad (3.15)$$

where n is the number of data objects $\mathbf{x} \in \mathbb{R}^d$, k is the number of clusters as well as mixture components, α_j 's are the mixture weights, $\sum_j \alpha_j=1$, and $f(\mathbf{x}|\theta_j)$, $j=1, \dots, k$, are the density functions, defined by parameter set θ_j , corresponding to the mixture components. $\Theta = \{\alpha_j, \theta_j\}_{j=1, \dots, k}$ is the set of all parameters to be estimated. $\mathbf{Z} = \{z_1, \dots, z_n\}$ are the label variables; $z_i=j$ indicates that $\mathbf{x}_i$ is generated from component j . Therefore, cluster assignments are soft assignments based on the posterior probabilities $P(z_i=j|\mathbf{x}_i)$. In the case of Gaussian distribution, the density function of component j in the mixture is:

$$f(\mathbf{x}|\theta_j) = \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right\} \quad (3.16)$$

in which $\theta_j = \{\boldsymbol{\mu}_j, \Sigma_j\}$, $\boldsymbol{\mu}_j$ is the d -dimensional mean vector and Σ_j is the $d \times d$ covariance matrix. Applying EM to maximize Eq. (3.15) consists of repeating the following two steps:

$$P(z_i=j|\mathbf{x}_i) = \frac{\alpha_j \cdot f(\mathbf{x}_i|\theta_j)}{\sum_{l=1}^k \alpha_l \cdot f(\mathbf{x}_i|\theta_l)} \quad (3.17)$$

$$\hat{\Theta}^{\text{new}} = \arg \max_{\Theta} \sum_{i=1}^n \sum_{j=1}^k P(z_i=j|\mathbf{x}_i) \log \{\alpha_j f(\mathbf{x}_i|\theta_j)\} \quad (3.18)$$

From Eq. (3.17), it is always satisfied that $\sum_{j=1}^k P(z_i=j|\mathbf{x}_i) = 1$, $P(z_i=j|\mathbf{x}_i) \in$

$[0, 1]$, allowing soft memberships. However, the calculation in Eq. (3.17) in the early cycles of EM depends greatly on the initialization of Θ , which is unreliable and can lead to poor local optimum. To overcome this problem, Ueda and Nakano applied the maximum entropy principle [107]:

$$\max H = - \sum_{i=1}^n \sum_{j=1}^k P(z_i=j|\mathbf{x}_i) \log P(z_i=j|\mathbf{x}_i) \quad (3.19)$$

This entropy constraint is incorporated into the objective Eq. (3.15). The aim is to increase the randomness of cluster assignments by enforcing equality among the posterior probabilities. As a result, the updating Eq. (3.18) still remains intact, but Eq. (3.17) is changed to:

$$P(z_i=j|\mathbf{x}_i) = \frac{\{\alpha_j \cdot f(\mathbf{x}_i|\theta_j)\}^\beta}{\sum_{l=1}^k \{\alpha_l \cdot f(\mathbf{x}_i|\theta_l)\}^\beta} \quad (3.20)$$

where β is the temperature parameter. In DAEM, β is initialized to a small value at first, $0 < \beta < 1$, and gradually increased $\beta_{\text{new}} = \beta_{\text{current}} \times c$, where constant parameter c is normally set from 1.1 to 1.5 according to the authors. Equations (3.20) and (3.18) are alternatively applied until convergence to update the model estimates at each temperature $1/\beta$. When β reaches 1, DAEM coincides with the original EM, and the algorithm stops.

Similarly, Zhong and Ghosh [96, 103] derived a DA framework for model-based clustering by adding entropy constraints to the log-likelihood function. However, they arrived at a slightly different updating formula for the posterior probabilities:

$$P(z_i=j|\mathbf{x}_i) = \frac{\alpha_j \cdot f(\mathbf{x}_i|\theta_j)^{1/T}}{\sum_{l=1}^k \alpha_l \cdot f(\mathbf{x}_i|\theta_l)^{1/T}} \quad (3.21)$$

where parameter T is the temperature, equivalent to $1/\beta$ in DAEM. Zhong and Ghosh have applied the DA versions of Bernoulli, multinomial and von Mises-Fisher models for document clustering. Their study shows that DA improves significantly the clustering quality of these models. However, the quality improvement comes with a trade-off of higher computational cost.

3.6.2 The Proposed EM Algorithm

Generally, when Gaussian mixture model is considered for document clustering, the covariance matrix is usually assumed to be in spherical or diagonal form. On the one hand, the high dimensionality of such data makes the number of

parameters in non-constrained Gaussian model very large, causing high computational demand. Singular covariance estimates are often encountered when the dimension is greater than the number of data objects, which is usually the case in document clustering. On the other hand, the sparseness of text data makes it reasonable enough to assume spherical or diagonal model. They work relatively well while requiring fewer parameters compared to non-constrained model. The covariance matrix of a spherical Gaussian component j is $\Sigma_j = \text{diag}(\sigma_j^2)$, where σ_j^2 is the variance. It represents the dispersion estimate of a cluster. It defines an ellipsoid which covers approximately the neighborhood of data objects that belong to the cluster. Our heuristic approach is then based on these matrices. At the beginning of EM, we force the coverage from all the ellipsoids to be large, so that data objects remain available to most, if not all, of the clusters. This is achieved by replacing Σ_j by $\tilde{\Sigma}_j = \text{diag}(\sigma_j^2 + \sigma_t^2)$, $\forall j=1, \dots, k$ in the calculation of the posterior probabilities:

$$P(z_i=j|\mathbf{x}_i) = \frac{\alpha_j \cdot f(\mathbf{x}_i|\{\boldsymbol{\mu}_j, \tilde{\Sigma}_j=\text{diag}(\sigma_j^2 + \sigma_t^2)\})}{\sum_{l=1}^k \alpha_l \cdot f(\mathbf{x}_i|\{\boldsymbol{\mu}_l, \tilde{\Sigma}_l=\text{diag}(\sigma_l^2 + \sigma_t^2)\})} \quad (3.22)$$

where σ_t^2 is a value decreasing over time. At the first iteration, σ_t^2 is initialized to a relatively large value σ_{max}^2 . In document clustering, documents are usually represented by L_2 -normalized unit vectors. Hence, it is reasonable to have $0 \ll \sigma_{max}^2 < 1$. As EM proceeds, we compress the volume of the ellipsoids by gradually decreasing σ_t^2 through the formula $\sigma_{t,new}^2 = \sigma_{t,old}^2 \times c$, where $0 < c < 1$. Our modified EM algorithm is described in Fig. 3.7.

According to Fig. 3.7, σ_t^2 creates an annealing effect during step 2. This step can be considered as a smoothened initial process, where the model parameters are estimated as usual, but the posterior probabilities are changed very slowly. When σ_t^2 is reduced to a value too small to have an impact, the algorithm switches to the standard EM in step 3.

3.6.3 Experimental Results

There are two separate comparisons carried out in this section. Firstly, we compare our algorithm in Fig. 3.7 against Gaussian models with standard EM and DAEM. The experiments were designed so that identical set of initial parameters was always used among the three methods. Secondly, we compare our algorithm with models of multinomial mixture (*mixmnl*s), DA multinomial (*damnls*), von Mises-Fisher mixture (*softvmfs*), vMF with DA (*davmfs*) and CLUTO

1. Initialize Θ , set c and $\sigma_t^2 \leftarrow \sigma_{max}^2$ ($0 < c, \sigma_{max}^2 < 1$)
2. Iterate the following modified EM steps,
until $\sigma_t^2 < \min\{\sigma_j^2\}_{j=1,\dots,k}$:
 - (a) Update posterior probabilities by Eq. (3.22)
 - (b) Update model parameters by Eq. (3.18)
 - (c) Decrease σ_t^2 by $\sigma_{t,new}^2 \leftarrow \sigma_{t,old}^2 \times c$
3. Iterate the standard EM steps, Eq. (3.17) and Eq. (3.18),
until convergence
4. $\forall \mathbf{x}_i, z_i = \arg \max_j P(z_i=j|\mathbf{x}_i), j = 1, \dots, k$

Fig. 3.7. Enhanced EM for spherical Gaussian model-based clustering

clustering. In each experiment, the Gaussian model-based algorithms were run 50 times, each time with a random initialization, to get the average and standard deviation of NMI score. Aitken acceleration-based stopping criterion [44] were used, with maximum 600 iterations in a complete EM process. NMI results by using CLUTO and the other models were reported as in the experiments done by Zhong and Ghosh [96].

When document vectors are normalized to have unit length, it is reasonable to initialize the variance parameter σ_t^2 in our algorithm with $0 \ll \sigma_{max}^2 < 1$. In all the experiments, we used $\sigma_{max}^2 = 0.1$ and $c = 0.8$. The temperature parameter of DAEM was set as in its paper [107]: $\beta_{min} = 0.5, \beta_{new} \leftarrow \beta_{current} \times 1.2$.

Table 3.8 presents the clustering results on 6 datasets *classic3*, *classic300*, *cranmed*, *A2*, *B4* and *tr12* (refer to Section 2.5 for the details of these datasets). Based on NMI evaluation, it is clearly shown that the proposed technique improves Gaussian model's performance significantly in this document clustering problem. While DAEM yields slightly better results than standard EM in some of the cases, our algorithm always gives the highest NMI scores with good margins from their score values. A similar outcome is obtained when using *Purity* as clustering evaluation metric. As displayed in Fig. 3.8, our algorithm yields better cluster purities than the other two algorithms on the given datasets.

We also report in Table 3.8 the clustering times taken by the algorithms. Since 50 repeated runs gave quite different time values, taking the average of all of them would not give a correct measurement. Instead, we calculated the average time of the shortest 20 out of 50 clustering trials. For *classic3*, *cranmed*, *A2*

Table 3.8
NMI results & clustering time by 3 Gaussian models

Data	EM	DAEM	Our algorithm
<i>classic3</i>	0.74 ± 0.10 2.66s	0.74 ± 0.10 9.70s	0.84 ± 0.00 4.97s
<i>classic300</i>	0.85 ± 0.10 0.06s	0.87 ± 0.08 0.14s	0.94 ± 0.00 0.22s
<i>cranmed</i>	0.68 ± 0.13 0.86s	0.68 ± 0.14 9.14s	0.86 ± 0.00 1.48s
<i>A2</i>	0.51 ± 0.21 0.03s	0.59 ± 0.21 0.06s	0.74 ± 0.01 0.04s
<i>B4</i>	0.23 ± 0.06 0.15s	0.29 ± 0.07 0.65s	0.45 ± 0.03 0.14s
<i>tr12</i>	0.49 ± 0.06 0.39s	0.49 ± 0.06 1.05s	0.66 ± 0.04 0.83s

Fig. 3.8. Clustering results in *Purity*. Top-to-bottom in legend corresponds to top-to-bottom in the plot.

Table 3.9

NMI results: Gaussian models compared with CLUTO and other probabilistic models

Data	EM	DAEM	mixmnls	damnls	softvmfs	davmfs	CLUTO	Our algorithm
<i>ohscal</i>	.38 ± .02	.38 ± .02	.37 ± .02	.39 ± .02	.44 ± .02	.47 ± .02	.44 ± .02	.38 ± .02
<i>hitech</i>	.26 ± .03	.27 ± .03	.23 ± .03	.27 ± .01	.29 ± .01	.30 ± .01	.33 ± .01	.31 ± .02
<i>k1b</i>	.57 ± .04	.57 ± .04	.56 ± .04	.61 ± .04	.60 ± .04	.67 ± .04	.62 ± .03	.63 ± .04
<i>tr11</i>	.52 ± .05	.52 ± .04	.39 ± .07	.61 ± .02	.60 ± .05	.66 ± .04	.68 ± .02	.66 ± .03
<i>tr23</i>	.32 ± .06	.32 ± .06	.15 ± .03	.31 ± .03	.36 ± .04	.41 ± .03	.43 ± .02	.44 ± .03
<i>tr41</i>	.60 ± .04	.61 ± .04	.50 ± .03	.61 ± .05	.62 ± .05	.69 ± .02	.67 ± .01	.64 ± .03
<i>tr45</i>	.58 ± .04	.58 ± .04	.43 ± .05	.56 ± .03	.66 ± .03	.68 ± .05	.62 ± .01	.71 ± .04

and *tr12*, our algorithm required more time than EM, but considerably less than DAEM. In general, this is expectedly so. Let us call I the number of iterations EM need to complete, and assume that step 3 in Fig. 3.7 as well as DAEM at each value of β require the same I to converge. Then, the total number of iterations needed by our algorithm is approximately $I + \log_c(\sigma_{min}^2/\sigma_{max}^2)$, where σ_{min}^2 is a relatively small value, while that number for DAEM is $I \times \log_c(1/\beta_{min})$. For *classic300*, our algorithm took the longest time. However, it steadily needed the same amount of time for all 50 runs to produce a good and consistent result. For dataset *B4*, it spent even less time than EM to provide much better clustering. As EM, DAEM and our algorithm were always initialized with identical set of parameters, this case shows that step 2 of the proposed algorithm in Fig. 3.7 must have helped step 3 converge faster than standard EM would.

In the second experiment, we evaluate the three Gaussian models with the popular clustering toolkit CLUTO and other probabilistic models. The *NMI* results are shown in Table 3.9 for another set of 6 datasets *ohscal*, *hitech*, *k1b*, *tr11*, *tr23*, *tr41* and *tr45* (refer to Section 2.5 for the details of these datasets). The first observation from the Table is that, similar to previous experiments, the modified EM proposed for Gaussian model-based clustering continues to provide better results than EM and DAEM. The only exception is dataset *ohscal*, on which all three Gaussian models yield the same result. What is more, for all the 6 datasets being examined here, the proposed algorithm helps Gaussian model improve its clustering performance to become even better than *mixmnls*, *damnls* and *softvmfs*. In previous studies, these models have been suggested to be more suitable for document clustering than Gaussian. Besides, our algorithm is very comparable to CLUTO and *davmfs*. It obtains the highest *NMI* scores when tested on *tr23* and *tr45*. The results on these two datasets are also illustrated by Fig. 3.9. On the other datasets, as shown in Table 3.9, our

Fig. 3.9. Clustering results in *NMI* on datasets *tr23* and *tr45*. Top-to-bottom in legend corresponds to left-to-right in the plot.

algorithm is either better than one of them or very close to both CLUTO and *dvmfs*. Finally, since the deterministic annealing procedure in *damnls* and *dvmfs* follow the same framework as in DAEM, it can be expected that these two methods also require more computational time than our algorithm, as we have discussed in the previous paragraph for the case of DAEM.

3.7 Conclusions

In this chapter, theoretical and empirical analysis have been carried out for the mixture model-based clustering approach, and two techniques for improving related mixture model-based algorithms have been proposed. Firstly, by empirical experiments, we have testified the feasibility of applying probabilistic mixture model on the clustering problem for sparse and high-dimensional data. Mixture model-based clustering (M2C) methods of two types of distribution, Gaussian and vMF, have produced clustering results of comparative quality to other well-known methods, such as the *k*-means variants and the recently proposed NMF. Especially, the directional distribution vMF has shown some dominant and promising performances.

We have also performed an analysis on the impacts of high dimensionality on various characteristics of M2C. Some successful model selection methods, which have been well-designed and used on lower-dimensional domain, fail easily when applied on text documents. The favorite soft-assignment characteristic of M2C does not really exist on the sparse and so high-dimensional space. The sensitiveness to initialization, which is already a problematic issue in lower dimension, becomes even more critical and hard to handle with. Understanding

all these breakdown points is extremely helpful for us in the research for a better approach to the unsupervised text classification problem.

Besides the comparative study and analysis of the algorithms, we have proposed two novel methods which aim to address two problems often encountered when using mixture models for high-dimensional data clustering. In the first problem, we have presented a technique for reducing document's dimension using mixture model of directional statistics. A mixture of von Mises-Fisher distributions is utilized to decompose the word space into a set of sub-topics, which are represented by their mean vectors. A projection matrix is then created based on word-to-mean cosine measure. Through this matrix, the document corpus is transformed into a new feature space of much lower dimension. Experimental results have shown that our proposed method improves document clustering quality. It is very comparable with LSA, and is better than LSA in some cases.

Since it is built on top of mixture model-based method, however, our technique encounters some familiar drawbacks. Firstly, it is the well-discussed sensitiveness to initialization. We would like to emphasize again the importance of 1) having a stable initialization scheme; 2) reducing the sensitiveness of this model itself. Secondly, the number of sub-topics must be predefined. This equals to the number of document's features left after the FR technique is performed. So, this is a problem our method and LSA have in common. Researchers working on mixture model-based clustering framework have proposed a few methods for automatically determining the number of mixture components. We have discussed this issue in sections 2.3.2 and 3.4.1. We have also noted that no model selection methods have proven effective performance on text data. Hence, if such an approach can be made feasible and combined with our technique, the question of finding the optimal number of features for document vectors is also resolved.

In the second problem, we have presented an annealing-like technique to improve the initial phase of EM algorithm when it is applied to high-dimensional Gaussian model-based clustering. In our approach, the ellipsoids of the Gaussian components are forced to remain large during the early stage of EM by a variance parameter. This helps to make the data objects remain available to all the clusters for the initial period of iterations, while model parameter estimates are being refined to be more reliable. Then, the ellipsoids are compressed to decrease their boundaries as the variance parameter gradually decreases. This creates an annealing effect that makes the transitions of data objects among clusters become smoothly.

Despite the fact that this approach seems heuristic, it offers great efficiency in document clustering applications. Numerical experiments show that our proposed EM algorithm for Gaussian model significantly outperforms the original EM and its deterministic annealing variant DAEM. It also has the advantage over DAEM in terms of shorter computational time. Moreover, it makes the performance of Gaussian model become more comparable to multinomial, von Mises-Fisher models and CLUTO, which have previously been deemed more suitable for document clustering.

Chapter 4

Robust Mixture Model-based Clustering with Genetic Algorithm Approach

4.1 Overview

In recent years, data clustering has become one of the most useful and important activity in data mining and analysis. The amounts of data have nowadays become tremendous, generated from diversely different application domains. However, one aspect of data clustering that needs to be studied more thoroughly is the effects of atypical observations, or outliers, to the quality and accuracy of a clustering result. Many efforts have been focused on developing and improving algorithms which mainly perform on non-contaminated datasets, whereas outlier problems in data clustering has now started to attract increasing attention. On the other hand, outliers often exist in data. They can arise due to many reasons, such as sampling errors, inaccurate measurements, uninteresting anomalous observations, distortions and so on. Robust clustering methods are in fact needed to take care of such uncertainty in data, so that results of complicated and costly cluster analyses do not become wasteful, or even misleading.

Many clustering methods rely on some distance metrics to determine cluster assignment for data observations. Such an example is Mahalanobis distance in equation (4.1). It measures the distance between a multivariate observation $\mathbf{x}_i$ and the estimated location $\boldsymbol{\mu}_j$ of data cluster j , with respect to its estimated covariance matrix Σ_j . The location and covariance estimates here can be, for

example, maximum likelihood estimates.

$$D_i(\boldsymbol{\mu}_j, \Sigma_j) = \sqrt{(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)}, \quad (4.1)$$

However, if outliers exist in the data, they can affect a cluster's location estimate, by attracting the estimate toward their location, and far away from the true cluster's location. Outliers can also inflate the covariance estimate in their direction. For those reasons, D_i value for an outlier may not necessarily be large, and that outlier can be viewed as a member of the cluster. This is called the “masking” effect, as the presence of some outliers mask the appearance of another outlier. On the other hand, D_i value of certain non-outlying observation may possibly become large, hence makes it misclassified as atypical to the cluster if based on this criterion. This second effect is called “swamping”. As a result, it is difficult to distinguish between typical data observations and outliers. The parameter estimates are inaccurate, and the clustering result is neither quality nor reliable.

Research related to outliers in multivariate data is not a new topic. More than a few methods have been proposed for estimation of data location and dispersion in the presence of outliers. Some examples are the minimum volume ellipsoid (MVE) and minimum covariance determinant (MCD) estimators [116, 117], M-estimators [118], S-estimators [119] and a number of robust regression methods. For a more up-to-date and complete review of this area, readers can refer to [120, 121]. Another approach, which is more related to probabilistic model, is the trimmed likelihood estimator first suggested by Neykov and Neytchev [122], and further investigated by Hadi and Luceño [123]. Recently, researchers have started to look into merging these robust analysis methods into classification [124] and clustering [125]. However, this research area is not very well-studied yet. Hence, the purpose of this part of our research is to make further improvement in developing clustering algorithms which are robust to outliers, particularly in the case of probabilistic mixture model-based clustering approach.

Probabilistic mixture model has been a well known approach to cluster analysis. However, as they rely on maximum likelihood estimation (MLE), the algorithms are often very sensitive to noise and outliers. In this Chapter, we address the robustness issue of maximum likelihood based methods. We implement a variant of the classical mixture model-based clustering (M2C), following a proposed general framework for handling outliers. Genetic Algorithm (GA) is incorporated into the framework to produce a novel algorithm called GA-based

Partial M2C (GA-PM2C). Analytical and experimental studies show that GA-PM2C can overcome the negative impact of outliers in data clustering, hence provides highly accurate and reliable clustering results. It also exhibits excellent consistency in performance and low sensitivity to initializations.

The structure of this Chapter is as follows. In section 4.2, we review classical mixture model-based clustering (classical M2C). We discuss how outliers affect its performance, then introduce a new framework in order to integrate robustness into M2C. Next, in section 4.3, a novel clustering algorithm based on the proposed framework is presented. Empirical experiments in section 4.4 show the performance of our proposed algorithm with comparison to existing methods. Finally, conclusions and future work are given in section 4.5.

4.2 M2C and Outliers

4.2.1 Classical M2C

Finite mixture model is an approach to data modeling with strong statistical foundation. It has been widely applied to a variety of data in the field of cluster analysis [43]. In M2C, data are assumed to be generated from a mixture of probability distributions. Let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$ be a random sample of size n . We say $\mathbf{x}_i$ follows a k -component finite mixture distribution if its probability density function can be written in the form:

$$f(\mathbf{x}_i|\Theta) = \sum_{j=1}^k \alpha_j f_j(\mathbf{x}_i|\theta_j) \quad (4.2)$$

where each f_j is a density function- a component of the mixture. Quantities $\alpha_1, \dots, \alpha_k$ are mixing probabilities ($\alpha_j \geq 0, \sum_{j=1}^k \alpha_j = 1$). θ_j denotes a set of parameters defining the j^{th} component, and $\Theta = \{\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k\}$ denotes the complete set of parameters needed to define the mixture. It is normally assumed that all the components f_j have the same functional form, e.g. Gaussian distribution.

For more review of the general M2C framework, readers can turn back to Section 2.2.7; and for particular types of mixture model, Section 3.2. For even further literature on mixture model, as well as M2C, of Gaussians and also other types of probabilistic distribution, readers can refer to [43–45]. Gaussian M2C contributes an important part in data clustering field. Some recent research works continue to show its useful applications in high-dimensional data cluster-

Fig. 4.1. Classical Gaussian M2C on the original dataset (a) and contaminated dataset (b). The contours are 95% ellipsoids of the Gaussians: thick lines represent true partitions; dashed lines are results from classical Gaussian M2C.

ing [108] and feature selection [109], gene microarray data clustering [110] or image segmentation [84].

Classical MLEs, and hence M2C methods, always try to fit the entire set of data presented to them. When noise, outliers or atypical observations exist in the data, they could produce inaccurate results, since estimates of means and covariance matrices based on equations (3.2) and (3.3) are not robust enough to handle such a case. For illustration, we consider an example of a mixture of three bivariate Gaussians. This dataset is similar to the simulated dataset discussed in [44]. It consists of 100 samples generated from a 3-component bivariate normal mixture with equal mixing probabilities and component parameters as:

$$\boldsymbol{\mu}_1 = (0 \ 3)^T, \ \boldsymbol{\mu}_2 = (3 \ 0)^T, \ \boldsymbol{\mu}_3 = (-3 \ 0)^T$$

$$\Sigma_1 = \begin{pmatrix} 2 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}, \ \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ \Sigma_3 = \begin{pmatrix} 2 & -0.5 \\ -0.5 & 0.5 \end{pmatrix}$$

An additional set of 50 outliers, generated from a uniform distribution within $[-10,10]$ on each dimension, is added to the original data to form a new contaminated set of 150 samples. As shown in Fig. 4.1 and Table 4.1, classical M2C with Gaussian components performs well on the former, but fails to yield correct result on the latter data.

Table 4.1

Confusion matrices resulted from classical Gaussian M2C (added outliers are not shown)

	Original dataset			Contaminated dataset		
	cluster1	cluster2	cluster3	cluster1	cluster2	cluster3
class1	31	2	0	2	31	0
class2	0	33	0	33	0	0
class3	1	0	33	6	25	3

4.2.2 Toward Robustness in M2C

There have been a few ideas proposed to deal with noise and outliers under probabilistic mixture model-based framework. Banfield and Raftery [74] introduced an additional component- a uniform distribution- into the mixture of Gaussian distributions to account for the presence of noise in data. McLachlan and Peel [44] used a t -mixture model to reduce outlier effect. However, according to Hennig [75], while providing a certain gain of stability in cluster analysis, these approaches do not prove a substantial robustness to outliers. Another approach is to employ Forward Search technique [73, 76, 77]. A Forward Search-based method starts by fitting a mixture model to a subset of data, assumed to be outlier-free. The rest of the data are then ordered based on some metric, e.g. Mahalanobis distance, with regarding to the fitted model. Next, the subset is updated by adding into it the “closest” sample. The search goes on by repeated fitting and updating until all the samples are included.

More lately, volume-based clustering algorithms were proposed [125]. These are examples of the combination of a robust estimator, the minimum volume ellipsoid (MVE) introduced by Rousseeuw and Leroy [117], into clustering. Basically, they extend the application of MVE from robustly fitting a single group of data to clustering a mixture of groups of data. One drawback of MVE, though, is its high computational complexity and low rate of convergence. Cuesta-Albertos et al. [126] approached the problem in the opposite direction. They made use of clustering method to provide estimate of normal mixture model. Firstly, a trimmed k -means [127] was applied to find the core of the data clusters. This was treated as the initial trimming. Then, the trimmed region was step-by-step expanded, with ML estimation performed at each of the trimmed levels. As described, the procedure is greatly dependent on the clustering method used at the initial stage.

Upon using a M2C method for clustering data, we intuitively agree to the following assumption:

Fig. 4.2. Partial mixture model-based clustering.

The given data are identically and independently distributed observations from a true mixture of probabilistic distributions.

We call this “the strong assumption”, since it describes a rigid approach to data modeling. It requires that all the data objects are i.i.d. observations generated from a particular mixture of distributions. This assumption is somewhat harsh. Fundamentally, it is unreasonable to expect this characteristic to always be true with real-life data. Therefore, we adopted what is called “the weak assumption” instead:

The given data are likely to be generated from a mixture of probabilistic distributions. Part of them, though, may not necessarily follow the mixture distribution.

A similar assumption called “the weak Gaussian assumption” was stated for the case of well-separated and spherical Gaussian mixture [105]. The weak assumption implies an imperfection in the data, meaning not all of the observations are i.i.d. under a mixture model. Some of them can be noise, some are outliers, or some of them simply can not conform to the particular mixture model of distributions. How large this fraction is within a given dataset depends on the nature of the data itself.

So how can “the weak assumption” be incorporated into M2C? The framework given in Fig. 4.2 is what we propose for such a purpose. We call it the Partial Mixture Model-based Clustering (Partial M2C), as only part of the data are assumed to follow mixture distribution (we call this part the model observations). The assumption leads to a subset selection step, where it is decided which data observations are to be included in the model, and which are not. ML estimation (MLE) is then carried out on the selected ones. If convergence has not been reached yet, the model observations are re-evaluated and re-selected,

until the most suitable group is found. At the end, the result is a set of clusters containing classified data, plus another group, simply labeled as “don’t-care”, containing potential noises and outliers.

There are two key issues in Partial M2C: 1) What should be the selection criterion in the Model Sample Selection stage? 2) How to make sure the EM’s monotonic property is preserved, or how to guarantee algorithm’s convergence? As long as convergence is guaranteed, any suitable objective function can be considered as selection criterion.

Neykov et al. proposed a method based on trimmed likelihood estimate (TLE) for robust fitting of mixtures [128]. They used an algorithm called FAST-TLE, which had previously been introduced for a single distribution [129], to find a subset of given data that fits the mixture model most in term of likelihood contribution. Firstly, a random subgroup of the given sample is used to fit the model. In subsequent iterations, a new subset of predefined size is selected based on previously estimated model, and then used to refine the model. FAST-TLE can be explained by the framework, since each of the algorithm’s refinement steps is equivalent to a cycle of Model Sample Selection and MLE of Partial M2C. It was also showed that the refinement procedure in FAST-TLE yielded monotonically nondecreasing sequence of log-likelihood, and since the number of subsets is finite, convergence is always guaranteed.

In the next section, we introduce a novel and robust clustering algorithm based on this Partial M2C framework. The proposed method, using Genetic Algorithm (GA) and TLE for Model Sample Selection, shows its effectiveness in overcoming noise and outlier problem in contaminated data.

4.3 GA-based Partial M2C

Genetic Algorithm (GA) [130] and its variants provide good selection methodologies. The GA’s reproduction and crossover processes involve evaluating a customized objective function, often called fitness function, and generate better solutions over generations. The formulation of this fitness function plays a key role in GA. In Partial M2C, this is where we can use GA to find the model observations. It will be explained in details in a few more lines. Besides, as mentioned above, when the model observations are re-selected, the likelihood value of ML estimates might not be monotonically nondecreasing anymore. When using GA, this can be prevented by always retaining in the next generation the formation of highest likelihood value from the current generation. Hence, we think GA could

be a suitable means to help us effectively search for the optimum set of model observations in Partial M2C. Some recent examples of integrating GA into clustering framework include: using GA to improve multi-objective clustering [131], reduce initialization sensitiveness [40], or help K-Means segment online shopping market effectively [132]. The proposed algorithm GA-based Partial M2C, or GA-PM2C, is given in Fig. 4.3.

Before going into the algorithm, a few parameters need to be declared as follows:

- n : total number of observations in the original data
- ϵ : assigned contamination rate, or trimming rate
- m : number of observations under probabilistic model, $m = (1 - \epsilon) \times n$
- G : maximum number of generations
- C : number of cycles when performing EM algorithm
- $P(t)$: parent population at time t
- $P'(t)$: offspring population at time t
- $|P|$: number of individuals in parent population
- $|P'|$: number of individuals in offspring population

Each individual in a population is represented by a chromosome, which is a binary vector of length n . The i -th bit of a chromosome is 1 if observation $\mathbf{x}_i$ is selected, 0 if $\mathbf{x}_i$ is considered outlier under the corresponding model. Attached to each chromosome is a Gaussian mixture modeling the selected data. Hence, each chromosome (and its corresponding mixture model) is a possible solution, showing two parts of the original data: observations belonging to the mixture model (i.e. the typical data) and “don’t-care” observations (i.e. outliers).

In Fig. 4.3, $P_i(t)$ to $P_{i+1}(t)$, or $P_i(t)'$ to $P_{i+1}(t)'$, represents the evolution of population $P_i(t)$, or $P_i(t)'$ respectively, from state i to state $(i + 1)$ due to certain process. The evaluation of the individuals in a population consists of three steps. Firstly, each individual goes through C cycles of EM, which will update the estimates of the model parameters attached to that individual. If EM converges faster, less than C cycles are needed. Besides, it is important to note that EM is only performed on the selected observations, corresponding to bits 1 of the individual. Secondly, the individuals undergo a process called Guided Mutation, which is explained in Fig. 4.4. Finally, their fitness values, $fScore$ ’s, are determined and stored for later comparison. In the following, we discuss our characterized GA-related operations used in the algorithm.

Guided Mutation: The original form of GA has three basic operators: Selection, Crossover and Mutation, which attempt to imitate the natural selection

```

1:  $t \leftarrow 0$ 
2: Initialize  $P(t)_0$ 
3: for  $iterate \leftarrow 1 : G$  do
4:    $P_1(t) \leftarrow$  perform  $C$  cycles of EM on  $P_0(t)$ 
5:    $P_2(t) \leftarrow$  Guided Mutation in  $P_1(t)$ 
6:    $fScore_2 \leftarrow$  evaluate  $P_2(t)$ 
7:    $P_0(t)' \leftarrow$  selection and crossover within  $P_2(t)$ 
8:    $P_1(t)' \leftarrow$  perform  $C$  cycles of EM on  $P_0(t)'$ 
9:    $P_2(t)' \leftarrow$  Guided Mutation in  $P_1(t)'$ 
10:   $fScore_2' \leftarrow$  evaluate  $P_2(t)'$ 
11:   $[P_3(t), fScore_3] \leftarrow$  select  $|P|$  individuals
    from  $\{[P_2(t), fScore_2], [P_2(t)', fScore_2']\}$ 
12:   $iBest \leftarrow$  best individual from  $P_3(t)$ 
13:  if  $iBest$  satisfies convergence condition then
14:    break
15:  end if
16:   $P_0(t+1) \leftarrow P_3(t)$ 
17:   $t \leftarrow t+1$ 
18: end for
19: Perform EM on  $iBest$  until convergence

```

Fig. 4.3. Algorithm: GA-PM2C

and genetic evolution in nature. However, under our problem formulation, we argue that Mutation is not helpful and powerful enough an engine. While occurring at a very low rate, in a random manner, not every mutation is a beneficial one. Hence, in this GA-based algorithm, we introduce another operator called Guided Mutation to replace the classical Mutation.

Guided Mutation applies on every individual during its development. This is where model observations are distinguished from potential outliers. According to Fig. 4.3, after their models are refined by some C cycles of EM, the chromosomes in a population are guided to mutate toward maximizing their fitness score values. In this study, we use the TLE function (4.3) as the GA's fitness function. Particularly, if A represents the model sample, it is a subset of size m out of n original observations. From (2.25), let $\log f(\mathbf{x}_i|\Theta)$ be the log-likelihood of $\mathbf{x}_i$ according to current model estimates. The objective is to maximize:

$$\log L_{TLE}(\mathbf{X}|\Theta) = \sum_{i=1}^n I_A(\mathbf{x}_i) \log f(\mathbf{x}_i|\Theta) \quad (4.3)$$

where $I_A(\cdot)$ is indicator function, $I_A(\mathbf{x}_i) = 1$ if $\mathbf{x}_i$ is included in the estimation ($\mathbf{x}_i \in A$), $I_A(\mathbf{x}_i) = 0$ if $\mathbf{x}_i$ is trimmed off, and $\sum_{i=1}^n I_A(\mathbf{x}_i) = m$. When using TLE as fitness function, a Guided Mutation is equivalent to one refinement step

Require: Chromosome A with $\log L_{TLE}(\mathbf{X}|\hat{\Theta}(t))$
Ensure: Altered chromosome A with $\log L_{TLE}'(\mathbf{X}|\hat{\Theta}(t)) \geq \log L_{TLE}(\mathbf{X}|\hat{\Theta}(t))$

```

1: procedure GUIDED-MUTATION
2:   for  $i \leftarrow 1 : n$  do
3:      $score_i = \log f(\mathbf{x}_i|\hat{\Theta}(t))$ 
4:   end for
5:   Sort  $score_{v(1)} \geq score_{v(2)} \dots \geq score_{v(n)}$ ,
   where  $v(1), \dots, v(n)$  are permutation of indices
6:   Set all bits in  $A$  to 0
7:   for  $i \leftarrow 1 : m$  do
8:     Set bit  $v(i)$ -th in  $A$  to 1
9:   end for
10:   $\log L_{TLE}'(\mathbf{X}|\hat{\Theta}(t)) = \sum_{i=1}^m score_{v(i)}$ 
11: end procedure

```

Fig. 4.4. Procedure: Guided Mutation

of FAST-TLE. Hence, our GA-PM2C nicely inherits the monotonic property proven for FAST-TLE [129]. For each chromosome, the log-likelihood is always nondecreasing during EM cycles, as already known, and also after Guided Mutation. Besides, since there is no random mutation affecting the best individual, as well as the rest of the population, at the end of each generation, the fittest individuals are carried unaltered to the next generation. The two characteristics above assure convergence of our GA-based algorithm.

Recombination: This process involves selecting potential pairs of parents and mating them to produce $|P'|$ offspring individuals. The size of offspring population can be determined based on a percentage p_o of the size of parent population, such that $|P'| = p_o \times |P|$. In our study, we use the standard techniques, roulette wheel rank weighting and single-point crossover [133].

Selection: The final operation in a cycle of the GA-based algorithm is to select $|P|$ individuals to carry to the next generation. The strategy of selection is that both the newly created offspring and the parents are considered. From the union of both the parent population $P_2(t)$ and the offspring population $P_2(t)'$, the $|P|$ best individuals are chosen to form the new generation $P_3(t)$.

For each generation $P_3(t)$, the best individual $iBest$ is identified and recorded to check for termination of GA. The process can be stopped by one of the following conditions: the maximum number of generations G is reached; or $iBest$ does not change within a certain number of consecutive generations. Once the GA evolution is terminated, a complete EM algorithm is performed one last time on the model of the best individual to make any possible improvement.

Normally, EM converges very fast at this time, possibly in 1 or 2 cycles, since the individual and its model have been ameliorated during the evolution process.

4.4 Empirical Study

The experiments below are used to examine and demonstrate the performance of GA-PM2C in cluster analysis of data with noise and outliers. Among various robust methods that have been discussed so far in the previous sections, FAST-TLE is the one most related to our algorithm. Hence, we will make a close comparison between the two throughout the experiments. Classical MLE, however, had been shown in other reports that it could not handle outliers [128]. Therefore it would not be necessary to include it in the comparison. Since we have been focusing on mixture of Gaussians, we continue to use this distribution model in the experiments. Other models, however, such as regression model or mixture of other distributions, are also applicable to our algorithm. Finally, our main objective in this work is to address robustness in cluster analysis, not model selection problem. We would not try to determine the number of clusters in the following experiments, but assume that this value is known a priori.

4.4.1 Parameter Setting

GA-PM2C requires some additional parameters, as declared in section 4.3, for the GA-based processes. The population size $|P|$, the number of EM cycles C and the assigned contamination rate ϵ affect the running time as well as efficiency of the algorithm. In each experiment, we varied the value of $|P|$ within a range to see the influence of this parameter. The number of EM cycles was set to 5 throughout the empirical study, since it has been verified that using a larger value or carrying out a complete EM algorithm does not lead to significantly better result. The assigned contamination rate specifies the amount of data observations being trimmed (trimming level). This was set at the true percentage of outliers of each dataset, and was also varied lower or higher around this true value to testify the robustness of the algorithms.

Besides, as pointed out by Neykov et al. [128], FAST-TLE should actually be run “finitely many times” after which the best solution is chosen. When comparing it with GA-PM2C, we followed the same procedure and repeated FAST-TLE as many times equal to $|P|$. So, in each trial, GA-PM2C was started with $|P|$ chromosomes, whereas FAST-TLE was run $|P|$ times simultaneously before its best outcome was recorded. The chromosomes in GA-PM2C and the

Table 4.2

Log-likelihood and success rates over 100 repetitions with $|P| = 4$

Algorithm	$\epsilon = 15\%$		$\epsilon = 25\%$		$\epsilon = 35\%$		$\epsilon = 45\%$	
FAST-TLE	-561.4 $\pm$ 2.6	92	-438.8 $\pm$ 0.09	100	-348.5 $\pm$ 0.4	100	-270.0 $\pm$ 1.9	100
GA-PM2C	-559.9 $\pm$ 0.5	100	-438.8 $\pm$ 0.01	100	-348.5 $\pm$ 0.2	100	-269.2 $\pm$ 0.8	100

Table 4.3

Confusion matrices resulted from GA-PM2C with $\epsilon = 0.35$

	cluster 1	cluster 2	cluster 3	outliers
class 1	1	30	0	2
class 2	30	0	0	3
class 3	0	0	28	6
outliers	3	3	2	42

subsamples in FAST-TLE were always initialized randomly. Finally, for EM algorithm, random initial assignment strategy and Aitken acceleration-based stopping criterion [44] were used. The maximum number of iterations in a complete EM process was 300 times.

4.4.2 Continue Experiment 4.2.1

Firstly, we revisit the dataset in Section 4.2.1 to see how the robust methods work on this, while classical model has failed. Table 4.2 records the results of 100 trials with $|P| = 4$. The average of log-likelihood and the number of times the algorithms successfully identify the three clusters are shown. From the table, it shows that GA-PM2C performs at least as well as FAST-TLE does on this dataset. The true contamination rate in this case is $\epsilon_0 = 33\%$. When the assigned contamination rate or trimming level ϵ is 15%, far below the true value, and $|P| = 4$ only, GA-PM2C does slightly better than FAST-TLE. It successfully identifies the three ellipsoid centers in all 100 trials, whereas FAST-TLE has 8 failures. When either ϵ or $|P|$ is set higher, the performance of FAST-TLE is improved. GA-PM2C with ϵ of 15%, 25%, 35% and 45% are shown in Fig. 4.5. FAST-TLE, once fits correctly, yields the same results as GA-PM2C does. At 25% or 35%, which are quite close to the true rate, the algorithms give excellent estimates of both means and covariances. The clustering result from GA-PM2C for the 35% case is shown in Table 4.3. When trimming is much lower, 15%, or much higher, 45%, the means are still determined correctly, but the covariances are estimated larger or smaller than the true values, because too many outliers have been considered as model samples, or too many true model samples have been pruned off respectively.

Fig. 4.5. GA-PM2C fits with ϵ at: (a) 0.15, (b) 0.25, (c) 0.35, (d) 0.45. The contours are 95% ellipsoids of the Gaussians: thick lines represent true partitions; thin lines are results from GA-PM2C; dashed lines are poor results that can be encountered from single-run FAST-TLE. With multiple runs and when fitting correctly, FAST-TLE's estimates are the same as GA-PM2C's.

Table 4.4
5-component Gaussian mixture with outliers

Component	μ, Σ	Number of samples
1	$(3 \ 4)^T, \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$	75
2	$(4.5 \ 6)^T, \begin{pmatrix} 0.36 & 0 \\ 0 & 0.3025 \end{pmatrix}$	100
3	$(11.5 \ 3)^T, \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$	100
4	$(14 \ 3)^T, \begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$	100
5	$(16 \ 7)^T, \begin{pmatrix} 0.3025 & 0 \\ 0 & 1.0 \end{pmatrix}$	150
Outliers		20

Fig. 4.5 also shows the cases of poor fitting that can be resulted from single-run FAST-TLE. As mentioned earlier, FAST-TLE should be run a certain number of times to select the best outcome from there. Running FAST-TLE only once and immediately accepting that result may not be a good idea if preceded by a poor initialization. Hence, multiple runs from different initial values are needed. This practice appears equivalent to using $|P|$ different parents in the initial population in GA-PM2C. However, the merit of our algorithm is more than just selecting the best result after various runs, as it will be demonstrated in the next experiments.

4.4.3 Mixture of Five Bivariate Gaussians with Outliers

In the previous dataset, the clusters are well-balanced and almost equally separated. In this section, we consider a more complex task. The dataset is described in Table 4.4. It contains 5 groups of data with different size, of which group 5 has the most number of observations, and also the largest covariance determinant. The centers of the groups are unequally separated: groups 1 & 2 are closer to each other than to the rest, so are groups 3 & 4, while group 5 is located far alone. Among the generated samples, 20 atypical points were added to create the outliers. The true contamination rate is, therefore, $\epsilon_0 = 3.7\%$.

In this experiment, we assigned ϵ to 3%, 4% and 5%, which are below, approximately equal and above the true contamination rate respectively. The number of parents in GA-PM2C (or the number of simultaneous runs of FAST-

Table 4.5
Success rates over 100 repetitions for dataset in Table 4.4

ϵ	Algorithm	$ P $				
		4	8	12	16	20
3%	FAST-TLE	1	1	2	6	7
	GA-PM2C	56	84	87	98	100
4%	FAST-TLE	9	13	19	36	37
	GA-PM2C	98	100	100	100	100
5%	FAST-TLE	60	94	96	100	100
	GA-PM2C	100	100	100	100	100

TLE) $|P|$ was also varied from 4 to 20. At each pair of $(\epsilon, |P|)$, 100 trials were executed. The number of times the algorithms correctly identified the 5 clusters is recorded in Table 4.5. It is shown that GA-PM2C outperforms FAST-TLE quite significantly in this case.

When $\epsilon = 3\%$, which is a little below the true contamination rate, FAST-TLE almost completely fails to distinguish the original groups of data. Even when $|P| = 20$, only 7 out of 100 trials are successful. On the other hand, GA-PM2C performs much better. With $|P| = 4$ only, its success rate is slightly higher than failure one. With $|P|$ equal 8 or greater, it has very high success rate, and with $|P|$ more than 16, it gives correct results all the times. When $\epsilon = 4\% \approx \epsilon_0$, FAST-TLE still has a higher failure rate than success one, whereas GA-PM2C has 100% success rate almost since $|P| = 4$. When ϵ is increased to 5%, higher than the true rate, FAST-TLE’s performance is then improved to be competent enough to GA-PM2C, which, at this stage, identifies the true classes perfectly with any values of $|P|$. The mixture components frequently estimated by the two algorithms with $\epsilon = 3\%$ and 4% are presented in Fig. 4.6. As shown, due to outlier effects, FAST-TLE mistakenly combines the two components 3 and 4 into one cluster. GA-PM2C, on the other hand, correctly distinguishes the outliers and the five distinct clusters.

The observation from this experiment clearly shows that FAST-TLE is more sensitive to the assigned contamination rate than GA-PM2C. Especially, with such an unbalanced and unequally distributed mixture of data, FAST-TLE may get trapped in local maxima due to the existence of outliers, even if just a few of them. It would be much safer for FAST-TLE to trim more data than the true percentage to get a higher chance of avoiding the “masking” and “swamping” effects of an outlier (although here, $\epsilon = 4\%$ is already greater than $\epsilon_0 = 3.7\%$). GA-PM2C, on the other hand, has an effective way to cancel out these effects

Fig. 4.6. GA-PM2C and FAST-TLE fits with ϵ at: (a) 0.03 and (b) 0.04. The contours are 95% ellipsoids of the Gaussians: thick lines represent true partitions; thin lines are results from GA-PM2C; dashed lines are incorrect fittings that are more often than not received from FAST-TLE. When FAST-TLE fits correctly, the estimates are the same as GA-PM2C's.

Fig. 4.7. An example of Recombination in GA-PM2C.

through Guided Mutation and Recombination. Within an individual chromosome, Guided Mutation helps to identify potential outliers. Later in the process, those outliers that could not be found by Guided Mutation may be picked out through Recombination between chromosomes. Such a phenomenon can be demonstrated by an example given in Fig. 4.7. The figure presents two segments in the chromosomes Parent 1 & Parent 2, which have just been guided-mutated and ready to mate to produce Offspring 1 & Offspring 2. The place where crossover occurs is shown by the dashed line. In each of the parents, there is a bold bit “1”, representing an outlier currently misassigned as model observation. Interestingly, the same bit is assigned correctly in the other parent. So, Parent 1 has a misassignment which is rightly determined in Parent 2, and vice versa. By crossover, the parents exchange their segments and produce Offspring 2 with all the correct assignments. Consequently, Offspring 2 is more likely to be selected ahead of its parents and Offspring 1, of course, to go to the next generation. Hence, in GA-PM2C, the interaction among individuals is very useful for selecting model observations and identifying outliers. With FAST-TLE, although we can have multiple runs to select the best outcome, the drawback in this design is that each run is totally an independent process and can not make use of any previous run to make any improvement.

4.4.4 Simulated Data in Higher Dimensions

Two datasets, $\mathbf{A}$ and $\mathbf{B}$, were created from four-component Gaussian mixtures in $\mathfrak{R}_5$ and $\mathfrak{R}_7$, respectively, for Monte-Carlo experiments. They are described in Table 4.6. For each dataset, 100 pairs of training sample and test sample were generated. The training samples were added with 50 data points produced from a uniform distribution within $(-10,10)$ in each dimension, but not the test samples.

Table 4.6
Datasets **A** and **B**

A			B		
Component	Train size	Test size	Component	Train size	Test size
$N_5(-10 \times \mathbf{1}, 16\mathbf{I})$	250	2500	$N_7(-8 \times \mathbf{1}, 16\mathbf{I})$	180	1800
$N_5(7 \times \mathbf{1}, \mathbf{I})$	85	850	$N_7(5 \times \mathbf{1}, \mathbf{I})$	70	700
$N_5(10 \times \mathbf{1}, \mathbf{I})$	75	750	$N_7(9 \times \mathbf{1}, 2.25\mathbf{I})$	50	500
$N_5([4 \times \mathbf{1}_3; -4 \times \mathbf{1}_2], 4\mathbf{I})$	150	1500	$N_7([-3 \times \mathbf{1}_3; \mathbf{0}_4], 4\mathbf{I})$	100	1000
$U_5(-10, 10)$	50		$U_7(-10, 10)$	50	

Table 4.7
Success rates over 100 Monte Carlo samples for datasets **A** and **B**

Algorithm	A			B		
	$\epsilon = 5\%$	$\epsilon = 10\%$	$\epsilon = 15\%$	$\epsilon = 5\%$	$\epsilon = 10\%$	$\epsilon = 15\%$
FAST-TLE	7	11	6	23	21	26
GA-PM2C	8	52	58	81	97	100
Classic GMM	6			18		

From simple calculation, we got $\epsilon_0 = 8.2\%$ for **A** and 11.1% for **B**. Setting parameter ϵ around these values at 5% , 10% and 15% would be appropriate.

In these experiments, we included the results of classical Gaussian mixture model (GMM) as to see whether robust models yield better classification performance. For each of the 100 pairs, the classical GMM, FAST-TLE and GA-PM2C models are used to fit the training set. A class label is then assigned to a component of the models if majority of the observations belonging to that component have that same label. Afterwards, using the estimated models, each observation in the test sample is classified with the class label of the component that has the highest likelihood of generating it. Error rate, which is the percentage of misclassifications, is calculated, and if it is greater than a threshold of 5×10^{-3} , the classification is considered as failed. Finally, the success rates over all 100 pairs of samples are used as measure of performance.

From the results in Table 4.7, it is clearly seen that classical model could not cope with the problem due to the existence of outliers in the training data. These atypical observations affect model estimation during training, and consequently lead to incorrect classification on the testing set. Classical GMM's success rates in both cases are low. The robust algorithms, FAST-TLE and GA-PM2C, show that they could produce better results. On both datasets, GA-PM2C outperforms FAST-TLE. It gives significant improvement, starting from $\epsilon = 10\%$ on **A**, or from the lowest of 5% on **B**.

Table 4.8
Cluster assignments with $k=3$ for Bushfire data

	GA-PM2C	Classic GMM
cluster 1	33 - 38	15 - 22, 32 - 38
cluster 2	7 - 11	7 - 14, 23, 24
cluster 3	1 - 6, 14 - 28	1 - 6, 25 - 31
trimmed	12, 13, 29 - 32	N.A.

4.4.5 Bushfire Data

This dataset was analyzed by Maronna and Zamar using their robust estimator for high-dimensional data in [134], [135]. It consists of 38 pixels of satellite measurements on 5 frequency bands. They considered the whole data as one class and, by various robust estimators of location and dispersion, pointed out that pixels 32-38 and pixels 7-11 were two groups of clear outliers, while 12, 29, 30 and 31 were somewhat suspect. They then suggested that the dataset could be classified into “burnt”, “unburnt” and “water”, and the suspect ones were at boundaries between classes. Hence, we can consider that the pixels are of three classes. Each group 32-38 and 7-11 forms one class, and the rest are of another class to some extent.

We used the proposed algorithm and the classical GMM with unconstrained covariance to cluster this dataset. With the former, a trimming level of 16% was used, which approximately equals the common recommended ratio $(1/\sqrt{38})$. Each method was run repeatedly for 20 times. Then, a similarity matrix A , whose element a_{ij} is the average number of times pixels i and j are assigned to same cluster, was constructed. We then based on this matrix to determine the average cluster assignments by the two algorithms. The results are shown in Table 4.8.

It can be seen that classical GMM does not recognize either 32-38 or 7-11 as separate cluster, but often mixes them with some samples from the remaining group. The proposed algorithm, on the other hand, clearly puts 33-38 and 7-11 into distinct clusters, and the rest into another, except pixels 12, 13, 29-32 have been selectively trimmed off. When viewing bushfire as a 3-class dataset, considering these samples as potential outliers helps clearly partition the rest into three groups as expected. This result is consistent with previous analysis in [134] and [135], where samples 12 and 29-31 have been suggested to locate on boundary areas between the classes. Pixel 13 being potential outlier is also

Table 4.9
Classification error rate (%) for Wisconsin data

Algorithm	$\epsilon = 5\%$	$\epsilon = 6\%$	$\epsilon = 7\%$
FAST-TLE	6.10 ± 0.70	5.84 ± 0.87	6.16 ± 1.01
GA-PM2C	5.61 ± 0.93	5.71 ± 0.38	5.78 ± 0.95
Classic GMM	10.50 ± 1.32		

agreeable, since it is possibly inferred from Fig. 1a in [134] and Table 5 in [135]. The only misclassified case is pixel 32, which has been said to be in same cluster with 33-38. Finally, it should also be noted that, from our observation, the proposed approach gave more consistent results throughout different trials than the classical Gaussian model.

4.4.6 Classification of Breast Cancer Data

Let us now examine the performance of GA-PM2C on a real-world problem, the popular Wisconsin diagnostic breast cancer data. This dataset can be found from the UCI Machine Learning repository [136]. It contains 569 instances of two classes, benign or malignant, with 30 attributes. When considering only 3 of the attributes, namely extreme area, extreme smoothness and mean texture, Fraley and Raftery [137] analyzed this dataset using three-group unconstrained-covariance Gaussian model. They pointed out that there were some “uncertain observations”. Hence, also with three-component unconstrained model, we carried out a classification procedure similar to section 4.4.4. The data were divided into 2 parts: 285 observations were randomly selected for training, and the rest were put in testing set. In this case, however, we do not have any clue about the percentage of noisy or atypical observations in this dataset. One way is to make use of a rule suggested in [105]: the fraction of data points that are placed arbitrarily in space is typically proportional to $\frac{1}{\sqrt{n}}$. Therefore, ϵ_0 in this circumstance was valued at $1/\sqrt{285} = 5.9\%$, and we set ϵ at different levels, specifically 5%, 6% and 7%, around this value.

Table 4.9 shows the average values with standard deviations of classification error rates over 100 repetitions of classical GMM, FAST-TLE and GA-PM2C. When compared with the classical model, both robust methods improve the classification quality significantly at all of the trimming levels considered. This indicates that some noisy observations do exist in the data. When they are taken care of in robust algorithms, the data models are estimated more precisely, and hence, yield better results. Among the three methods, GA-PM2C produces the

Fig. 4.8. Classification performance at different trimming rates: (a) Success rates for dataset **A**; (b) Success rates for dataset **B**; (c) Error rates for Wisconsin data.

best results.

In the above experiments, we have been cautious when deciding the amount of data to be trimmed. We have allowed this value to vary within a certain range around the value of ϵ_0 , which is either known a priori for the simulated datasets or determined by the guideline given by Dasgupta and Schulman [105] for the real dataset. One argument might be that, when suspecting outliers in a given dataset, it would be better to choose a generously large trimming rate. In our opinion, however, this must not always be the case. In clustering performance point of view, trimming off more data observations without improving precision means that recall is decreased. In term of model-based classification, trimming off too many training observations may bring inaccuracy to model estimation, and therefore, increase error rate. To examine such circumstance, we repeated the experiments on datasets **A**, **B** and Wisconsin for different values of ϵ from 3% to 50%. For dataset **A** in Fig. 4.8a and **B** in Fig. 4.8b, the success rates

start to decrease after around 30% to 40%. For the Wisconsin data in Fig. 4.8c, the classification error rates of the robust methods become even worse than that of classical model after 25% of trimming. Thus, it is encouraging that GAPM2C is able to offer satisfactory clustering quality at trimming levels which are relatively low, or not too high over the true contamination rate in data.

4.4.7 Running Time

In generally, GA approach is known to have high computational requirement. It is reasonable to say that GAPM2C is best suitable for small and medium size datasets. However, there are a few factors that help to speed up our algorithm and make it computationally bearable.

Firstly, in standard GA, the computational cost is most likely attributed to randomness. In our approach, process such as random mutation is replaced by guided mutation, which is directed toward a clear objective function. The random effect is better controlled here.

On the other hand, in GAPM2C, EM is only applied on the selected observations. During a cycle, the observations which are currently considered as suspected outliers do not take part in any computation. Then, it should be noted that full cycles of EM, i.e. from initialization to convergence, are not required. For each evolution, only a small number of cycles, specified by parameter C , are carried out. It has been shown that for our experiments, only a small number of cycles ($C = 5$) and a small number of chromosomes (e.g. $|P| = 4$) are needed to yield reasonably good results. What is more, experiments have shown that applying GAPM2C is even faster than running FAST-TLE a number of times equal to $|P|$ (and select the best result). Fig. 4.9 plots the running time durations (training time + testing time + output report) on datasets **A** and **B** at different trimming levels in the experiments in Section 4.4.4. It can be observed that GAPM2C requires less time than FAST-TLE in most of the cases. This is due to the fact, which has been discussed earlier, that the interaction among $|P|$ chromosomes in GAPM2C helps to improve the search process and speed up convergence. In contrast, there is no such interaction or no information exchanged among separate runs of FAST-TLE.

4.5 Conclusions

In this chapter, we implement a variant of classical M2C, named Partial M2C, in which “the weak assumption” is recommended over “the strong assumption”.

Fig. 4.9. Running time on datasets **A** and **B**.

A new general framework for the Partial M2C is proposed. The framework has a Model Sample Selection stage, where data observations are selected as either observations generated from a probabilistic model or outliers. We also propose GA-based Partial M2C algorithm, or GA-PM2C. The algorithm is capable of clustering data effectively in the presence of noise and outliers. We apply GA with a novel Guided Mutation operation to help filter out the effects of outliers. Empirical studies conducted have shown the effectiveness and efficiency of GA-PM2C. When compared with a closely related work FAST-TLE, GA-PM2C is much less sensitive to initializations, and gives more stable and consistent results.

GA with trimmed likelihood as fitness function has been used for Model Sample Selection in this study. However, any suitable methods other than GA, or fitness functions other than trimmed likelihood, can be applied for this stage. We believe that this is where a promising combination of discriminative approach and generative approach in data clustering can take place, because it is practically involving both mere objective function optimization and data modeling at the same time. Therefore, this can be a potential direction to explore further in the future.

Chapter 5

Multi-Viewpoint based Similarity Measure and Clustering Criterion Functions

5.1 Overview

Clustering is one of the most interesting and important topics in data mining. The aim of clustering is to find intrinsic structures in data, and organize them into meaningful subgroups for further study and analysis. There have been many clustering algorithms published every year. They can be proposed for very distinct research fields, and developed using totally different techniques and approaches. Nevertheless, according to a recent study [6], more than half a century after it was introduced, the simple algorithm k -means still remains as one of the top 10 data mining algorithms nowadays. It is the most frequently used partitional clustering algorithm in practice. Another recent scientific discussion [138] states that k -means is the favorite algorithm that practitioners in the related fields choose to use. Needless to mention, k -means has more than a few basic drawbacks, such as sensitiveness to initialization and to cluster size, and its performance can be worse than other state-of-the-art algorithms in many domains. In spite of that, its simplicity, understandability and scalability are the reasons for its tremendous popularity. An algorithm with adequate performance and usability in most of application scenarios could be preferable to one with better performance in some cases but limited usage due to high complexity. While offering reasonable results, k -means is fast and easy to combine with other methods in larger systems.

A common approach to the clustering problem is to treat it as an optimization

process. An optimal partition is found by optimizing a particular function of similarity (or distance) among data. Basically, there is an implicit assumption that the true intrinsic structure of data could be correctly described by the similarity formula defined and embedded in the clustering criterion function. Hence, effectiveness of clustering algorithms under this approach depends on the appropriateness of the similarity measure to the data at hand. For instance, the original k -means has sum-of-squared-error objective function that uses Euclidean distance. In a very sparse and high-dimensional domain like text documents, spherical k -means, which uses cosine similarity instead of Euclidean distance as the measure, is deemed to be more suitable [11, 139].

In [140], Banerjee et al. showed that Euclidean distance was indeed one particular form of a class of distance measures called Bregman divergences. They proposed Bregman hard-clustering algorithm, in which any kind of the Bregman divergences could be applied. Kullback-Leibler divergence was a special case of Bregman divergences that was said to give good clustering results on document datasets. Kullback-Leibler divergence is a good example of non-symmetric measure. Also on the topic of capturing dissimilarity in data, Pakalska et al. [141] found that the discriminative power of some distance measures could increase when their non-Euclidean and non-metric attributes were increased. They concluded that non-Euclidean and non-metric measures could be informative for statistical learning of data. In [142], Pelillo even argued that the symmetry and non-negativity assumption of similarity measures was actually a limitation of current state-of-the-art clustering approaches. Simultaneously, clustering still requires more robust dissimilarity or similarity measures; recent works such as [143] illustrate this need.

The work in this chapter is motivated by investigations from the above and similar research findings. It appears to us that the nature of similarity measure plays a very important role in the success or failure of a clustering method. Our first objective is to derive a novel method for measuring similarity between data objects in sparse and high-dimensional domain, particularly text documents. From the proposed similarity measure, we then formulate new clustering criterion functions and introduce their respective clustering algorithms, which are fast and scalable like k -means, but are also capable of providing high-quality and consistent performance.

The remaining of this chapter is organized as follows. In Section 5.2, we review related literature on similarity and clustering of documents. We then present our proposal for document similarity measure in Section 5.3. It is fol-

Table 5.1
Notations

Notation	Description
n	number of documents
m	number of terms
c	number of classes
k	number of clusters
d	document vector, $\ d\ = 1$
$S = \{d_1, \dots, d_n\}$	set of all the documents
S_r	set of documents in cluster r
$D = \sum_{d_i \in S} d_i$	composite vector of all the documents
$D_r = \sum_{d_i \in S_r} d_i$	composite vector of cluster r
$C = D/n$	centroid vector of all the documents
$C_r = D_r/n_r$	centroid vector of cluster r , $n_r = S_r $

lowed by two criterion functions for document clustering and their optimization algorithms in Section 5.4. Extensive experiments on real-world benchmark datasets are presented and discussed in Sections 5.5 and 5.6. Finally, conclusions and potential future work are given in Section 5.7.

5.2 Related Work

First of all, Table 5.1 summarizes the basic notations that will be used extensively throughout this chapter to represent documents and related concepts. Each document in a corpus corresponds to an m -dimensional vector d , where m is the total number of terms that the document corpus has. Document vectors are often subjected to some weighting schemes, such as the standard Term Frequency-Inverse Document Frequency (TF-IDF), and normalized to have unit length.

The principle definition of clustering is to arrange data objects into separate clusters such that the intra-cluster similarity as well as the inter-cluster dissimilarity is maximized. The problem formulation itself implies that some forms of measurement are needed to determine such similarity or dissimilarity. There are many state-of-the-art clustering approaches that do not employ any specific form of measurement, for instance, probabilistic model-based method [144], non-negative matrix factorization [23], information theoretic co-clustering [145] and so on. In this chapter, though, we primarily focus on methods that indeed do utilize a specific measure. In the literature, Euclidean distance is one of the

most popular measures:

$$Dist(d_i, d_j) = \|d_i - d_j\| \quad (5.1)$$

It is used in the traditional k -means algorithm. The objective of k -means is to minimize the Euclidean distance between objects of a cluster and that cluster's centroid:

$$\min \sum_{r=1}^k \sum_{d_i \in S_r} \|d_i - C_r\|^2 \quad (5.2)$$

However, for data in a sparse and high-dimensional space, such as that in document clustering, cosine similarity is more widely used. It is also a popular similarity score in text mining and information retrieval [146]. Particularly, similarity of two document vectors d_i and d_j , $Sim(d_i, d_j)$, is defined as the cosine of the angle between them. For unit vectors, this equals to their inner product:

$$Sim(d_i, d_j) = \cos(d_i, d_j) = d_i^t d_j \quad (5.3)$$

Cosine measure is used in a variant of k -means called spherical k -means [139]. While k -means aims to minimize Euclidean distance, spherical k -means intends to maximize the cosine similarity between documents in a cluster and that cluster's centroid:

$$\max \sum_{r=1}^k \sum_{d_i \in S_r} \frac{d_i^t C_r}{\|C_r\|} \quad (5.4)$$

The major difference between Euclidean distance and cosine similarity, and therefore between k -means and spherical k -means, is that the former focuses on vector magnitudes, while the latter emphasizes on vector directions. Besides direct application in spherical k -means, cosine of document vectors is also widely used in many other document clustering methods as a core similarity measurement. The min-max cut graph-based spectral method is an example [31]. In graph partitioning approach, document corpus is consider as a graph $G = (V, E)$, where each document is a vertex in V and each edge in E has a weight equal to the similarity between a pair of vertices. Min-max cut algorithm tries to minimize the criterion function:

$$\min \sum_{r=1}^k \frac{Sim(S_r, S \setminus S_r)}{Sim(S_r, S_r)} \quad (5.5)$$

where $Sim(S_q, S_r) = \sum_{1 \leq q, r \leq k} \sum_{d_i \in S_q, d_j \in S_r} Sim(d_i, d_j)$

and when the cosine as in Eq. (5.3) is used, minimizing the criterion in Eq. (5.5) is equivalent to:

$$\min \sum_{r=1}^k \frac{D_r^t D}{\|D_r\|^2} \quad (5.6)$$

There are many other graph partitioning methods with different cutting strategies and criterion functions, such as Average Weight [147] and Normalized Cut [30], all of which have been successfully applied for document clustering using cosine as the pairwise similarity score [33, 148]. In [149], an empirical study was conducted to compare a variety of criterion functions for document clustering.

Another popular graph-based clustering technique is implemented in a software package called CLUTO [32]. This method first models the documents with a nearest-neighbor graph, and then splits the graph into clusters using a min-cut algorithm. Besides cosine measure, the extended Jaccard coefficient can also be used in this method to represent similarity between nearest documents. Given non-unit document vectors u_i, u_j ($d_i = u_i/\|u_i\|$, $d_j = u_j/\|u_j\|$), their extended Jaccard coefficient is:

$$Sim_{eJacc}(u_i, u_j) = \frac{u_i^t u_j}{\|u_i\|^2 + \|u_j\|^2 - u_i^t u_j} \quad (5.7)$$

Compared with Euclidean distance and cosine similarity, the extended Jaccard coefficient takes into account both the magnitude and the direction of the document vectors. If the documents are instead represented by their corresponding unit vectors, this measure has the same effect as cosine similarity. In [102], Strehl et al. compared four measures: Euclidean, cosine, Pearson correlation and extended Jaccard, and concluded that cosine and extended Jaccard are the best ones on web documents.

In nearest-neighbor graph clustering methods, such as the CLUTO's graph method above, the concept of similarity is somewhat different from the previously discussed methods. Two documents may have a certain value of cosine similarity, but if neither of them is in the other one's neighborhood, they have no connection between them. In such a case, some context-based knowledge or relativeness property is already taken into account when considering similarity. Interestingly, through an algorithm called Locality Sensitive Hashing (LSH) [150, 151], the nearest neighbors of a data point can be estimated effectively without having to actually compute their similarities. The principle idea of LSH is to hash the data points, using multiple hashing functions, such that

the closer a pair of data points are to each other (in the sense of some similarity metric), the higher the probability of collision is. Since its introduction, LSH has been applied into clustering, mostly to improve the computational efficiency of the clustering algorithms due to the ability of identifying nearest neighbors quickly. It is particularly useful for clustering algorithms such as hierarchical clustering [152], where originally the full similarity matrix must have been explicitly calculated, and for clustering of very large web repository [153, 154].

Instead confining similarity measure to the neighborhood of a data point in full-dimensional space, a branch of clustering approaches, such as subspace clustering or projected clustering, take a further step to localize similarity to only extracted subspaces of the original dimensional space. Projected clustering algorithms such as ORCLUS [155] project data into several directions such that the subspaces can be specific to individual clusters and, hence, similarity among data points in a cluster is expressed the most in its subspace. In this case, the concept of similarity is localized and, because data partitioning and subspace formation are carried out simultaneously, measure of similarity is adaptively changed during the clustering process.

Recently, Ahmad and Dey [156] proposed a method to compute distance between two categorical values of an attribute based on their relationship with all other attributes. Subsequently, Ienco et al. [157] introduced a similar context-based distance learning method for categorical data. However, for a given attribute, they only selected a relevant subset of attributes from the whole attribute set to use as the context for calculating distance between its two values.

There are also phrase-based and concept-based similarity measures for documents. Lakkaraju et al. [158] employed a conceptual tree-similarity measure to identify similar documents. This method requires representing documents as concept trees with the help of a classifier. For clustering, Chim and Deng [159] proposed a phrase-based document similarity by combining suffix tree model and vector space model. They then used Hierarchical Agglomerative Clustering algorithm to perform the clustering task. However, a drawback of this approach is the high computational complexity due to the needs of building the suffix tree and calculating pairwise similarities explicitly before clustering. There are also measures designed specifically for capturing structural similarity among XML documents [160]. They are essentially different from the document-content measures that are discussed in this chapter.

In general, cosine similarity still remains as the most popular measure because of its simple interpretation and easy computation, though its effectiveness

is yet fairly limited. In the following sections, we propose a novel way to evaluate similarity between documents, and consequently formulate new criterion functions for document clustering.

5.3 Multi-Viewpoint based Similarity

5.3.1 Our Novel Similarity Measure

The cosine similarity in Eq. (5.3) can be expressed in the following form without changing its meaning:

$$Sim(d_i, d_j) = \cos(d_i - 0, d_j - 0) = (d_i - 0)^t (d_j - 0) \quad (5.8)$$

where 0 is vector 0 that represents the origin point. According to this formula, the measure takes 0 as one and only reference point. The similarity between two documents d_i and d_j is determined w.r.t. the angle between the two points when looking from the origin.

To construct a new concept of similarity, it is possible to use more than just one point of reference. We may have a more accurate assessment of how close or distant a pair of points are, if we look at them from many different viewpoints. From a third point d_h , the directions and distances to d_i and d_j are indicated respectively by the difference vectors $(d_i - d_h)$ and $(d_j - d_h)$. By standing at various reference points d_h to view d_i , d_j and working on their difference vectors, we define similarity between the two documents as:

$$Sim(d_i, d_j) = \frac{1}{n - n_r} \sum_{d_h \in S \setminus S_r} Sim(d_i - d_h, d_j - d_h) \quad (5.9)$$

As described by the above equation, similarity of two documents d_i and d_j - given that they are in the same cluster - is defined as the average of similarities measured relatively from the views of all other documents outside that cluster. What is interesting is that the similarity here is defined in a close relation to the clustering problem. A presumption of cluster memberships has been made prior to the measure. The two objects to be measured must be in the same cluster, while the points from where to establish this measurement must be outside of the cluster. We call this proposal the Multi-Viewpoint based Similarity, or MVS. From this point onwards, we will denote the proposed similarity measure between two document vectors d_i and d_j by $MVS(d_i, d_j | d_i, d_j \in S_r)$, or occasionally $MVS(d_i, d_j)$ for short.

The final form of MVS in Eq. (5.9) depends on particular formulation of the individual similarities within the sum. If the relative similarity is defined by dot-product of the difference vectors, we have:

$$\begin{aligned}
& \text{MVS}(d_i, d_j | d_i, d_j \in S_r) \\
&= \frac{1}{n - n_r} \sum_{d_h \in S \setminus S_r} (d_i - d_h)^t (d_j - d_h) \\
&= \frac{1}{n - n_r} \sum_{d_h} \cos(d_i - d_h, d_j - d_h) \|d_i - d_h\| \|d_j - d_h\| \quad (5.10)
\end{aligned}$$

The similarity between two points d_i and d_j inside cluster S_r , viewed from a point d_h outside this cluster, is equal to the product of the cosine of the angle between d_i and d_j looking from d_h and the Euclidean distances from d_h to these two points. This definition is based on the assumption that d_h is not in the same cluster with d_i and d_j . The smaller the distances $\|d_i - d_h\|$ and $\|d_j - d_h\|$ are, the higher the chance that d_h is in fact in the same cluster with d_i and d_j , and the similarity based on d_h should also be small to reflect this potential. Therefore, through these distances, Eq. (5.10) also provides a measure of inter-cluster dissimilarity, given that points d_i and d_j belong to cluster S_r , whereas d_h belongs to another cluster. The overall similarity between d_i and d_j is determined by taking average over all the viewpoints not belonging to cluster S_r . It is possible to argue that while most of these viewpoints are useful, there may be some of them giving misleading information just like it may happen with the origin point. However, given a large enough number of viewpoints and their variety, it is reasonable to assume that the majority of them will be useful. Hence, the effect of misleading viewpoints is constrained and reduced by the averaging step. It can be seen that this method offers more informative assessment of similarity than the single origin point based similarity measure.

5.3.2 Analysis and Practical Examples of MVS

In this section, we present analytical study to show that the proposed MVS could be a very effective similarity measure for data clustering. In order to demonstrate its advantages, MVS is compared with cosine similarity (CS) on how well they reflect the true group structure in document collections. Firstly, exploring Eq. (5.10), we have:

$$\text{MVS}(d_i, d_j | d_i, d_j \in S_r) = \frac{1}{n - n_r} \sum_{d_h \in S \setminus S_r} (d_i^t d_j - d_i^t d_h - d_j^t d_h + d_h^t d_h)$$

$$\begin{aligned}
\text{MVS}(d_i, d_j | d_i, d_j \in S_r) &= d_i^t d_j - \frac{1}{n-n_r} d_i^t \sum_{d_h} d_h - \frac{1}{n-n_r} d_j^t \sum_{d_h} d_h + 1, \quad \|d_h\|=1 \\
&= d_i^t d_j - \frac{1}{n-n_r} d_i^t D_{S \setminus S_r} - \frac{1}{n-n_r} d_j^t D_{S \setminus S_r} + 1 \\
&= d_i^t d_j - d_i^t C_{S \setminus S_r} - d_j^t C_{S \setminus S_r} + 1
\end{aligned} \tag{5.11}$$

where $D_{S \setminus S_r} = \sum_{d_h \in S \setminus S_r} d_h$ is the composite vector of all the documents outside cluster r , called the outer composite w.r.t. cluster r , and $C_{S \setminus S_r} = D_{S \setminus S_r} / (n - n_r)$ the outer centroid w.r.t. cluster r , $\forall r = 1, \dots, k$. From Eq. (5.11), when comparing two pairwise similarities $\text{MVS}(d_i, d_j)$ and $\text{MVS}(d_i, d_l)$, document d_j is more similar to document d_i than the other document d_l is, if and only if:

$$\begin{aligned}
&d_i^t d_j - d_j^t C_{S \setminus S_r} > d_i^t d_l - d_l^t C_{S \setminus S_r} \\
&\Leftrightarrow \cos(d_i, d_j) - \cos(d_j, C_{S \setminus S_r}) \|C_{S \setminus S_r}\| > \\
&\quad \cos(d_i, d_l) - \cos(d_l, C_{S \setminus S_r}) \|C_{S \setminus S_r}\|
\end{aligned} \tag{5.12}$$

From this condition, it is seen that even when d_l is considered “closer” to d_i in terms of CS, i.e. $\cos(d_i, d_j) \leq \cos(d_i, d_l)$, d_l can still possibly be regarded as less similar to d_i based on MVS if, on the contrary, it is “closer” enough to the outer centroid $C_{S \setminus S_r}$ than d_j is. This is intuitively reasonable, since the “closer” d_l is to $C_{S \setminus S_r}$, the greater the chance it actually belongs to another cluster rather than S_r and is, therefore, less similar to d_i . For this reason, MVS brings to the table an additional useful measure compared with CS.

To further justify the above proposal and analysis, we carried out a validity test for MVS and CS. The purpose of this test is to check how much a similarity measure coincides with the true class labels. It is based on one principle: if a similarity measure is appropriate for the clustering problem, for any of a document in the corpus, the documents that are closest to it based on this measure should be in the same cluster with it.

The validity test is designed as following. For each type of similarity measure, a similarity matrix $A = \{a_{ij}\}_{n \times n}$ is created. For CS, this is simple, as $a_{ij} = d_i^t d_j$. The procedure for building MVS matrix is described in Fig. 5.1. Firstly, the outer composite w.r.t. each class is determined. Then, for each row $\mathbf{a}_i$ of A , $i = 1, \dots, n$, if the pair of documents d_i and $d_j, j = 1, \dots, n$ are in the same class, a_{ij} is calculated as in line 10, Fig. 5.1. Otherwise, d_j is assumed to be in d_i ’s class, and a_{ij} is calculated as in line 12, Fig. 5.1. After matrix A is formed, the procedure in Fig. 5.2 is used to get its validity score. For each document d_i corresponding to row $\mathbf{a}_i$ of A , we select q_r documents closest to d_i . The value of

```

1: procedure BUILDMVSMATRIX( $A$ )
2:   for  $r \leftarrow 1 : c$  do
3:      $D_{S \setminus S_r} \leftarrow \sum_{d_i \notin S_r} d_i$ 
4:      $n_{S \setminus S_r} \leftarrow |S \setminus S_r|$ 
5:   end for
6:   for  $i \leftarrow 1 : n$  do
7:      $r \leftarrow \text{class of } d_i$ 
8:     for  $j \leftarrow 1 : n$  do
9:       if  $d_j \in S_r$  then
10:         $a_{ij} \leftarrow d_i^t d_j - d_i^t \frac{D_{S \setminus S_r}}{n_{S \setminus S_r}} - d_j^t \frac{D_{S \setminus S_r}}{n_{S \setminus S_r}} + 1$ 
11:      else
12:         $a_{ij} \leftarrow d_i^t d_j - d_i^t \frac{D_{S \setminus S_r} - d_j}{n_{S \setminus S_r} - 1} - d_j^t \frac{D_{S \setminus S_r} - d_i}{n_{S \setminus S_r} - 1} + 1$ 
13:      end if
14:    end for
15:  end for
16:  return  $A = \{a_{ij}\}_{n \times n}$ 
17: end procedure

```

Fig. 5.1. Procedure: Build MVS similarity matrix.

q_r is chosen relatively as *percentage* of the size of the class r that contains d_i , where *percentage* $\in (0, 1]$. Then, validity w.r.t. d_i is calculated by the fraction of these q_r documents having the same class label with d_i , as in line 12, Fig. 5.2. The final validity is determined by averaging over all the rows of A , as in line 14, Fig. 5.2. It is clear that validity score is bounded within 0 and 1. The higher validity score a similarity measure has, the more suitable it should be for the clustering task.

Two real-world document datasets are used as examples in this validity test. The first is *reuters7*, a subset of the famous collection, *Reuters-21578* Distribution 1.0, of Reuter’s newswire articles¹. *Reuters-21578* is one of the most widely used test collection for text categorization. In our validity test, we selected 2,500 documents from the largest 7 categories: “acq”, “crude”, “interest”, “earn”, “money-fx”, “ship” and “trade” to form *reuters7*. Some of the documents may appear in more than one category. The second dataset is *k1b*, a collection of 2,340 web pages from the Yahoo! subject hierarchy, including 6 topics: “health”, “entertainment”, “sport”, “politics”, “tech” and “business”. It was created from a past study in information retrieval called WebAce [100], and is now available with the CLUTO toolkit [32].

The two datasets were preprocessed by stop-word removal and stemming.

¹<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

Require: $0 < \textit{percentage} \leq 1$

```

1: procedure GETVALIDITY(validity, A, percentage)
2:   for  $r \leftarrow 1 : c$  do
3:      $q_r \leftarrow \lfloor \textit{percentage} \times n_r \rfloor$ 
4:     if  $q_r = 0$  then  $\triangleright$  percentage too small
5:        $q_r \leftarrow 1$ 
6:     end if
7:   end for
8:   for  $i \leftarrow 1 : n$  do
9:      $\{a_{iv[1]}, \dots, a_{iv[n]}\} \leftarrow \text{Sort } \{a_{i1}, \dots, a_{in}\}$ 
10:    s.t.  $a_{iv[1]} \geq a_{iv[2]} \geq \dots \geq a_{iv[n]}$ 
11:     $\{v[1], \dots, v[n]\} \leftarrow \text{permute } \{1, \dots, n\}$ 
12:     $r \leftarrow \text{class of } d_i$ 
13:     $\textit{validity}(d_i) \leftarrow \frac{|\{d_{v[1]}, \dots, d_{v[q_r]}\} \cap S_r|}{q_r}$ 
14:  end for
15:   $\textit{validity} \leftarrow \frac{\sum_{i=1}^n \textit{validity}(d_i)}{n}$ 
16: return validity
17: end procedure

```

Fig. 5.2. Procedure: Get validity score.

Moreover, we removed words that appear in less than two documents or more than 99.5% of the total number of documents. Finally, the documents were weighted by TF-IDF and normalized to unit vectors. The full characteristics of *reuters7* and *k1b* are presented in Fig. 5.3.

Fig. 5.4 shows the validity scores of CS and MVS on the two datasets relative to the parameter *percentage*. The value of *percentage* is set at 0.001, 0.01, 0.05, 0.1, 0.2, ..., 1.0. According to Fig. 5.4, MVS is clearly better than CS for both datasets in this validity test. For example, with *k1b* dataset at *percentage* = 1.0, MVS' validity score is 0.80, while that of CS is only 0.67. This indicates that, on average, when we pick up any document and consider its neighborhood of size equal to its true class size, only 67% of that document's neighbors based on CS actually belong to its class. If based on MVS, the number of valid neighbors increases to 80%. This validity test has shown the potential advantage of the new multi-viewpoint based similarity measure compared to the cosine measure. More similar results of the validity test on datasets *tr31*, *reviews*, *la12*, *sports*, *tr12* and *tr23* are illustrated in Figures 5.5, 5.6 and 5.7.

Fig. 5.3. Characteristics of *reuters7* and *k1b* datasets.

Fig. 5.4. Validity test on *reuters10* and *k1b*.

5.4 Multi-Viewpoint based Clustering

5.4.1 Two Clustering Criterion Functions I_R and I_V

Having defined our similarity measure, we now formulate our clustering criterion functions. The first function, called I_R , is the cluster size-weighted sum of average pairwise similarities of documents in the same cluster. Firstly, let us

Fig. 5.5. Validity test on *tr31* and *reviews*.

Fig. 5.6. Validity test on *la12* and *sports*.

express this sum in a general form by function F :

$$F = \sum_{r=1}^k n_r \left[\frac{1}{n_r^2} \sum_{d_i, d_j \in S_r} Sim(d_i, d_j) \right] \quad (5.13)$$

Fig. 5.7. Validity test on $tr12$ and $tr23$.

We would like to transform this objective function into some suitable form such that it could facilitate the optimization procedure to be performed in a simple, fast and effective way. According to Eq. (5.10):

$$\begin{aligned}
 \sum_{d_i, d_j \in S_r} Sim(d_i, d_j) &= \sum_{d_i, d_j \in S_r} \frac{1}{n - n_r} \sum_{d_h \in S \setminus S_r} (d_i - d_h)^t (d_j - d_h) \\
 &= \frac{1}{n - n_r} \sum_{d_i, d_j} \sum_{d_h} (d_i^t d_j - d_i^t d_h - d_j^t d_h + d_h^t d_h)
 \end{aligned}$$

Since

$$\begin{aligned}
 \sum_{d_i \in S_r} d_i &= \sum_{d_j \in S_r} d_j = D_r, \\
 \sum_{d_h \in S \setminus S_r} d_h &= D - D_r \text{ and } \|d_h\| = 1,
 \end{aligned}$$

we have

$$\begin{aligned}
 \sum_{d_i, d_j \in S_r} Sim(d_i, d_j) &= \sum_{d_i, d_j \in S_r} d_i^t d_j - \frac{2n_r}{n - n_r} \sum_{d_i \in S_r} d_i^t \sum_{d_h \in S \setminus S_r} d_h + n_r^2 \\
 &= D_r^t D_r - \frac{2n_r}{n - n_r} D_r^t (D - D_r) + n_r^2 \\
 &= \frac{n + n_r}{n - n_r} \|D_r\|^2 - \frac{2n_r}{n - n_r} D_r^t D + n_r^2
 \end{aligned}$$

Substituting into Eq. (5.13) to get:

$$F = \sum_{r=1}^k \frac{1}{n_r} \left[\frac{n+n_r}{n-n_r} \|D_r\|^2 - \left(\frac{n+n_r}{n-n_r} - 1 \right) D_r^t D \right] + n$$

Because n is constant, maximizing F is equivalent to maximizing $\bar{F}$:

$$\bar{F} = \sum_{r=1}^k \frac{1}{n_r} \left[\frac{n+n_r}{n-n_r} \|D_r\|^2 - \left(\frac{n+n_r}{n-n_r} - 1 \right) D_r^t D \right] \quad (5.14)$$

If comparing $\bar{F}$ with the min-max cut in Eq. (5.5), both functions contain the two terms $\|D_r\|^2$ (an intra-cluster similarity measure) and $D_r^t D$ (an inter-cluster similarity measure). Nonetheless, while the objective of min-max cut is to minimize the inverse ratio between these two terms, our aim here is to maximize their weighted difference. In $\bar{F}$, this difference term is determined for each cluster. They are weighted by the inverse of the cluster's size, before summed up over all the clusters. One problem is that this formulation is expected to be quite sensitive to cluster size. From the formulation of COSA [161] - a widely known subspace clustering algorithm - we have learned that it is desirable to have a set of weight factors $\lambda = \{\lambda_r\}_1^k$ to regulate the distribution of these cluster sizes in clustering solutions. Hence, we integrate λ into the expression of $\bar{F}$ to have it become:

$$\bar{F}_\lambda = \sum_{r=1}^k \frac{\lambda_r}{n_r} \left[\frac{n+n_r}{n-n_r} \|D_r\|^2 - \left(\frac{n+n_r}{n-n_r} - 1 \right) D_r^t D \right] \quad (5.15)$$

In common practice, $\{\lambda_r\}_1^k$ are often taken to be simple functions of the respective cluster sizes $\{n_r\}_1^k$ [162]. Let us use a parameter α called the regulating factor, which has some constant value ($\alpha \in [0, 1]$), and let $\lambda_r = n_r^\alpha$ in Eq. (5.15), the final form of our criterion function I_R is:

$$I_R = \sum_{r=1}^k \frac{1}{n_r^{1-\alpha}} \left[\frac{n+n_r}{n-n_r} \|D_r\|^2 - \left(\frac{n+n_r}{n-n_r} - 1 \right) D_r^t D \right] \quad (5.16)$$

In the empirical study of Section 5.5.3, it appears that I_R 's performance dependency on the value of α is not very critical. The criterion function yields relatively good clustering results for $\alpha \in (0, 1)$.

In the formulation of I_R , a cluster quality is measured by the average pairwise similarity between documents within that cluster. However, such an approach

can lead to sensitiveness to the size and tightness of the clusters. With CS, for example, pairwise similarity of documents in a sparse cluster is usually smaller than those in a dense cluster. Though not as clear as with CS, it is still possible that the same effect may hinder MVS-based clustering if using pairwise similarity. To prevent this, an alternative approach is to consider similarity between each document vector and its cluster's centroid instead. This is expressed in objective function G :

$$\begin{aligned}
G &= \sum_{r=1}^k \sum_{d_i \in S_r} \frac{1}{n-n_r} \sum_{d_h \in S \setminus S_r} \text{Sim} \left(d_i - d_h, \frac{C_r}{\|C_r\|} - d_h \right) \\
G &= \sum_{r=1}^k \frac{1}{n-n_r} \sum_{d_i \in S_r} \sum_{d_h \in S \setminus S_r} (d_i - d_h)^t \left(\frac{C_r}{\|C_r\|} - d_h \right) \quad (5.17)
\end{aligned}$$

Similar to the formulation of I_R , we would like to express this objective in a simple form that we could optimize more easily. Exploring the vector dot product, we get:

$$\begin{aligned}
& \sum_{d_i \in S_r} \sum_{d_h \in S \setminus S_r} (d_i - d_h)^t \left(\frac{C_r}{\|C_r\|} - d_h \right) \\
&= \sum_{d_i} \sum_{d_h} \left(d_i^t \frac{C_r}{\|C_r\|} - d_i^t d_h - d_h^t \frac{C_r}{\|C_r\|} + 1 \right) \\
&= (n-n_r) D_r^t \frac{D_r}{\|D_r\|} - D_r^t (D - D_r) - n_r (D - D_r)^t \frac{D_r}{\|D_r\|} \\
&\quad + n_r (n - n_r) \quad , \text{ since } \frac{C_r}{\|C_r\|} = \frac{D_r}{\|D_r\|} \\
&= (n + \|D_r\|) \|D_r\| - (n_r + \|D_r\|) \frac{D_r^t D}{\|D_r\|} + n_r (n - n_r)
\end{aligned}$$

Substituting the above into Eq. (5.17) to have:

$$G = \sum_{r=1}^k \left[\frac{n + \|D_r\|}{n - n_r} \|D_r\| - \left(\frac{n + \|D_r\|}{n - n_r} - 1 \right) \frac{D_r^t D}{\|D_r\|} \right] + n$$

Again, we could eliminate n because it is a constant. Maximizing G is equivalent to maximizing I_V below:

$$I_V = \sum_{r=1}^k \left[\frac{n + \|D_r\|}{n - n_r} \|D_r\| - \left(\frac{n + \|D_r\|}{n - n_r} - 1 \right) \frac{D_r^t D}{\|D_r\|} \right] \quad (5.18)$$

I_V calculates the weighted difference between the two terms: $\|D_r\|$ and $D_r^t D / \|D_r\|$,

which again represent an intra-cluster similarity measure and an inter-cluster similarity measure, respectively. The first term is actually equivalent to an element of the sum in spherical k -means objective function in Eq. (5.4); the second one is similar to an element of the sum in min-max cut criterion in Eq. (5.6), but with $\|D_r\|$ as scaling factor instead of $\|D_r\|^2$. We have presented our clustering criterion functions I_R and I_V in the simple forms. Next, we show how to perform clustering by using a greedy algorithm to optimize these functions.

5.4.2 Optimization Algorithm and Complexity

We denote our clustering framework by MVSC, meaning Clustering with Multi-Viewpoint based Similarity. Subsequently, we have MVSC- I_R and MVSC- I_V , which are MVSC with criterion function I_R and I_V respectively. The main goal is to perform document clustering by optimizing I_R in Eq. (5.16) and I_V in Eq. (5.18). For this purpose, the incremental k -way algorithm [149, 163] - a sequential version of k -means - is employed. Considering that the expression of I_V in Eq. (5.18) depends only on n_r and D_r , $r = 1, \dots, k$, I_V can be written in a general form:

$$I_V = \sum_{r=1}^k I_r(n_r, D_r) \quad (5.19)$$

where $I_r(n_r, D_r)$ corresponds to the objective value of cluster r . The same is applied to I_R . With this general form, the incremental optimization algorithm, which has two major steps Initialization and Refinement, is described in Fig. 5.8. At Initialization, k arbitrary documents are selected to be the seeds from which initial partitions are formed. Refinement is a procedure that consists of a number of iterations. During each iteration, the n documents are visited one by one in a totally random order. Each document is checked if its move to another cluster results in improvement of the objective function. If yes, the document is moved to the cluster that leads to the highest improvement. If no clusters are better than the current cluster, the document is not moved. The clustering process terminates when an iteration completes without any documents being moved to new clusters. Unlike the traditional k -means, this algorithm is a step-wise optimal procedure. While k -means only updates after all n documents have been re-assigned, the incremental clustering algorithm updates immediately whenever each document is moved to new cluster. Since every move when happens increases the objective function value, convergence to a local optimum is guaranteed.

During the optimization procedure, in each iteration, the main sources of

```

1: procedure INITIALIZATION
2:   Select  $k$  seeds  $s_1, \dots, s_k$  randomly
3:    $cluster[d_i] \leftarrow p = \arg \max_r \{s_r^t d_i\}, \forall i = 1, \dots, n$ 
4:    $D_r \leftarrow \sum_{d_i \in S_r} d_i, n_r \leftarrow |S_r|, \forall r = 1, \dots, k$ 
5: end procedure
6: procedure REFINEMENT
7:   repeat
8:      $\{v[1 : n]\} \leftarrow$  random permutation of  $\{1, \dots, n\}$ 
9:     for  $j \leftarrow 1 : n$  do
10:       $i \leftarrow v[j]$ 
11:       $p \leftarrow cluster[d_i]$ 
12:       $\Delta I_p \leftarrow I(n_p - 1, D_p - d_i) - I(n_p, D_p)$ 
13:       $q \leftarrow \arg \max_{r, r \neq p} \{I(n_r + 1, D_r + d_i) - I(n_r, D_r)\}$ 
14:       $\Delta I_q \leftarrow I(n_q + 1, D_q + d_i) - I(n_q, D_q)$ 
15:      if  $\Delta I_p + \Delta I_q > 0$  then
16:        Move  $d_i$  to cluster  $q$ :  $cluster[d_i] \leftarrow q$ 
17:        Update  $D_p, n_p, D_q, n_q$ 
18:      end if
19:    end for
20:  until No move for all  $n$  documents
21: end procedure

```

Fig. 5.8. Algorithm: Incremental clustering.

computational cost are:

- Searching for optimum clusters to move individual documents to: $O(nz \cdot k)$.
- Updating composite vectors as a result of such moves: $O(m \cdot k)$.

where nz is the total number of non-zero entries in all document vectors. Our clustering approach is partitional and incremental; therefore, computing similarity matrix is absolutely not needed. If τ denotes the number of iterations the algorithm takes, since nz is often several tens times larger than m for document domain, the computational complexity required for clustering with I_R and I_V is $O(nz \cdot k \cdot \tau)$.

5.5 Performance Evaluation of MVSC

To verify the advantages of our proposed methods, we evaluate their performance in experiments on document data. The objective of this section is to compare MVSC- I_R and MVSC- I_V with the existing algorithms that also use specific similarity measures and criterion functions for document clustering. The similarity

measures to be compared includes Euclidean distance, cosine similarity and extended Jaccard coefficient.

5.5.1 Experimental Setup and Evaluation

In order to demonstrate how well MVSCs can perform, we compare them with five other clustering methods on twenty document datasets, including *fbis*, *hitech*, *k1a*, *k1b*, *la1*, *la2*, *re0*, *re1*, *tr31*, *reviews*, *wap*, *classic*, *la12*, *new3*, *sports*, *tr11*, *tr12*, *tr23*, *tr45* and *reuters7* (refer to Section 2.5 for the details of these datasets). In short descriptions, the seven clustering algorithms are:

- MVSC- I_R : MVSC using criterion function I_R
- MVSC- I_V : MVSC using criterion function I_V
- k -means: standard k -means with Euclidean distance
- Spkmeans: spherical k -means with CS
- graphCS: CLUTO’s graph method with CS
- graphEJ: CLUTO’s graph with extended Jaccard
- MMC: Spectral Min-Max Cut algorithm [31]

Our MVSC- I_R and MVSC- I_V programs are implemented in Java. The regulating factor α in I_R is always set at 0.3 during the experiments. We observed that this is one of the most appropriate values. A study on MVSC- I_R ’s performance relative to different α values is presented in a later section. The other algorithms are provided by the C library interface which is available freely with the CLUTO toolkit [32]. For each dataset, cluster number is predefined equal to the number of true class, i.e. $k = c$.

None of the above algorithms are guaranteed to find global optimum, and all of them are initialization-dependent. Hence, for each method, we performed clustering a few times with randomly initialized values, and chose the best trial in terms of the corresponding objective function value. In all the experiments, each test run consisted of 10 trials. Moreover, the result reported here on each dataset by a particular clustering method is the average of 10 test runs.

After a test run, clustering solution is evaluated by comparing the documents’ assigned labels with their true labels provided by the corpus. Three types of external evaluation metric are used to assess clustering performance. They are *FScore*, *NMI* and *Accuracy* (refer to Section 2.6 for the information about these measures).

Fig. 5.9. Clustering results in *Accuracy*. Left-to-right in legend corresponds to left-to-right in the plot.

5.5.2 Experimental Results

Fig. 5.9 shows the *Accuracy* of the seven clustering algorithms on the twenty text collections. Presented in a different way, clustering results based on *FScore* and *NMI* are reported in Table 5.2 and Table 5.3 respectively. For each dataset in a row, the value in bold and underlined is the best result, while the value in bold only is the second to best.

It can be observed that MVSC- I_R and MVSC- I_V perform consistently well. In Fig. 5.9, 19 out of 20 datasets, except *reviews*, either both or one of MVSC approaches are in the top two algorithms. The next consistent performer is Sp k means. The other algorithms might work well on certain dataset. For example, graphEJ yields outstanding result on *classic*; graphCS and MMC are good on *reviews*. But they do not fare very well on the rest of the collections.

To have a statistical justification of the clustering performance comparisons,

Table 5.2
Clustering results in $FScore$

Data	MVSC- I_R	MVSC- I_V	k -means	Sp k means	graphCS	graphEJ	MMC
fbis	<u>.645</u>	.613	.578	.584	.482	.503	.506
hitech	.512	<u>.528</u>	.467	.494	.492	.497	.468
k1a	<u>.620</u>	.592	.502	.545	.492	.517	.524
k1b	<u>.873</u>	.775	.825	.729	.740	.743	.707
la1	.719	<u>.723</u>	.565	.719	.689	.679	.693
la2	.721	<u>.749</u>	.538	.703	.689	.633	.698
re0	.460	.458	.421	.421	<u>.468</u>	.454	.390
re1	<u>.514</u>	.492	.456	.499	.487	.457	.443
tr31	.728	<u>.780</u>	.585	.679	.689	.698	.607
reviews	.734	.748	.644	.730	<u>.759</u>	.690	.749
wap	<u>.610</u>	.571	.516	.545	.513	.497	.513
classic	.658	.734	.713	.687	.708	<u>.983</u>	.657
la12	.719	<u>.735</u>	.559	.722	.706	.671	.693
new3	.548	.547	.500	<u>.558</u>	.510	.496	.482
sports	.803	<u>.804</u>	.499	.702	.689	.696	.650
tr11	<u>.749</u>	.728	.705	.719	.665	.658	.695
tr12	.743	<u>.758</u>	.699	.715	.642	.722	.700
tr23	<u>.560</u>	.553	.486	.523	.522	.531	.485
tr45	.787	.788	.692	<u>.799</u>	.778	.798	.720
reuters7	.774	<u>.775</u>	.658	.718	.651	.670	.687

we also carried out statistical significance tests. Each of MVSC- I_R and MVSC- I_V was paired up with one of the remaining algorithms for a paired t -test [164]. Given two paired sets $\mathbf{X}$ and $\mathbf{Y}$ of N measured values, the null hypothesis of the test is that the differences between $\mathbf{X}$ and $\mathbf{Y}$ come from a population with mean 0. The alternative hypothesis is that the paired sets differ from each other in a significant way. In our experiment, these tests were done based on the evaluation values obtained on the twenty datasets. The typical 5% significance level was used. For example, considering the pair (MVSC- I_R , k -means), from Table 5.2, it is seen that MVSC- I_R dominates k -means w.r.t. $FScore$. If the paired t -test returns a p -value smaller than 0.05, we reject the null hypothesis and say that the dominance is significant. Otherwise, the null hypothesis is true and the comparison is considered insignificant.

Table 5.3
Clustering results in *NMI*

Data	MVSC- I_R	MVSC- I_V	k -means	Sp k means	graphCS	graphEJ	MMC
fbis	.606	.595	.584	.593	.527	.524	.556
hitech	.323	.329	.270	.298	.279	.292	.283
k1a	.612	.594	.563	.596	.537	.571	.588
k1b	.739	.652	.629	.649	.635	.650	.645
la1	.569	.571	.397	.565	.490	.485	.553
la2	.568	.590	.381	.563	.496	.478	.566
re0	.399	.402	.388	.399	.367	.342	.414
re1	.591	.583	.532	.593	.581	.566	.515
tr31	.613	.658	.488	.594	.577	.580	.548
reviews	.584	.603	.460	.607	.570	.528	.639
wap	.611	.585	.568	.596	.557	.555	.575
classic	.574	.644	.579	.577	.558	.928	.543
la12	.574	.584	.378	.568	.496	.482	.558
new3	.621	.622	.578	.626	.580	.580	.577
sports	.669	.701	.445	.633	.578	.581	.591
tr11	.712	.674	.660	.671	.634	.594	.666
tr12	.686	.686	.647	.654	.578	.626	.640
tr23	.432	.434	.363	.413	.344	.380	.369
tr45	.734	.733	.640	.748	.726	.713	.667
reuters7	.633	.632	.512	.612	.503	.520	.591

The outcomes of the paired t -tests are presented in Table 5.4. As the paired t -tests show, the advantage of MVSC- I_R and MVSC- I_V over the other methods is statistically significant. A special case is the graphEJ algorithm. On the one hand, MVSC- I_R is not significantly better than graphEJ if based on $FScore$ or NMI . On the other hand, when MVSC- I_R and MVSC- I_V are tested obviously better than graphEJ, the p -values can still be considered relatively large, although they are smaller than 0.05. The reason is that, as observed before, graphEJ’s results on *classic* dataset are very different from those of the other algorithms. While interesting, these values can be considered as outliers, and including them in the statistical tests would affect the outcomes greatly. Hence, we also report in Table 5.4 the tests where *classic* was excluded and only results on the other 19 datasets were used. Under this circumstance, both MVSC- I_R

Table 5.4

Statistical significance of comparisons based on paired t -tests with 5% significance level

		k -means	Spkmeans	graphCS	graphEJ*	MMC
$FScore$	MVSC- I_R	$\gg$	$\gg$	$\gg$	$> (\gg)$	$\gg$
		1.77E-5	1.60E-3	4.61E-4	.056 (7.68E-6)	3.27E-6
	MVSC- I_V	$\gg$	$\gg$	$\gg$	$\gg (\gg)$	$\gg$
		7.52E-5	1.42E-4	3.27E-5	.022 (1.50E-6)	2.16E-7
NMI	MVSC- I_R	$\gg$	$\gg$	$\gg$	$> (\gg)$	$\gg$
		7.42E-6	.013	2.39E-7	.060 (1.65E-8)	8.72E-5
	MVSC- I_V	$\gg$	$\gg$	$\gg$	$\gg (\gg)$	$\gg$
		4.27E-5	.013	4.07E-7	.029 (4.36E-7)	2.52E-4
$Accuracy$	MVSC- I_R	$\gg$	$\gg$	$\gg$	$\gg (\gg)$	$\gg$
		1.45E-6	1.50E-4	1.33E-4	.028 (3.29E-5)	8.33E-7
	MVSC- I_V	$\gg$	$\gg$	$\gg$	$\gg (\gg)$	$\gg$
		1.74E-5	1.82E-4	4.19E-5	.014 (8.61E-6)	9.80E-7

“ $\gg$ ” (or “ $\ll$ ”) indicates the algorithm in the row performs significantly better (or worse) than the one in the column; “ $>$ ” (or “ $<$ ”) indicates an insignificant comparison. The values right below the symbols are p -values of the t -tests.

* Column of graphEJ: entries in parentheses are statistics when *classic* dataset is not included.

and MVSC- I_V outperform graphEJ significantly with good p -values.

5.5.3 Effect of α on MVSC- I_R ’s performance

It has been known that criterion function based partitional clustering methods can be sensitive to cluster size and balance. In the formulation of I_R in Eq. (5.16), there exists parameter α which is called the regulating factor, $\alpha \in [0, 1]$. To examine how the determination of α could affect MVSC- I_R ’s performance, we evaluated MVSC- I_R with different values of α from 0 to 1, with 0.1 incremental interval. The assessment was done based on the clustering results in NMI , $FScore$ and $Accuracy$, each averaged over all the twenty given datasets. Since the evaluation metrics for different datasets could be very different from each other, simply taking the average over all the datasets would not be very meaningful. Hence, we employed the method used in [149] to transform the metrics into *relative metrics* before averaging. On a particular document collection S , the

Fig. 5.10. MVSC- I_R 's performance with respect to α .

relative FScore measure of MVSC- I_R with $\alpha = \alpha_i$ is determined as following:

$$\text{relative_FScore}(I_R; S, \alpha_i) = \frac{\max_{\alpha_j} \{FScore(I_R; S, \alpha_j)\}}{FScore(I_R; S, \alpha_i)}$$

where $\alpha_i, \alpha_j \in \{0.0, 0.1, \dots, 1.0\}$, $FScore(I_R; S, \alpha_i)$ is the *FScore* result on dataset S obtained by MVSC- I_R with $\alpha = \alpha_i$. The same transformation was applied to *NMI* and *Accuracy* to yield *relative_NMI* and *relative_Accuracy* respectively. MVSC- I_R performs the best with an α_i if its relative measure has a value of 1. Otherwise its relative measure is greater than 1; the larger this value is, the worse MVSC- I_R with α_i performs in comparison with other settings of α . Finally, the average relative measures were calculated over all the datasets to present the overall performance.

Figure 5.10 shows the plot of average *relative FScore*, *NMI* and *Accuracy* w.r.t. different values of α . In a broad view, MVSC- I_R performs the worst at the extreme values of α (0 and 1), and tends to get better when α is set at some soft values in between 0 and 1. Based on our experimental study, MVSC- I_R always produces results within 5% of the best case, regarding any types of evaluation metric, with α from 0.2 to 0.8.

5.6 MVSC as Refinement for k -means

5.6.1 Introduction

From the analysis of Eq. (5.12) in Section 5.3.2, MVS provides an additional criterion for measuring the similarity among documents compared with CS. Alternatively, MVS can be considered as a refinement for CS, and consequently MVSC algorithms as refinements for spherical k -means, which uses CS. To further investigate the appropriateness and effectiveness of MVS and its clustering algorithms, we carried out another set of experiments in which solutions obtained by $\text{Sp}k\text{means}$ were further optimized by $\text{MVSC-}I_R$ and $\text{MVSC-}I_V$. The rationale for doing so is that if the final solutions by $\text{MVSC-}I_R$ and $\text{MVSC-}I_V$ are better than the intermediate ones obtained by $\text{Sp}k\text{means}$, MVS is indeed good for the clustering problem. These experiments would reveal more clearly if MVS actually improves the clustering performance compared with CS.

In the previous section, MVSC algorithms have been compared against the existing algorithms that are closely related to them, i.e. ones that also employ similarity measures and criterion functions. In this section, we make use of the extended experiments to further compare the MVSC with a different type of clustering approach, the NMF methods [23], which do not use any form of explicitly defined similarity measure for documents.

5.6.2 Experimental Setup

The following clustering methods:

- $\text{Sp}k\text{means}$: spherical k -means
- $\text{rMVSC-}I_R$: refinement of $\text{Sp}k\text{means}$ by $\text{MVSC-}I_R$
- $\text{rMVSC-}I_V$: refinement of $\text{Sp}k\text{means}$ by $\text{MVSC-}I_V$
- $\text{MVSC-}I_R$: normal MVSC using criterion I_R
- $\text{MVSC-}I_V$: normal MVSC using criterion I_V

and two new document clustering approaches that do not use any particular form of similarity measure:

- NMF: Non-negative Matrix Factorization method
- NMF-NCW: Normalized Cut Weighted NMF

were involved in the performance comparison. When used as a refinement for *Spkmeans*, the algorithms *rMVSC- I_R* and *rMVSC- I_V* worked directly on the output solution of *Spkmeans*. The cluster assignment produced by *Spkmeans* was used as initialization for both *rMVSC- I_R* and *rMVSC- I_V* . We also investigated the performance of the original *MVSC- I_R* and *MVSC- I_V* further on the new datasets. Besides, it would be interesting to see how they and their *Spkmeans*-initialized versions fare against one another. What is more, two well-known document clustering approaches based on non-negative matrix factorization, NMF and NMF-NCW [23], are also included in the comparison. Our algorithms and the NMFs are different in nature: the formers utilize a document similarity measure, which is the proposed MVS, whereas the latters do not define any explicit measure.

For variety and thoroughness, in this empirical study, we used two more document corpora: *TDT2* and *Reuters-21578* (refer to Section 2.5, Table 2.3 for the details of these datasets). During the experiments, each of the two corpora were used to create 6 different test cases, each of which corresponded to a distinct number of topics used ($c = 5, \dots, 10$). For each test case, c topics were randomly selected from the corpus and their documents were mixed together to form a test set. This selection was repeated 50 times so that each test case had 50 different test sets. The average performance of the clustering algorithms with $k = c$ were calculated over these 50 test sets. This experimental set-up is inspired by the similar experiments conducted in the NMF paper [23]. Furthermore, similar to previous experimental setup in Section 5.5.1, each algorithm (including NMF and NMF-NCW) actually considered 10 trials on any test set before using the solution of the best obtainable objective function value as its final output.

5.6.3 Experimental Results

The clustering results on *TDT2* and *Reuters-21578* are shown in Table 5.5 and 5.6 respectively. For each test case in a column, the value in bold and underlined is the best among the results returned by the algorithms, while the value in bold only is the second to best. From the tables, several observations can be made. Firstly, *MVSC- I_R* and *MVSC- I_V* continue to show they are good clustering algorithms by outperforming other methods frequently. They are always the best in every test case of *TDT2*. Compared with NMF-NCW, they are better in almost all the cases, except only the case of *Reuters-21578*, $k = 5$, where NMF-NCW is the best based on *Accuracy*.

The second observation, which is also the main objective of this empirical

Table 5.5
Clustering results on *TDT2*

Algorithms	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
<i>NMI</i>						
Spkmeans	.690	.704	.700	.677	.681	.656
rMVSC- I_R	.753	.777	.766	.749	.738	.699
rMVSC- I_V	.740	.764	.742	.729	.718	.676
MVSC- I_R	.749	.790	.797	.760	.764	.722
MVSC- I_V	.775	.785	.779	.745	.755	.714
NMF	.621	.630	.607	.581	.593	.555
NMF-NCW	.713	.746	.723	.707	.702	.659
<i>Accuracy</i>						
Spkmeans	.708	.689	.668	.620	.605	.578
rMVSC- I_R	.855	.846	.822	.802	.760	.722
rMVSC- I_V	.839	.837	.801	.785	.736	.701
MVSC- I_R	.884	.867	.875	.840	.832	.780
MVSC- I_V	.886	.871	.870	.825	.818	.777
NMF	.697	.686	.642	.604	.578	.555
NMF-NCW	.788	.821	.764	.749	.725	.675

study, is that by applying MVSC to refine the output of spherical k -means, clustering solutions are improved significantly. Both rMVSC- I_R and rMVSC- I_V lead to higher *NMIs* and *Accuracies* than Spkmeans in all the cases. Interestingly, there are many circumstances where Spkmeans' result is worse than that of NMF clustering methods, but after refined by MVSCs, it becomes better. To have a more descriptive picture of the improvements, we could refer to the radar charts in Fig. 5.11. The figure shows details of a particular test case where $k = 5$. Remember that a test case consists of 50 different test sets. The charts display result on each test set, including the accuracy result obtained by Spkmeans, and the results after refinement by MVSC, namely rMVSC- I_R and rMVSC- I_V . For effective visualization, they are sorted in ascending order of the accuracies by Spkmeans (clockwise). As the patterns in both Fig. 5.11(a) and Fig. 5.11(b) reveal, improvement in accuracy is most likely attainable by rMVSC- I_R and rMVSC- I_V . Many of the improvements are with a considerably large margin, especially when the original accuracy obtained by Spkmeans is

Table 5.6
Clustering results on *Reuters-21578*

Algorithms	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
<i>NMI</i>						
Spkmeans	.370	.435	.389	.336	.348	.428
rMVSC- I_R	.386	.448	.406	.347	.359	.433
rMVSC- I_V	.395	.438	.408	.351	.361	.434
MVSC- I_R	.377	.442	.418	.354	.356	.441
MVSC- I_V	.375	.444	.416	.357	.369	.438
NMF	.321	.369	.341	.289	.278	.359
NMF-NCW	.355	.413	.387	.341	.344	.413
<i>Accuracy</i>						
Spkmeans	.512	.508	.454	.390	.380	.429
rMVSC- I_R	.591	.592	.522	.445	.437	.485
rMVSC- I_V	.591	.573	.529	.453	.448	.477
MVSC- I_R	.582	.588	.538	.473	.477	.505
MVSC- I_V	.589	.588	.552	.475	.482	.512
NMF	.553	.534	.479	.423	.388	.430
NMF-NCW	.608	.580	.535	.466	.432	.493

low. There are only few exceptions where after refinement, accuracy becomes worst. Nevertheless, the decreases in such cases are small.

Finally, it is also interesting to notice from Table 5.5 and Table 5.6 that MVSC preceded by spherical k -means does not necessarily yields better clustering results than MVSC with random initialization. There are only a small number of cases in the two tables that rMVSC can be found better than MVSC. This phenomenon, however, is understandable. Given a local optimal solution returned by spherical k -means, rMVSC algorithms as a refinement method would be constrained by this local optimum itself and, hence, their search space might be restricted. The original MVSC algorithms, on the other hand, are not subjected to this constraint, and are able to follow the search trajectory of their objective function from the beginning. Hence, while performance improvement after refining spherical k -means' result by MVSC proves the appropriateness of MVS and its criterion functions for document clustering, this observation in fact only reaffirms its potential.

Fig. 5.11. *Accuracies* on the 50 test sets (in sorted order of $Spkmeans$) in the test case $k = 5$.

5.7 Conclusions

In this chapter, we propose a Multi-Viewpoint based Similarity measuring method, named MVS. Theoretical analysis and empirical examples show that MVS is potentially more suitable for text documents than the popular cosine similarity. Based on MVS, two criterion functions, I_R and I_V , and their respective clustering algorithms, MVSC- I_R and MVSC- I_V , have been introduced. Compared with other state-of-the-art clustering methods that use different types of similarity measure, on a large number of document datasets and under different evaluation metrics, the proposed algorithms show that they could provide significantly improved clustering performance.

The key contribution of this chapter is the fundamental concept of similarity measure from multiple viewpoints. Future methods could make use of the same principle, but define alternative forms for the relative similarity in Eq. (5.10), or do not use average but have other methods to combine the relative similarities according to the different viewpoints. Besides, the work presented in this chapter focuses on partitional clustering of documents. In the future, it would also be possible to apply the proposed criterion functions for hierarchical clustering algorithms. Finally, we have shown the application of MVS and its clustering algorithms for text data. It would be interesting to explore how they work on other types of sparse and high-dimensional data.

Chapter 6

Applications

6.1 Collecting Meaningful English Tweets

6.1.1 Introduction to Sentiment Analysis

The lightning speed growth of social media networks such as blogs, Facebook, Twitter, LinkedIn... has created a very rich and unending source of information on the Internet. Millions of people log into their Facebook or Twitter account everyday to share information, or to post their feeling or opinion about anything that matter to them. These pieces of information are then read and passed on by other millions of the social network users. The important part of this story is that among these users are also customers of some businesses, and the topics that they comment or give opinion about are products and services sold by these companies. Hence, this is a gold mine of collective and extremely useful information for the companies to study their market, for example: how customers rate a product; if they are happy and satisfied with a service or not; how customers react against a certain policy or advertisement that has just been carried out. The process of analyzing textual information on the web, extracting meaningful patterns and discovering online opinions, from which to support appropriate and fact-based decision making, is called Sentiment Analysis.

Sentiment Analysis needs the cooperation of different fields, including Natural Language Processing, Computational Linguistics, Text Analytics, Machine Learning and so on, in order to identify and extract subjective information correctly. Data Mining techniques are also useful and applied in Sentiment Analysis, therefore it is sometimes referred to as Opinion Mining. For example, Twitter Sentiment¹ - a result of a Stanford classroom project - is a tool that allow you to

¹<http://twittersentiment.appspot.com>

Fig. 6.1. Twitter Sentiment from a Stanford academic project.

discover sentiment about a product, brand or topic by collecting and classifying tweets. If you are thinking about buying an iPhone 4, and wondering what people think about this Apple mobile phone, you can have Twitter Sentiment find it out for you, as shown in Fig. 6.1. According to the latest finding by Twitter Sentiment based on tweets, 58% of the tweets that mention “iPhone 4” have positive sentiment, while the other 42% are negative. These figures show that there is still a strong divide in opinions about one of the hottest IT gadget recently. Examples of other online tools and web sites that provide similar sentiment analysis services are TweetFeel², OpinionCrawl³ and Twendz⁴.

So how does Sentiment Analysis, such as Twitter Sentiment, work? Fig. 6.2 describes the basic idea behind a Twitter sentiment analysis model. Almost everyday we have something to talk about. You either hate or love a movie that you just watch in the cinema. You are happy with the food at some restaurant, or you are perhaps very upset with a poor service provided by a company. With the rise of social networks and the convenience that IT technologies bring to us nowadays, people often love to post their thought to share it all over the world. A lot of people use tweets to express their feeling. About 140 million is the average number of tweets people send per day, according to Twitter blog’s number⁵ in March 2011. Hence, by querying Twitter’s database, a resourceful collection of people’s opinions about a particular topic can be retrieved. By

²<http://www.tweetfeel.com>

³<http://www.opinioncrawl.com>

⁴<http://twendz.waggeneredstrom.com>

⁵<http://blog.twitter.com>

Fig. 6.2. Twitter sentiment analysis.

processing, learning and analyzing the collection of data, a predictive model is able to conclude the overall sentiment, percentage of positive, negative or neutral, the people have on the topic.

The building of the predictive model can be simple or sophisticated depending on the particular approach that is used. The simplest model works by having a predefined list of positive and negative keywords or emoticons (for example, “love” is good and “boring” is bad) and scanning through a given text to count these words to categorize the text as positive or negative. More complex models involve linguistic or NLP techniques to recognize patterns. Another approach is to use supervised learning algorithms from Machine Learning. The model is built by allowing it to train from a set of labeled information, from which it learns the sentimental language. In Twitter sentiment analysis, the training data are tweets with known sentiment (positive or negative). Once trained, the model is applied to categorize the new and unseen tweets collected from Twitter database when a new topic is queried.

6.1.2 Applying GA-PM2C to Differentiate English from Non-English Tweets

Each method used to build the predictive model for sentiment analysis has its own advantages as well as shortcomings. However, we do not focus on the predictive model here, but rather the tweet data preprocessing step, which occurs after querying data from Twitter database and before feeding the data into the predictive model. That is where we demonstrate a useful application of our

clustering algorithm. It is obvious that the Twitter community consists of people from all over the world and, therefore, while English is a popular language, tweets may be posted in all sorts of other languages such as Spanish, French, Dutch and so on. In applications developed on English tweets only, there is a need to differentiate and separate English tweets from non-English ones. The Twitter API does have an option for us to inquire for English-only tweets. Nonetheless, tweets in other languages may still be returned together with the English ones. To address this problem, we use our algorithm GA-PM2C to cluster the tweet data into two major groups. As English should be the most popular language among all, the larger group should contain English tweets, and the other, smaller group should be formed by tweets in other languages.

One important property of tweet data is that they are extremely noisy. Being created in daily life, in a very casual environment and literally can be written by anyone, not every tweet is written in proper language. Some of them may just be meaningless phrases, let alone bearing any sentiment significance. Hence, these tweets are just noise and should be filtered out. Unlike methods such as k -means, our clustering algorithm GA-P2MC has the functionality to differentiate outliers and noisy data from the true samples.

To demonstrate the application of GA-PM2C, we use this algorithm to cluster a collection of 5,000 tweets that consists of English and other languages. It should be noted that removing non-English tweets is only one part of the preprocessing step. For each tweet, we need also to remove irrelevant words, including user names (often preceded by character '@'), common word “RT” (stand for “Re-Tweet”), numbers and icons. This removal is done before clustering. Besides, when applying our algorithm, we represented the tweets as tri-grams vectors, i.e. each attribute is a unique sequence of three consecutive characters. An interesting feature of tweet language is that there exist words such as “huuuuungry” or “booooring”. Hence, tri-grams that consist of three identical characters were also removed. The parameters specific to the algorithm are set as following: population size is 20; crossover rate is 0.5; the maximum number of generations is 60 and the mutation rate is $10e-3$. We estimate a contamination rate of 8% in this dataset.

Fig. 6.3 shows a snapshot of the tweet clustering result given by GA-PM2C. Just by eyeballing through the categorization of the tweets, we observe that the algorithm has performed a decent job by labeling most of the English tweets as one group and most of the non-English tweets as the other. The noisy tweets are also identified correctly most of the times. They are often either sentences

Tweets	Type
finally getting rid of this old phone. :D	English
I'm back in the UK. sorry for lack of tweeting but I was put on the slopes skiing :)	English
lol Blah -_- I take it back !	English
@faye_faye_xo it would be a funny scene !! I just constantly laugh :) I'm happy when I'm high lol and I love ... http://tmi.me/7bqQ0	English
@JBTourUpdates appearing on the movie! :P 2YearsJonas3DMovie	English
Sounds like a perfect day! :) RT @MClaireOConnor: Fourth fountain diet coke of the day. #outofcontrol #obsessed	English
RT @DizzCarter: @CraigMitchSuave @AineyZion yous 2 defoo know about Cavary the best Roasts!!! â€¢ Ishh soo nice :p	English
@MarryMeBieber_ yes it is -_- i was like, am i supposed to know whos that? :D anyways, ignore it :)	English
Create iPhone Apps without Programming http://goo.gl/e6aEy :) RT @Gamerztwit iPhone 5-Bildschirm gesichtet: GrÃ¼Ã¶eres Display fÃ¼r mehr Sp	English
@corspekkie ghehe, een beetje :p , kwerd t zelf ookal soortvan zat, al dat sirieuze gelul xd ghehe	NonEnglish
@BieberWorld_2 haha, en ge moet belgiumbelieber ne keer ontmoeten!!! :D en btw: hoe noemt ge eigenlijk? haha kweet da nog nie, he!	NonEnglish
@LovesBjarne woooww en @Dyensi houook (niet) van jou :D	NonEnglish
Twitter = the cure for my boredom. :)	English
Studying for this test tomorrow :/	English
@DollyRoseSN1 Lol Im not :)	English
O @manaao acabou de ir embora :(Preciso MUITO falar com ele, tenho uma Ã³tima noticia!	NonEnglish
RT @FatosJonas: Vamos fazer uma vaquinha e construir um paÃ³s sÃ³ para o pessoal do twitter ? :D	NonEnglish
@gabriel_levi oloco nao exagera. dormir = respirar pra mim! hahahah	NonEnglish
eu acho MESMO que o @pelurestart deveria seguir a @peluteamo ! :)	NonEnglish
SerÃ¡ que O Impostor conseguiu entrar no Oscar esse ano? Mal posso esperar parar comeÃ§ar PÃ¡nico logo! @ProgramaPanico hoje estÃ¡ demais! :)	NonEnglish
@Mr_Jamieson but that's not Portsmouth :(get to portsmouth! #anewfan!!	English
@ShakeltStyles they're doing like tour diaries :D xx	English
@justinnloveer ok good bye and i will tweet you if i back :) love you so much and forever <3 <3 <3	English
RT @shellssx: RT @ItsGriffinn: RT @SonnyKomen: #NF = #NFB x	noisy
@weedevil28 I think when there like 17 move in together 18, get married and like after that i think sophiewould want to adopt! :)	English
Just took in how @justinbieber and I have the same birthday which is 2 days :)	English
What a match!! The Cricket World Cup 2011 has truly begun. :)	English
@justinbieber 's speech "im from a small town of 30,000 people" WE GET IT, WE GET IT BIEBS :) !!lol	English
@ThatCheesyVader I guess so. Haha. :)	NonEnglish
RT "@LoveKeyanna: i feel like i was drinking last night. O_o" --- #alchy lol	English
@alisonjonesxx thanks :)	English
ohhh & addd my faceboook ladies & gents :) facebook.com/cynthiatiwaa	English
@lizwoolly Nope.. It really is a bad job of a tattoo :)	English
Y que???? No voy y que??? Muy mi Vida :)	NonEnglish
@thaisliira KKKKK, me respeite! eu lembro de tudo, pode perguntar ;D	NonEnglish
Acabei de criar um Quiz: "Voce sabe tudo sobre Bill Kaulitz?" e voce pode responder agora! :) http://rdae.eu/h4ZCnO	NonEnglish
RT @SequenciasFodas: Estou no MSN > ComeÃ§am a ficar Online e Offline pra subir Plaquinhas > Fico puto > Bloqueio > FIM :D	NonEnglish
@carolinijuste Issae ;)	noisy
@PurpleNinja :P <3	noisy
Ben cÃ±kÃ±yoruum âˆž YarÃ±n okuul var .. SÃ±kÃ±cÃ± gunu gorun o zamaaan :P Ã°yi akÃ°ymlaar.	NonEnglish
ik ga vroeg slapen vandaag O_o #geloofjehetzelf #twexit	NonEnglish
@BeyonddLove sweeet, hope you have a great day at church... :) how was your night?	English

Fig. 6.3. A snapshot of tweet clustering result by GA-PM2C algorithm.

that are too short to be understood properly or phrases that contain meaningless characters. Some tweets are also reasonably detected as noisy because of language encoding error.

The detection of noisy tweet data are desirable because, practically, we are only interested in proper English tweets. To further demonstrating the benefit of using GA-PM2C, we compared its clustering result with that given by Spherical k -means. The examples of tweets being classified differently by GA-PM2C and Sp k means are shown in Fig. 6.4. Ignoring the fact that there are some tweets identified by GA-PM2C correctly as English (non-English) but recognized as non-English (English) by Sp k means, let us pay more attention on the tweets that are marked as noise in the left column by GA-PM2C. Most of them should be indeed considered as noise because the information they contain is unclear and useless. It is practically reasonable to treat them as neither English nor non-English tweets, so that they do not affect the categorization of the other tweets. For Sp k means, as there is no option for noisy data, the algorithm inconsistently assign them as either English or non-English, although the tweets do not really belong to any type.

6.2 Web Search Result Clustering with MVSC

6.2.1 Overview of Web Search Result Clustering

In Section 2.4.1, we have discussed a few potential applications of clustering techniques in Web Mining and Information Retrieval. In this second part of the chapter, we focus on one particular application area - the use of clustering algorithms for enhancing the organization and presentation of web search results. We explain how our MVSC algorithm is integrated into an existing open source web search and clustering software, and demonstrate how it is used to categorize web pages returned by popular search engines such as Bing and Yahoo.

The systematic procedure of a web search and clustering engine typically consists of the following steps: firstly, retrieve search results according to user's query; secondly, preprocess the returned information so that they are ready for the clustering and other processing steps; next, cluster the web pages into sub-topics; subsequently, build the labels that summarize the sub-topics; finally, visualize the clusters and present them to the user. Fig. 6.5 illustrates the overall picture of the procedure. In this picture, step (1)- information retrieval- is performed by usual web search engines (e.g. Google, Yahoo, Bing). Step (4)- visualization of results- is implemented and customized by specific software

[illegible]

Fig. 6.4. Examples of tweets classified differently by GA-PM2C & Spkmeans.

Fig. 6.5. Web search and clustering.

systems. We will focus more on steps (2) and (3) where clustering algorithms are involved.

It should be noted that text clustering in the context of online web search has some distinctive characteristics compared to offline document clustering. Applying our clustering algorithm here is little different from what we have done in the previous chapters. We list a few points that have impacts on the algorithm implementation below:

- **Computation time:** Needless to say, one of the important characteristics is that the time it takes to return the search results has to be fast (how fast: less than a fraction of a second or so). Take longer than that and your users will lose patience and walk away. Due to this strict requirement in computation time, for each web page, only its URL, title and a snippet that summarizes its content are used for the clustering part. This practice helps to reduce the total number of words, i.e. the number of feature dimensions, and in turn reducing the computational demand. It differs from offline document clustering, in which full document's content is used.
- **Topic labels:** Another additional requirement in web search result clustering is that the returned clusters must also be tagged with some labels which describe their topics. This is necessary for users to see what a

cluster of web pages is about. The labels will become part of the visual representation in the final output.

- Clustering and labeling: To address topic label construction problem, some technique of summarization or representative feature selection is needed. Web clustering systems differ from each other in how this task is carried out in step (3) in Fig. 6.5. Some systems extract representative words or phrases for the clusters after clustering algorithms are performed; some start with finding a set of descriptive label candidates first before assigning snippets to the labels to form clusters; other systems would do the two tasks of clustering and labeling simultaneously, for example by using a co-clustering algorithm.
- Stability of results: The final goal of applying clustering to web search is to present to users the retrieved web pages in a more informative format. As a side but equally important requirement, there must be some stability in the way we return the search results. Given the same set of web pages, the same grouping must be produced every time. Users should not see drastic changes in the system's recommendations when they repeat the same query, at least within the same querying session. In algorithms such as k -means and our MVSC, where clustering output is sensitive to initial stage, special initialization technique is required to handle this situation.

There are currently quite a few web search and clustering systems that have been fully developed in the market. They can be either results of research-focused projects or complete products of commercial companies. The outstanding examples in this field include Vivisimo Velocity⁶, WebClust⁷, Yippy⁸ and Carrot Search⁹. Many of such systems are meta search engines: they do not actually crawl or index the web, but redirect queries to several other search engines, then combine and process the results from these multiple sources. They focus on making major improvements, through clustering algorithms, in organizing and presenting the information to users.

In the next section, we demonstrate how our algorithm, MVSC- I_V , can be applied to play the clustering role in a similar system. We make use of an open source software called Carrot², which is a lab project version¹⁰ developed by

⁶<http://vivisimo.com>

⁷<http://www.webclust.com>

⁸<http://yippy.com>

⁹<http://carrotsearch.com>

¹⁰<http://www.carrot2.org>

Fig. 6.6. A screenshot of Carrot²'s GUI.

the founders of Carrot Search company. We integrate our implementation of MVSC- I_V into Carrot² framework to perform some real web search and clustering activities.

6.2.2 Integration of MVSC into Carrot² Search Result Clustering Engine

6.2.2.1 System Settings

Carrot² implements several different clustering algorithms, including STC [165], Lingo [166,167] and bisecting k -means. It also has ready-to-use APIs to retrieve search results from many sources such as Bing¹¹, Yahoo¹², eTools¹³ and so on. What is more, Carrot² also provides some very interesting visualizations of results, and a benchmarking tool to measure clustering performance. A screenshot of Carrot²'s GUI is shown in Fig. 6.6.

In order to apply the MVSC algorithm in Carrot²'s web clustering framework, we made some specific improvements to the algorithm's implementation

¹¹<http://www.bing.com>

¹²<http://ch.search.yahoo.com>

¹³<http://www.etoools.ch>

compared to the previous chapter. As mentioned in the preceding section, stability of clustering results is important in a practical and user-oriented system. Therefore, random initialization, like in previous implementation of MVSC, is not very suitable in this case. To address this issue, we chose Singular Value Decomposition (SVD) as a tool to find the initial clusters. As it has been studied, SVD technique decomposes the $d \times n$ term-document matrix $\mathbf{X}$ into $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^t$, where $\mathbf{U}$ is $d \times k$, $\mathbf{S}$ is $k \times k$ and $\mathbf{V}$ is $n \times k$. The k column vectors of matrix $\mathbf{U}$ form the orthogonal basis of the term space, and can be considered as the approximations of the k main topics. Hence, we selected the k column vectors of $\mathbf{U}$ as representatives of the initial clusters, and assigned the web page feature vectors to a cluster having the closest representative. As a result, we always obtained a consistent initialization.

One drawback of the above strategy is that it incurs extra computational demand because of the SVD. While the clustering algorithm was designed to be computationally efficient, the additional computation led to more than double of the clustering time itself. The second option, which we had experimented to find satisfying performance, was to initialize the feature vectors to clusters “randomly”, but according to the order of the returned web pages. For 1 to n feature vectors entered into the clustering algorithm in order, the feature vectors 1 to k were assigned to cluster 1 to cluster k respectively; this procedure was repeated for each of the next k feature vectors. This strategy still assures randomness but unique initialization for a particular ordered set of web pages.

Besides stability in solution, another issue we need resolve is the construction of cluster labels. While this is an aspect as critical and important as the clustering quality, we are more interested in demonstrating the clustering functionality here. Therefore, we resorted to simple, but showed to be effective enough, method. The method is as follows:

- Obtain the clusters returned by the clustering process
- Specify the expected number of words L to have in cluster label
- Specify a threshold parameter $p, 0 \leq p \leq 1$
- For each cluster j , find the word with the largest feature value $D_{j,max}$ in the centroid vector D_j , $j = 1, \dots, k$; $max \in \{1, \dots, d\}$
- For each cluster j , select up to L words w_l such that: $D_{jl} \geq p \times D_{j,max}$, to form the cluster label, $l \in \{1, \dots, d\}$

We usually selected maximum 2 or 3 words to have in the cluster labels, as these are reasonable label length. The control parameter p is used so that only the most relevant words are considered; we set $p = 0.70$ in our study. According to the above procedure, the construction of topic labels are carried out after the clusters are found. Our method is different from the STC and Lingo methods used in Carrot². The latter two employ an inverse approach: they start the clustering process by finding a set of potential cluster labels first, and only then carry on to assign web pages to the relevant labels. For more details about STC and Lingo, readers can refer to their respective papers [165] and [166, 167].

Finally, we need to discuss the problem of cluster number. While there are quite a number of algorithms have been developed to determine the number of clusters automatically, to our knowledge no one can provide a completely satisfying solution. The Lingo algorithm in Carrot² also define the cluster number automatically, though through the setting of another predefined threshold parameter. Our clustering algorithm does not such a functionality, but it is possible to employ a method similar to what Lingo uses by setting a threshold for the ratio of the Frobenius norms of the SVD-induced matrix $\mathbf{X}_k$ and the original term-document matrix $\mathbf{X}$. Nevertheless, we observed that for web clustering scenarios, there is not an exact answer to the number of clusters. A good value should fall around 10 to 20 clusters. For our study, we decided to use the default setting of STC algorithm in Carrot², which is to generate 16 clusters every time. All the other algorithms were tuned to produce the same number of clusters. This practice, moreover, enables us to compare the clustering algorithms more easily.

6.2.2.2 A Web Clustering Scenario

After all the system implementation were done, we performed some web querying activities. In the following example, we searched the web with the keyword “apple”. “Apple” is the name of one of the most popular companies in the world nowadays, so we could expect a lot of web pages returned are about this company. On the other hand, “apple” is also the name of a popular fruit. The number of web pages to be returned was capped at 200. The same search results were then processed by four clustering algorithms: STC, Lingo, k -means and MVSC- I_V (renamed in the system as MVSC2). The clusters suggested by the respective algorithms are shown in Fig. 6.7.

It can be observed from Fig. 6.7 that MVSC2 and the other algorithms recommend some common clusters with similar labels, for examples: “Mac”,

Fig. 6.7. Clusters with topic labels recommended for query "apple".

Cider, Fruit
<i>Apple juice is a fruit juice manufactured by the maceration and pressing of apples. ... Apple juice is one of the most common fruit juices in the world, with world ...</i>
<i>An apple martini (appletini for short) is a cocktail containing vodka and one or more of apple juice, apple cider, apple liqueur, or apple brandy. ...</i>
<i>Apple sauce or applesauce is a purée made of apples. It can be made with peeled or unpeeled apples and a variety of spices (commonly cinnamon and allspice) ...</i>
iPhone
<i>Apple created the device during a secretive and unprecedented collaboration with ... Apple rejected the "design by committee" approach that had yielded the Motorola ...</i>
<i>Apple had already received the iPhone prototype prior to the raid when it was ... Steve Jobs, Apple CEO, holding a white iPhone 4. The white iPhone 4 would not be ...</i>
<i>At the time he had been considering having Apple work on tablet PCs, which later came to ... Apple closed its stores at 2:00 PM local time to prepare for the 6:00 ...</i>
Keyboard, Mouse
<i>For the specific wired model currently sold as the "Apple Mouse", see Apple Mighty Mouse. ... The Apple Mouse began as one of the first commercial mice available to consumers. ...</i>
<i>The Apple Keyboard is a keyboard designed by Apple first for the Apple line, then the Macintosh line of computers. It is available in both wired ...</i>
<i>The Apple Adjustable Keyboard was an ergonomic and adjustable keyboard introduced by Apple Computer in 1993 ... The last Apple computer released compatible with this keyboard ...</i>

Fig. 6.8. Clusters with representative snippets.

“Software”, “Steve Jobs”, “MacBook” and “Macintosh”. These are all themes related to Apple Inc. What is more, MVSC2 also produces some very sensible clusters, such as “iPhone” (one of the hottest groundbreaking products of Apple Inc.), “Keyboard, Mouse” (2 related computer equipments), “iTunes, iWork” (2 related products of Apple Inc.). Interestingly, another distinct topic that MVSC2 is able to find out is “Cider, Fruit”, which is about apple fruit and juice. This is very encouraging outcome, because it is surely not easy to dig out this topic among the overwhelming data on Apple Inc. Some examples of the clusters and their representative snippets are shown in Fig. 6.8 , and the visualization of the clusters created by Carrot² is displayed in Fig. 6.9.

Fig. 6.9. MVSC2's clusters visualized by Carrot².

Table 6.1
Clustering time (in second).

Algorithm	Avg. Time	Std. Dev.	Min. Time	Max. Time
MVSC2 + SVD	.073	.002	.071	.080
MVSC2	.039	.001	.037	.041
STC	.041	.003	.038	.057
Lingo	.202	.074	.181	.787
<i>k</i> -means	.345	.005	.341	.378

Finally, to examine MVSC2's performance in terms of computation time, we used Carrot²'s benchmarking tool to measure the clustering time spent by the algorithms to produce the above results. The time durations in second are recorded in Table 6.1. STC implements an efficient suffix-tree data structure, so it is expected to be the most speedy among all the algorithms. Our implementation of MVSC2 with SVD as initialization technique needed nearly double the amount of time required by STC. However, it was still faster than Lingo and *k*-means by a large margin. *k*-means is expected to have approximately the same computational demand with Lingo; however, in this case it required the longest clustering time. In the third row of Table 6.1 is the MVSC2 that was not initialized by SVD, but by the second technique explained in Section 6.2.2.1. As discussed before and backed by the time values measured in the Table, the clustering time in this second implementation was reduced dramatically, to even slightly faster than STC, because no SVD computation was involved. This reduction in clustering time proves that the core computation in MVSC2 is really

efficient.

In this chapter, we have demonstrated the use of our clustering algorithms in two real-life interesting applications. Both case studies have showed that the proposed algorithms are practically useful, and able to perform the tasks presented to them effectively and efficiently.

Chapter 7

Conclusions

7.1 Summary of Research

The research work in this thesis emphasizes on the development of novel data clustering algorithms. The objective of clustering is high-dimensional data, which are in most of our cases web or text documents. While our focus is on proposing new concepts and developing effective and efficient algorithms, we also demonstrate that the proposed work has practical use in real-life related application areas. The following paragraphs summarize our research study in this thesis.

In Chapter 2, we have carried out a literature survey of the important background knowledge in the field of data clustering. It includes a variety of existing clustering algorithms and systems, together with their applications in various domains. We have also pointed out some critical problems that researchers have encountered when working with high-dimensional data. The challenges to address these problems are the main motivation for the research work in this thesis. They are also the common challenges for the data clustering community.

In Chapter 3, we have performed theoretical and empirical analysis of different models of probabilistic mixture-based clustering approach, and proposed two techniques for improving the related algorithms. Empirical experiments have been implemented to compare the Gaussian and von Mises-Fisher models with other well-known methods such as the k -Means variants and the recently proposed NMF.

The impacts of high dimensionality of data on various characteristics of mixture model-based clustering (M2C) have been analyzed. The understanding of these impacts is very useful for the research of better solutions to the unsupervised text classification problem. The fact is that some model selection methods,

which have been designed successfully for low-dimensional domains, no longer work well on text documents. Besides, the soft-assignment characteristic of M2C does not remain the same on the sparse and high-dimensional space. And therefore, the issue of sensitiveness to initialization is also more difficult to cope with.

In addition to the analysis, we have also proposed two techniques to improve the clustering quality of M2C methods when applied on text data. The first technique uses a mixture of the directional distributions von Mises-Fisher to decompose the term space and, as a result, reduce the feature dimensions. The second is an annealing-like technique which aims to improve the initial phase of EM algorithm for high-dimensional Gaussian model. During the early stage of EM, the ellipsoids of the Gaussian components are maintained at large size while the model parameters are being adjusted to more sensible initial values. When the ellipsoids are gradually compressed, the change in document assignments among clusters occurs smoothly. Experiments have shown that our techniques lead to good improvement in clustering results compared with existing methods.

With Chapter 4, we have proposed the Partial M2C framework which takes into consideration the existence of outliers during clustering. In this framework, a Model Sample Selection step is performed to determine whether a data observation is either generated from a probabilistic model or it is an outlier. From this framework, we also proposed the GA-based Partial M2C algorithm, or GA-PM2C. Techniques from GA and a newly designed Guided Mutation operation help the algorithm filter out noisy data and outliers to produce better and more reliable clusters.

In Chapter 5, we have introduced the Multi-Viewpoint based Similarity measure, or MVS. This sparse and high-dimensional vector based similarity measure has the potential to be more suitable for document clustering than the cosine measure. The main novelty of this work is the fundamental concept of similarity measure from multiple viewpoints, which have been explained clearly. Based on this new concept, we have formulated two clustering criterion functions I_R and I_V , and developed the respective clustering algorithms called MVSC- I_R and MVSC- I_V . The interesting thing about our algorithms is that they are as simple and easy to implement as the popular k -Means algorithm, but they have been shown to be significantly more effective than k -Means. Since the latter is used widely in many real-life applications, our proposed algorithms have the potential to be very applicable and useful too.

Finally, Chapter 6 has showcased the practical scenarios in which our pro-

posed algorithms are used to solve real-world problems. When applied to the task of differentiating English and non-English tweets from Twitter, GA-PM2C is not only able to cluster a set of tweets into English and non-English data, but also recognize the noisy, abnormal tweets. In another scenario, we have used our MVSC algorithm to perform the clustering task in a web search and clustering system. Web search result clustering is an exciting and engaging activity in terms of both research challenge and industrial interest; our algorithm has exhibited some promising results in this application area.

7.2 Future Work

The research described in this thesis has produced new concepts, techniques and algorithms that help to improve clustering performance in high-dimensional data domain. As we have mentioned in the respective chapters, there are some potential future research directions that can be continue from this research.

Similar to many research studies of data clustering, our approaches are based on the assumption that the number of clusters is known or pre-selected. This given information is, in fact, truly available in many real life situations. There have been quite a number of proposed works that aims to find the natural number of data clusters automatically. However, there is not yet any method that can claim to yield correct number for every data set. We can expect this to be a very challenging and difficult problem to solve. The fact is that for any collection of data, there is always more than one way to perceive and divide into groups. For example, given a set of documents, even two human readers can categorize them into different sub-topics, depending on their personal understanding and appreciation of the contents. Nonetheless, in the case where there is no requirement for an exactly correct number of clusters, the ability to reasonably estimate this number will provide a good advantage. In this thesis, such cases are: how many number of features to remain after the FR procedure (Chapter 3), and how many topics a group of web pages should be divided into (Chapter 6). A proper model selection method or even a heuristic technique that can help to decide an appropriate and reasonable number would be very useful.

In the Partial M2C framework, other types of algorithm and fitness function can be designed for the Model Sample Selection stage, rather than GA and the trimmed likelihood function. We would like to emphasize a possible combination of discriminative approach and generative approach in this stage to perform

the model sample selection task. In addition, another huge improvement we aim to make in future work is to be able to determine the contamination level automatically, or to adjust the level dynamically when data change.

As mentioned earlier, the main novelty of MVSC is the principle of measuring similarity from multiple viewpoints. From this concept, it is possible to define new forms of similarity, as well as to formulate new forms of clustering criterion functions. It would also be interesting to explore whether the similarity measure and criterion functions can be applied effectively to other types of clustering, such as hierarchical clustering and semi-supervised clustering. Moreover, we have explained in the web search result clustering application how a relatively simple procedure has been used to define the topic labels from our algorithm. The resulted labels are formed by groups of individual words, although complete phrases should be more comprehensive. In order to improve the algorithm's effectiveness, especially from the perspective of user interpretation of the categorized results, more sophisticated label construction techniques can be developed. A topic summarization method or an appropriate phrase detection algorithm can be employed here to derive topic labels from the contents of the clusters. Such improvements will surely add in even greater values to the system.

Finally, in this thesis, we have carried out experiments and implemented applications with text document and web content data. Nevertheless, there are other forms of high-dimensional data that also need to be studied. Gene microarray data are a good example. Future extension of our studies to other types of high-dimensional data and application domains will definitely provide more insights into other facets of the proposed algorithms. There is still a very long way for the research community to be able to find out the "best" clustering algorithm. We hope that our research can help to address a few challenging problems encountered in the field, and bring us a few step closer to more effective and efficient clustering.

Author's Publications

1. D. Thang Nguyen, L.H. Chen and C.K. Chan, "Clustering with Multi-Viewpoint Based Similarity Measure," *IEEE Transactions on Knowledge and Data Engineering*, preprint, Apr. 2011, doi=10.1109/TKDE.2011.86.
2. D. Thang Nguyen, L.H. Chen and C.K. Chan, "Robust mixture model-based clustering with genetic algorithm approach," *Intelligent Data Analysis*, vol. 15, no. 3, pp. 357-373, IOS Press, Jan. 2011.
3. D. Thang Nguyen, L.H. Chen and C.K. Chan, "Multi-viewpoint based similarity measure and optimality criteria for document clustering," In *Proc. of the 6th Asia Information Retrieval Societies Conference 2010*, LNCS 6458, pp. 49-60, 2010.
4. D. Thang Nguyen, L.H. Chen and C.K. Chan, "An outlier-aware data clustering algorithm in mixture models," In *Proc. of the 7th International Conference on Information, Communication and Signal Processing (ICICS 2009)*, pp. 1-5, 8-10 Dec. 2009.
5. D. Thang Nguyen, L.H. Chen and C.K. Chan, "Feature reduction using mixture model of directional distributions," In *Proc. of the 10th International Conference on Control Automation Robotics & Vision: ICARV2008*, vol. 1, no. 4, pp. 2208-2212, 2008.
6. D. Thang Nguyen, L.H. Chen and C.K. Chan, "An enhanced EM algorithm for improving Gaussian model-based clustering of high-dimensional data," Submitted for publication.

Bibliography

- [1] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” *ACM Comput. Surv.*, vol. 31, pp. 264–323, September 1999.
- [2] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern Recognit. Lett.*, Sep 2009.
- [3] P. Berkhin, “Survey of clustering data mining techniques,” tech. rep., Accrue Software, San Jose, CA, 2002.
- [4] R. Xu and Li, “Survey of clustering algorithms,” *IEEE Trans. on Neural Networks*, vol. 16, pp. 645–678, May 2005.
- [5] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proc. 5th Berkeley Symp.*, vol. 1, 1967.
- [6] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, “Top 10 algorithms in data mining,” *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2007.
- [7] I. S. Dhillon and D. S. Modha, “Concept decompositions for large sparse text data using clustering,” *Mach. Learn.*, vol. 42, no. 1/2, pp. 143–175, 2001.
- [8] M. Steinbach, G. Karypis, and V. Kumar, “A comparison of document clustering techniques,” In *Proceedings of Workshop on Text Mining, 6th ACM SIGKDD International Conference on Data Mining (KDD’00)*, pp. 109–110, August 20–23 2000.
- [9] E. Y. Chan, W.-K. Ching, M. K. Ng, and J. Z. Huang, “An optimization algorithm for clustering using weighted dissimilarity measures,” *Pattern Recognition*, vol. 37, no. 5, pp. 943–952, 2004.

- [10] L. Jing, M. K. Ng, J. Xu, and J. Z. Huang, "Subspace clustering of text documents with feature weighting k-means algorithm," in *PAKDD*, pp. 802–812, 2005.
- [11] S. Zhong, "Efficient online spherical k-means clustering," *IEEE International Joint Conference on Neural Networks*, vol. 5, pp. 3180–3185, 2005.
- [12] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [13] T. Kohonen, *Self-Organizing Maps*. Springer, 2001.
- [14] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, V. Paatero, and A. Saarela, "Self-organization of a massive document collection," *IEEE Transactions on Neural Networks*, vol. 11, pp. 574–585, 2000.
- [15] K. Lagus, S. Kaski, and T. Kohonen, "Mining massive document collections by the websom method," *Inf. Sci.*, vol. 163, no. 1-3, pp. 135–156, 2004.
- [16] G. Yen and Z. Wu, "A self-organizing map based approach for document clustering and visualization," *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pp. 3279–3286, 2006.
- [17] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.
- [18] R. Krishnapuram and J. Keller, "A possibilistic approach to clustering," *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 98–111, May 1993.
- [19] N. R. Pal, K. Pal, J. M. Keller, and J. C. Bezdek, "A possibilistic fuzzy c-means clustering algorithm," *IEEE Trans. on Fuzzy Systems*, vol. 13, pp. 517–530, Aug. 2005.
- [20] K. Kummamuru, A. Dhawale, and R. Krishnapuram, "Fuzzy co-clustering of documents and keywords," *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, vol. 2, pp. 772–777 vol.2, 2003.
- [21] H. Frigui and O. Nasraoui, "Simultaneous clustering and dynamic keyword weighting for text documents," in *Survey of Text Mining* (M. W. Berry, ed.), pp. 45–72, Springer, 2003.

- [22] W.-C. Tjhi and L. Chen, “Possibilistic fuzzy co-clustering of large document collections,” *Pattern Recognition*, vol. 40, pp. 3452–3466, DEC 2007.
- [23] W. Xu, X. Liu, and Y. Gong, “Document clustering based on non-negative matrix factorization,” in *SIGIR*, pp. 267–273, 2003.
- [24] T. L. C. Ding and M. Jordan, “Convex and semi-nonnegative matrix factorizations for clustering and low-dimensional representation,” tech. rep., Lawrence Berkeley National Laboratory, 2006.
- [25] T. Li and C. Ding, “The relationships among various nonnegative matrix factorization methods for clustering,” in *ICDM ’06: Proceedings of the Sixth International Conference on Data Mining*, (Washington, DC, USA), pp. 362–371, IEEE Computer Society, 2006.
- [26] C. Ding, T. Li, W. Peng, and H. Park, “Orthogonal nonnegative matrix tri-factorizations for clustering,” in *KDD ’06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 126–135, ACM, 2006.
- [27] C. Boutsidis and E. Gallopoulos, “SVD based initialization: A head start for nonnegative matrix factorization,” *Pattern Recognition*, vol. 41, pp. 1350–1362, APR 2008.
- [28] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, “Algorithms and applications for approximate nonnegative matrix factorization,” *Computational Statistics & Data Analysis*, vol. 52, pp. 155–173, SEP 15 2007.
- [29] J. Y. Zien, M. D. F. Schlag, and P. K. Chan, “Multilevel spectral hypergraph partitioning with arbitrary vertex sizes,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 18, no. 9, pp. 1389–1399, 1999.
- [30] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 888–905, 2000.
- [31] C. Ding, X. He, H. Zha, M. Gu, and H. Simon, “A min-max cut algorithm for graph partitioning and data clustering,” in *IEEE ICDM*, pp. 107–114, 2001.
- [32] G. Karypis, “CLUTO a clustering toolkit,” tech. rep., Dept. of Computer Science, Uni. of Minnesota, 2003. <http://glaros.dtc.umn.edu/gkhome/views/cluto>.

- [33] I. S. Dhillon, “Co-clustering documents and words using bipartite spectral graph partitioning,” in *KDD*, pp. 269–274, 2001.
- [34] M. Li and L. Zhang, “Multinomial mixture model with feature selection for text clustering,” *Knowledge-Based Systems*, 2008. Article in Press.
- [35] T. Zhang, Y. Tang, B. Fang, and Y. Xiang, “Document clustering in correlation similarity measure space,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2011.
- [36] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Chang, “Parallel spectral clustering in distributed systems,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, pp. 568–586, march 2011.
- [37] S. Bandyopadhyay and S. Saha, “Gaps: A clustering method using a new point symmetry-based distance measure,” *Pattern Recognition*, vol. 40, pp. 3430–3451, 2007.
- [38] K. Krishna and M. Narasimha Murty, “Genetic k-means algorithm,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, pp. 433–439, jun 1999.
- [39] H. jun Sun and L. huan Xiong, “Genetic algorithm-based high-dimensional data clustering technique,” in *Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on*, vol. 1, pp. 485–489, aug. 2009.
- [40] M.-F. Pernkopf and S. M.-D. Bouchaffra, “Genetic-based em algorithm for learning gaussian mixture models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1344–1348, 2005.
- [41] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay, “Multiobjective genetic algorithm-based fuzzy clustering of categorical attributes,” *Evolutionary Computation, IEEE Transactions on*, vol. 13, pp. 991–1005, oct. 2009.
- [42] T. Ozyer and R. Alhajj, “Parallel clustering of high dimensional data by integrating multi-objective genetic algorithm with divide and conquer,” *APPLIED INTELLIGENCE*, vol. 31, pp. 318–331, DEC 2009.
- [43] G. McLachlan and K. Basford, *Mixture Models: Inference and Applications to Clustering*. New York: M.Dekker, 1988.

- [44] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: John Wiley & Sons, 2000.
- [45] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. New York: John Wiley & Sons, 1997.
- [46] S. Dasgupta, “Learning mixtures of gaussians,” in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pp. 634–644, 1999.
- [47] C. Constantinopoulos and A. Likas, “Unsupervised learning of gaussian mixtures based on variational component splitting,” *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 745–755, 2007.
- [48] N. Ueda and Z. Ghahramani, “Bayesian model search for mixture models based on optimizing variational bounds,” *Neural Networks*, vol. 15, pp. 1223–1241, DEC 2002.
- [49] A. Corduneanu and C. M. Bishop, “Variational Bayesian model selection for mixture distributions,” in *Artificial Intelligence and Statistics*, 2001.
- [50] P. Berkhin, “Web mining research: a survey,” tech. rep., Accrue Software, San Jose, California, 2002.
- [51] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in *Proceedings of ICML-97, 14th International Conference on Machine Learning* (D. H. Fisher, ed.), (Nashville, US), pp. 412–420, Morgan Kaufmann Publishers, San Francisco, US, 1997.
- [52] T. Liu, S. Liu, Z. Chen, and W.-Y. Ma, “An evaluation on feature selection for text clustering,” in *ICML*, pp. 488–495, 2003.
- [53] J. Novovicova, A. Malik, and P. Pudil, “Feature selection using improved mutual information for text classification,” *Structural, Syntactic, and Statistical Pattern Recognition, Proceedings*, vol. 3138, pp. 1010–1017, 2004.
- [54] F. Song, D. Zhang, Y. Xu, and J. Wang, “Five new feature selection metrics in text categorization,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, pp. 1085–1101, SEP 2007.
- [55] Y. Li, C. Luo, and S. M. Chung, “Text clustering with feature selection by using statistical data,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 20, pp. 641–652, MAY 2008.

- [56] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.
- [57] K. Lerman, "Document clustering in reduced dimension vector space." <http://www.isi.edu/~lerman/papers/Lerman99.pdf>, 1999.
- [58] W. Song and S. C. Park, "A novel document clustering model based on latent semantic analysis," in *SKG '07: Proceedings of the Third International Conference on Semantics, Knowledge and Grid*, (Washington, DC, USA), pp. 539–542, IEEE Computer Society, 2007.
- [59] J. Meng, H. Mo, Q. Liu, L. Han, and L. Weng, "Dimension reduction of latent semantic indexing extracting from local feature space," *Journal of Computational Information Systems*, vol. 4, no. 3, pp. 915–922, 2008.
- [60] B. Draper, D. Elliott, J. Hayes, and K. Baek, "EM in high-dimensional spaces," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, pp. 571–577, june 2005.
- [61] S. Dasgupta, "Experiments with random projection," in *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pp. 143–151, 2000.
- [62] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review.," *SIGKDD Explorations*, vol. 6, no. 1, pp. 90–105, 2004.
- [63] M. Law, M. A. Figueiredo, and A. K. Jain, "Simultaneous feature selection and clustering using mixture models," *IEEE Trans. On Pattern Analysis and Machine Interlligence*, vol. 26, no. 9, September 2004.
- [64] C. Constantinopoulos and M. K. Titsias, "Bayesian feature and model selection for gaussian mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 1013–1018, 2006. Senior Member-Aristidis Likas.
- [65] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The Computer Journal*, vol. 41, pp. 578–588, 1998.
- [66] C. S. Wallace and D. L. Dowe, "Minimum message length and Kolmogorov complexity," *The Computer Journal*, vol. 42, no. 4, pp. 270–283, 1999.

- [67] M. A. T. Figueiredo and A. K. Jain, “Unsupervised learning of finite mixture models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 381–396, 2002.
- [68] H. Wang, B. Luo, Q. bing Zhang, and S. Wei, “Estimation for the number of components in a mixture model using stepwise split-and-merge em algorithm,” *Pattern Recognition Letters*, vol. 25, no. 16, pp. 1799–1809, 2004.
- [69] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton, “SMEM algorithm for mixture models,” *Neural Computation*, vol. 12, no. 9, pp. 2109–2128, 2000.
- [70] Z. Zhang, C. Chen, J. Sun, and K. L. Chan, “Em algorithms for gaussian mixtures with split-and-merge operation,” *Pattern Recognition*, vol. 36, no. 9, pp. 1973–1983, 2003.
- [71] B. Zhang, C. Zhang, and X. Yi, “Competitive em algorithm for finite mixture models,” *Pattern Recognition*, vol. 37, no. 1, pp. 131–144, 2004.
- [72] A. S. Hadi, “A modification of a method for the detection of outliers in multivariate samples,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 56, no. 2, pp. 393–396, 1994.
- [73] D. G. Calo, “Mixture models in forward search methods for outlier detection,” in *Data Analysis, Machine Learning and Applications*, pp. 103–110, 2007.
- [74] J. D. Banfield and A. E. Raftery, “Model-based Gaussian and non-Gaussian clustering,” *Biometrics*, vol. 49, pp. 803–821, 1993.
- [75] C. Hennig, “Breakdown points for maximum likelihood estimators of location-scale mixtures,” *Ann. Statist.*, vol. 32, pp. 1313–1340, 2004.
- [76] A. Atkinson and M. Riani, “The forward search and data visualisation,” *Computational Statistics*, vol. 19, no. 1, pp. 29–54, 2004.
- [77] D. Coin, “Testing normality in the presence of outliers,” *Statistical Methods and Applications*, vol. 17, no. 1, pp. 3–12, 2008.
- [78] G. Macintyre, J. Bailey, D. Gustafsson, I. Haviv, and A. Kowalczyk, “Using gene ontology annotations in exploratory microarray clustering to un-

- derstand cancer etiology,” *Pattern Recogn. Lett.*, vol. 31, pp. 2138–2146, October 2010.
- [79] J.-P. Brunet, P. Tamayo, T. Golub, and J. Mesirov, “Metagenes and molecular pattern discovery using matrix factorization,” *Proc. of The National Academy of Sciences*, vol. 101, pp. 4164–4169, 2004.
 - [80] T. Grotkjr, O. Winther, B. Regenberg, J. Nielsen, and L. K. Hansen, “Robust multi-scale clustering of large dna microarray datasets with the consensus algorithm,” *Bioinformatics/computer Applications in The Biosciences*, vol. 22, pp. 58–67, 2006.
 - [81] R. Kashef and M. S. Kamel, “Towards better outliers detection for gene expression datasets,” in *Proceedings of the 2008 International Conference on Biocomputation, Bioinformatics, and Biomedical Technologies*, pp. 149–154, 2008.
 - [82] M. D. Rasmussen, M. S. Deshpande, G. Karypis, J. Johnson, J. A. Crow, and E. F. Retzel, “wcluto : A web-enabled clustering toolkit 1,” *Plant Physiology*, vol. 133, pp. 510–516, 2003.
 - [83] M. A. T. Figueiredo, D. S. Cheng, and V. Murino, “Clustering under prior knowledge with application to image segmentation,” in *Advances in Neural Information Processing Systems 19*, MIT Press, 2007.
 - [84] K. P. Pyun, J. Lim, C. S. Won, and R. M. Gray, “Image segmentation using hidden Markov Gauss mixture models,” *IEEE Trans. on Image Processing*, vol. 16, pp. 1902–1911, JUL 2007.
 - [85] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.
 - [86] Y. Zhang, A. N. Zincir-Heywood, and E. E. Milios, “Term-based clustering and summarization of web page collections,” in *Canadian Conference on AI*, pp. 60–74, 2004.
 - [87] M. M. Shafiei, S. Wang, R. Zhang, E. E. Milios, B. Tang, J. Tougas, and R. J. Spiteri, “Document representation and dimension reduction for text clustering,” in *ICDE Workshops*, pp. 770–779, 2007.
 - [88] W. B. Cavnar, “Using an n-gram-based document representation with a vector processing retrieval model,” in *TREC*, pp. 0–, 1994.

- [89] Y. Miao, V. Kešelj, and E. Milios, “Document clustering using character n-grams: a comparative evaluation with term-based and word-based clustering,” in *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, (New York, NY, USA), pp. 357–358, ACM, 2005.
- [90] J. Koberstein and Y.-K. Ng, “Using word clusters to detect similar web documents,” *Knowledge Science, Engineering and Management*, vol. 4092, pp. 215–228, 2006.
- [91] M. R. Amini, N. Usunier, and P. Gallinari, “Automatic text summarization based on word clusters and ranking algorithms,” in *In Proceedings of the 27th European Conference on Information Retrieval*, pp. 142–156, 2005.
- [92] <http://wordnet.princeton.edu/>.
- [93] D. R. Recupero, “A new unsupervised method for document clustering by using wordnet lexical and conceptual relations,” *Inf. Retr.*, vol. 10, no. 6, pp. 563–579, 2007.
- [94] S. R. El-Beltagy, M. Hazman, and A. Rafea, “Ontology based annotation of text segments,” in *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, (New York, NY, USA), pp. 1362–1367, ACM, 2007.
- [95] M. Bernotas, K. Karkliius, R. Laurutis, and A. Slotkiene, “The peculiarities of the text document representation, using ontology and tagging-based clustering technique,” *Information Technology and Control*, vol. 36, no. 2, pp. 217–220, 2007.
- [96] S. Zhong and J. Ghosh, “A unified framework for model-based clustering,” *J. Mach. Learn. Res.*, vol. 4, pp. 1001–1037, Nov 2003.
- [97] S. Zhong and J. Ghosh, “A comparative study of generative models for document clustering,” in *SIAM Int. Conf. Data Mining Workshop on Clustering High Dimensional Data and Its Applications*, 2003.
- [98] Y. Zhao and G. Karypis, “Criterion functions for document clustering: Experiments and analysis,” tech. rep., University of Minnesota, 2002.
- [99] L. Jing, M. K. Ng, and J. Z. Huang, “An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 8, pp. 1026–1041, 2007.

- [100] E.-H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, “Webace: a web agent for document categorization and exploration,” in *AGENTS '98: Proc. of the 2nd ICAA*, pp. 408–415, 1998.
- [101] A. K. McCallum, “Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering.” <http://www.cs.cmu.edu/~mccallum/bow/>, 1996.
- [102] A. Strehl, J. Ghosh, and R. Mooney, “Impact of similarity measures on web-page clustering,” in *Proc. of the 17th National Conf. on Artif. Intell.: Workshop of Artif. Intell. for Web Search*, pp. 58–64, AAAI, July 2000.
- [103] S. Zhong and J. Ghosh, “Generative model-based document clustering: a comparative study,” *Knowl. Inf. Syst.*, vol. 8, no. 3, pp. 374–384, 2005.
- [104] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *J. R. Stat. Soc. Series B Stat. Methodol.*, vol. 39, no. 1, pp. 1–38, 1977.
- [105] S. Dasgupta and L. Schulman, “A probabilistic analysis of em for mixtures of separated, spherical gaussians,” *J. Mach. Learn. Res.*, vol. 8, pp. 203–226, 2007.
- [106] K. Rose, “Deterministic annealing for clustering, compression, classification, regression, and related optimization problems,” in *Proc. of the IEEE*, pp. 2210–2239, 1998.
- [107] N. Ueda and R. Nakano, “Deterministic annealing EM algorithm,” *Neural Netw.*, vol. 11, pp. 271–282, Mar 1998.
- [108] C. Bouveyron, S. Girard, and C. Schmid, “High-dimensional data clustering,” *Computational Statistics & Data Analysis*, vol. 52, pp. 502–519, September 2007.
- [109] C.-Y. Tsai and C.-C. Chiu, *An efficient feature selection approach for clustering: Using a Gaussian mixture model of data dissimilarity*. Springer Berlin/ Heidelberg, 2007.
- [110] S. Wang and J. Zhu, “Variable selection for model-based high-dimensional clustering and its application to microarray data,” *Biometrics*, vol. 64, pp. 440–448, JUN 2008.

- [111] M. Meila and D. Heckerman, “An experimental comparison of model-based clustering methods,” *Machine Learning*, vol. 42, pp. 9–29, 2001.
- [112] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, “Generative model-based clustering of directional data,” in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, 2003.
- [113] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra, “Clustering on the unit hypersphere using von mises-fisher distributions,” *Journal of Machine Learning Research*, vol. 6, pp. 1345–1382, 2005.
- [114] K. Mardia and P. Jupp, *Directional Statistics*. John Wiley and Sons Ltd., 2nd ed., 2000.
- [115] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Pennsylvania: Addison-Wesley, 1989.
- [116] P. J. Rousseeuw, “Multivariate estimation with high breakdown point,” *Mathematical Statistics and Applications*, 1985.
- [117] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. New York, NY, USA: John Wiley & Sons, Inc., 1987.
- [118] R. A. Maronna, “Robust M-Estimators of Multivariate Location and Scatter,” *Ann. of Statist.*, vol. 4, pp. 51–67, 1976.
- [119] P. Davies, “Asymptotic behavior of s-estimators of multivariate location parameters and dispersion matrices,” *Ann. Statist.*, vol. 15, pp. 1269–1292, 1987.
- [120] M. Hubert, P. J. Rousseeuw, and S. V. Aelst, “High-breakdown robust multivariate methods,” *Statistical Science*, vol. 23, pp. 92–119, 2008.
- [121] R. A. Maronna, D. R. Martin, and V. J. Yohai, *Robust Statistics: Theory and Methods*. New York: John Wiley and Sons, 2006.
- [122] N. Neykov and P. Neytchev, “A robust alternative of the maximum likelihood estimators,” *COMPSTAT 1990, Short Communications*, pp. 99–100, 1990.

- [123] A. Hadi, “Maximum trimmed likelihood estimators: a unified approach, examples, and algorithms,” *Computational Statistics & Data Analysis*, vol. 25, pp. 251–272, Aug. 1997.
- [124] M. Hubert and K. van Driessen, “Fast and robust discriminant analysis,” *Computational Statistics & Data Analysis*, vol. 45, no. 2, pp. 301–320, 2004.
- [125] M. Kumar and J. B. Orlin, “Scale-invariant clustering with minimum volume ellipsoids,” *Comput. Oper. Res.*, vol. 35, pp. 1017–1029, April 2008.
- [126] J. A. Cuesta-Albertos, C. Matrn, and A. Mayo-Isar, “Robust estimation in the normal mixture model based on robust clustering,” *J. R. Statist. Soc. Series B - Statistical Methodology*, vol. 70, pp. 779–802, 2008.
- [127] J. Cuesta-albertos, A. Gordaliza, and C. Matran, “Trimmed k-means: an attempt to robustify quantizers,” *Ann. Statist.*, vol. 25, pp. 553–576, 1997.
- [128] N. Neykov, P. Filzmoser, R. Dimova, and P. Neytchev, “Robust fitting of mixtures using the trimmed likelihood estimator,” *Computational Statistics & Data Analysis*, vol. 52, pp. 299–308, Sept. 2007.
- [129] N. Neykov and C. Muller, “Breakdown point and computation of trimmed likelihood estimators in generalized linear models,” *Developments in Robust Statistics*, pp. 277–286, 2003.
- [130] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.
- [131] E. E. Korkmaz, J. Du, R. Alhajj, and K. Barker, “Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering,” *Intell. Data Anal.*, vol. 10, pp. 163–182, March 2006.
- [132] K. jae Kim and H. Ahn, “A recommender system using ga k-means clustering in an online shopping market,” *Expert Syst. Appl.*, vol. 34, no. 2, pp. 1200–1209, 2008.
- [133] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. Wiley-Interscience, 2004.

- [134] R. A. Maronna and R. H. Zamar, “Robust estimates of location and dispersion for high-dimensional datasets,” *Technometrics*, vol. 44, pp. 307–317, 2002.
- [135] R. Maronna and V. Yohai, “The behavior of the stahel-donoho robust multivariate estimator,” *J. Amer. Stat. Assoc.*, vol. 90, pp. 330–341, 1995.
- [136] D. N. A. Asuncion, “UCI machine learning repository,” 2007.
- [137] C. Fraley and A. E. Raftery, “Model-based clustering, discriminant analysis, and density estimation,” *Journal of The American Statistical Association*, vol. 97, pp. 611–631, 2002.
- [138] I. Guyon, U. von Luxburg, and R. C. Williamson, “Clustering: Science or Art?,” *NIPS’09 Workshop on Clustering Theory*, 2009.
- [139] I. Dhillon and D. Modha, “Concept decompositions for large sparse text data using clustering,” *Mach. Learn.*, vol. 42, pp. 143–175, Jan 2001.
- [140] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, “Clustering with Bregman divergences,” *J. Mach. Learn. Res.*, vol. 6, pp. 1705–1749, Oct 2005.
- [141] E. Pekalska, A. Harol, R. P. W. Duin, B. Spillmann, and H. Bunke, “Non-Euclidean or non-metric measures can be informative,” in *Structural, Syntactic, and Statistical Pattern Recognition*, vol. 4109 of *LNCS*, pp. 871–880, 2006.
- [142] M. Pelillo, “What is a cluster? Perspectives from game theory,” in *Proc. of the NIPS Workshop on Clustering Theory*, 2009.
- [143] D. Lee and J. Lee, “Dynamic dissimilarity measure for support based clustering,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 6, pp. 900–905, 2010.
- [144] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, “Clustering on the unit hypersphere using von Mises-Fisher distributions,” *J. Mach. Learn. Res.*, vol. 6, pp. 1345–1382, Sep 2005.
- [145] I. S. Dhillon, S. Mallela, and D. S. Modha, “Information-theoretic co-clustering,” in *KDD*, pp. 89–98, 2003.
- [146] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. Press, Cambridge U., 2009.

- [147] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon, “Spectral relaxation for k-means clustering,” in *NIPS*, pp. 1057–1064, 2001.
- [148] Y. Gong and W. Xu, *Machine Learning for Multimedia Content Analysis*. Springer-Verlag New York, Inc., 2007.
- [149] Y. Zhao and G. Karypis, “Empirical and theoretical comparisons of selected criterion functions for document clustering,” *Mach. Learn.*, vol. 55, pp. 311–331, Jun 2004.
- [150] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proc. of the thirtieth annual ACM symposium on Theory of computing*, STOC ’98, pp. 604–613, 1998.
- [151] A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” in *Proc. of The 25th International Conference on Very Large Data Bases*, pp. 518–529, 1999.
- [152] H. Koga, T. Ishibashi, and T. Watanabe, “Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing,” *Knowledge and Information Systems*, vol. 12, pp. 25–53, May 2007.
- [153] T. H. Haveliwala, A. Gionis, and P. Indyk, “Scalable techniques for clustering the web,” in *Proc. of the Third International Workshop on the Web and Databases, WebDB 2000, in conjunction with ACM PODS/SIGMOD 2000*, pp. 129–134, 2000.
- [154] S. Vadrevu, C. H. Teo, S. Rajan, K. Punera, B. Dom, A. J. Smola, Y. Chang, and Z. Zheng, “Scalable clustering of news search results,” in *Proc. of the fourth ACM international conference on Web search and data mining*, WSDM ’11, pp. 675–684, 2011.
- [155] C. C. Aggarwal and P. S. Yu, “Redefining clustering for high-dimensional applications,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 14, pp. 210–225, March 2002.
- [156] A. Ahmad and L. Dey, “A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set,” *Pattern Recognit. Lett.*, vol. 28, no. 1, pp. 110 – 118, 2007.
- [157] D. Ienco, R. G. Pensa, and R. Meo, “Context-based distance learning for categorical data clustering,” in *Proc. of the 8th Int. Symp. IDA*, pp. 83–94, 2009.

- [158] P. Lakkaraju, S. Gauch, and M. Speretta, “Document similarity based on concept tree distance,” in *Proc. of the 19th ACM conf. on Hypertext and hypermedia*, pp. 127–132, 2008.
- [159] H. Chim and X. Deng, “Efficient phrase-based document similarity for clustering,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, no. 9, pp. 1217–1229, 2008.
- [160] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese, “Fast detection of xml structural similarity,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 2, pp. 160–175, 2005.
- [161] J. Friedman and J. Meulman, “Clustering objects on subsets of attributes,” *J. R. Stat. Soc. Series B Stat. Methodol.*, vol. 66, no. 4, pp. 815–839, 2004.
- [162] L. Hubert, P. Arabie, and J. Meulman, *Combinatorial data analysis: optimization by dynamic programming*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.
- [163] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: John Wiley & Sons, 2nd ed., 2001.
- [164] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.
- [165] J. Stefanowski and D. Weiss, “Carrot and language properties in web search results clustering,” in *AWIC*, pp. 240–249, 2003.
- [166] S. Osinski, “Dimensionality reduction techniques for search results clustering,” master thesis, Department of Computer Science, The University of Sheffield, UK, 2004.
- [167] S. Osinski and D. Weiss, “A concept-driven algorithm for clustering search results,” *IEEE Intelligent Systems*, vol. 20, no. 3, pp. 48–54, 2005.