SUBSPACE CLUSTERING METHODS FOR UNDERSTANDABLE INFORMATION ORGANIZATION

by

Juhua Hu

M.Sc., Nanjing University, 2012
B.Sc., Nanjing University, 2009

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2017

Approval

Name:	**Juhua Hu**
Degree:	**Doctor of Philosophy (Computing Science)**
Title:	***SUBSPACE CLUSTERING METHODS FOR UNDERSTANDABLE INFORMATION ORGANIZATION***
Examining Committee:	**Chair:** Dr. Jiannan Wang Assistant Professor

Dr. Jian Pei
Senior Supervisor
Professor _____

Dr. Martin Ester
Supervisor
Professor _____

Dr. Jiguo Cao
Internal Examiner
Associate Professor
Department of Statistics and Actuarial Science _____

Dr. Jingrui He
External Examiner
Assistant Professor
Computer Science and Engineering
Arizona State University _____

Date Defended : December 8, 2017

Abstract

Clustering has been widely used to identify possible structures in data and help users understand data in an unsupervised manner. Although clustering methods can provide group information for users, it is still challenging for users to efficiently and effectively understand clustering structures. Subspace clustering methods address this challenge by providing each clustering an understandable feature subspace. Moreover, as high-dimensional data has become more and more popular in real-world applications due to the advances of big data technologies, some subspace clustering methods are especially scalable for high-dimensional data.

In this thesis, we study the subspace clustering problem in different application scenarios (e.g., semi-supervised and unsupervised) to bridge the semantic gap between low-level clustering structures and high-level understandable meanings, that is, providing understandable information organization for end users. Specifically, we develop a series of novel subspace clustering methods for different application purposes. Taking image retrieval in CBIR without query input as an example, we study how subspace clustering methods can help retrieve relevant images by relevance feedback, by efficient iterative search, or by data summarization.

To tackle the challenges arising from real-world applications, we develop three efficient and effective subspace clustering methods to provide preferred or understandable clustering structures for end users. First, we present a semi-supervised subspace clustering method to discover a feature subspace, in which a preferred clustering structure is hidden. Second, we propose a subspace hierarchical clustering method that can generate a balanced hierarchy with semantics to help users search friendly and efficiently. Third, we develop a subspace multi-clustering method that can automatically discover a certain number of feature subspaces, where different clustering structures reside. Comprehensive empirical studies using synthetic and real data sets demonstrate the effectiveness of our proposed methods.

Keywords: Understandable information organization; semi-supervised subspace clustering; subspace hierarchical clustering; subspace multi-clustering

To my parents, my husband, and my son

"All models (assumptions) are wrong, but some are useful."

— attributed to George Box, Statistician (1919-2013)

Acknowledgements

First and foremost, I wish to express my deepest gratitude to my senior supervisor and mentor Dr. Jian Pei. I met him for the first time when I was a master student at Nanjing University. He opened a different but very insightful window for me. After I joined his group as a PhD student, he devoted a lot of his time to encourage and support me in different aspects: thinking, problem solving, presenting, etc. His strict but friendly guidance is exhaustive and tireless, which makes me more and more confident about my career. His relentless passion on research also influences me much beyond the study.

My gratitude also goes to my supervisor, Dr. Martin Ester, for his insightful comments and helpful suggestions. He inspired me a lot during discussions on research. I really appreciate his help to improve the quality of my research. I am very thankful to Dr. Jiguo Cao and Dr. Jingrui He for serving as examiners of my thesis, and thank Dr. Jiannan Wang for chairing my thesis defence.

I would like to express my sincere thanks to my research collaborators Dr. Qi Qian, Dr. Rong Jin, Dr. Shenghuo Zhu, and Dr. Jie Tang. I thank them for the knowledge, skills, and continuous support they imparted through collaboration. Working with them is always so enjoyable and very rewarding. A sincere thanks to Dr. Yuan Jiang and Dr. Zhi-Hua Zhou, my mentors at Nanjing University, for leading me to such a wonderful research trip on machine learning and data mining.

I would also like to thank many people in our department, support staff and faculty, for always being helpful over the years. I thank my friends at Simon Fraser University. A particular acknowledgement goes to Dr. Guanting Tang, Dr. Lingyang Chu, Xiao Meng, Xiangbo Mao, Chuancong Gao, Yu Yang, Zicun Cong, Xiaoning Xu, Yu Tao, Beier Lu, Jiaxin Liang, Lin Liu, Li Xiong, Xiaojian Wang, Zijin Zhao, Yajie Zhou, Xia Hu and Yanyan Zhang. They have made my life in Vancouver enjoyable and memorable.

Last but not least, I am very grateful to my dear family for their continuous moral support and encouragement. My particular thanks to my husband, who continuously supports and encourages me in both research and life. Every achievement of mine has his contribution. The sweetest acknowledgement goes to my son, who initialized my trip as a Mother at the time I started my PhD studying and accompanies me through my whole PhD life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Due to the advances of big data technologies, ample data (e.g., electronic health-care records or credit card transactions), which are being continuously collected, offer human beings great opportunity to discover new knowledge. However, it has become more and more difficult for humans to manually extract useful information from data. For example, it is impractical for a human being to find Amazon customers with similar behaviors in a limited time, considering that the number of active Amazon customers had reached 304 millions in 2015 [1].

Clustering [61] has been widely used to identify possible structures in data and help end users understand data in an unsupervised manner. Specifically, clustering methods aim to group a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. Taking computational genomics as an example, clustering can be used to find groups of genes with related expression patterns and such groups contain functionally related proteins, such as enzymes for a specific pathway [12].

1.1 Motivation

Apparently, clustering methods can provide useful group information of data, but also lead to a challenge to end users: how to efficiently and effectively understand clustering structures. It is due to the reason that there is a big gap between low-level clustering structures and high-level understandable meanings, that is, semantics.

Meaningful semantics are often hidden in feature subspaces, e.g., a feature subspace comprised by RGB (i.e., red, green, blue) features of marbles implying a semantic meaning of color. Subspace clustering methods [100, 123, 57] focus on discoveries of clustering structures that are hidden in feature subspaces. One major advantage of subspace clustering methods is that they can provide a corresponding feature subspace for each clustering

[1] https://www.statista.com/statistics/237810/number-of-active-amazon-customer-accounts-worldwide/

structure. Then, end users can use feature subspaces to intuitively interpret and understand clustering structures. Moreover, subspace clustering methods are in a great demand for high-dimensional data. High dimensionality has become a major challenge in this big data era. For example, more and more attributes have been collected for each object, and thus data have become sparse, that is, very few objects are similar to each other on many attributes. However, groups are observable in specific feature subspaces (e.g., subsets of features).

End users in many real-world applications can benefit from subspace clustering methods [100, 57]. In this thesis, we focus on several remaining challenges in this area. Here, we take content-based image retrieval (CBIR) as an example.

Example 1.1 (Retrieving relevant images in CBIR). CBIR [78] aims to retrieve images based on their content that refers to colors, shapes, textures, or any other information that can be derived from an image itself. CBIR is desirable because searches that rely purely on meta-data (e.g., keywords or tags) are dependent on annotation quality and completeness. Moreover, having humans manually annotate images by entering keywords or tags in a large database can be time-consuming and may not capture the keywords desired to describe an image. CBIR has a board range of applications such as crime prevention, medical diagnosis, and photograph archives [42].

In a typical CBIR setting, a user poses a query and asks the system to bring out relevant images from the database. Humans prefer searching images by keywords like *sky*, *beach*, *sunset*, and so on. However, CBIR systems do not know the relationships between keywords and image content. Therefore, most existing CBIR systems require users to translate keywords to particular colors, shapes, or textures (e.g., providing icons, user-drawn sketches, or example images) [89], so that CBIR systems can compare the query with images in database using an image similarity measure [113]. In this example, we consider three different application scenarios in a more user-friendly CBIR system, which does not require users to input or translate keyword queries.

Scenario I: Retrieving by relevance feedback. One potential way to learn a user's query is asking the user to provide relevance feedback on retrieved images. For example, a CBIR system may randomly show a batch of images (e.g., 20 images as shown in Figure 1.1) from the database at the beginning. Then, a user can simply tick the relevant images to provide feedback as shown in Figure 1.1. Thereafter, the ticked images are logged as positive samples, while the others are regarded as negative samples. A user's feedback can be used to guide the search of the next batch of images. This process can be repeated until the user stops or the search converges (i.e., no more relevant images). Finally, the CBIR system brings out a ranked list of relevant images to the user, in which the most relevant image is top-ranked based on the user's feedback.

Figure 1.1: An example of user feedback in CBIR [55]. © 2006 IEEE.

Distance metric learning (DML) [136] is one of the popular techniques that has been used to learn a user's query from his/her relevance feedback. DML aims to find a new feature subspace (i.e., a linear mapping of the original features), in which samples from the same group are closer to each other than to those from different groups. Specifically, positive samples from a user's feedback are pushed together, while negative samples are pulled apart from positive samples. Then, a set of potentially relevant images grouped together with positive samples can be obtained by clustering the database in the new feature subspace [132, 23]. Finally, the CBIR system can provide the user a ranked list of these relevant images, in which the most relevant image is the one nearest to the cluster center in the new feature subspace.

However, DML aims to learn a transformation matrix M with size of $d \times d$, where d is the total number of features in the original feature space. Learning so many parameters can easily lead to the overfitting problem, which results in bad generalization performance. In Chapter 3, we study how to alleviate the overfitting problem in DML.

Scenario II: Retrieving by iterative search. Scenario I provides a possible way to learn a user's query by his/her relevance feedback. However, it is hard to determine how many images a user need to check before he/she obtains the desired result. Many users may not be willing to check so many images one by one. Moreover, one image is usually a composition of multiple objects, each having a semantic meaning. For example, the image in Figure 1.2(a) contains *mountains*, *sea*, *trees*, and *goose*. When a user ticks it relevant in Scenario I, it is hard to tell which object(s) in the image is relevant.

(a) An image with multiple objects

(b) An example of a specific question

Figure 1.2: Exemplar images in CBIR (Images are from *scene* data [143]).

In Chapter 4, we study how subspace clustering methods can be used to learn a user's query by asking the user a sequential set of specific questions in an iterative process as

1. Initialize the whole database as candidate images;
2. Ask one specific question to the user;
3. Obtain the user's answer;
4. Generate a new candidate set of images (i.e., a subset of the previous candidate images);
5. Repeat Steps 2 to 4 until there is only one candidate image or the user stops.

Specifically, we explore the following two questions: 1) Can we make the question as specific as possible? As an instance, in each iteration, the CBIR system shows only one image and asks if a specific object in it (e.g., the goose shown in Figure 1.2(b)) is relevant. Then, the user can tick to show that it is relevant, or not otherwise; 2) Can we make the number of questions as less as possible? That is, the maximum number of objects that a user has to check is bounded.

Scenario III: Retrieving by summarizing. Scenario II can make the retrieving process more user-friendly and efficient. However, a user may need to answer many questions one by one until he/she obtains the desired image(s), although the number of questions is upper-bounded. Clustering was used to summarize a CBIR database, so that a user can quickly locate the group of images desired [26]. Specifically, when a user initiates a search, CBIR systems directly show a certain number of images as shown in Figure 1.3, where each image is a representative image of a group of similar images. Then, a user can easily tick the relevant image. If the banana image is ticked, the CBIR system may bring out a ranked list of banana images to the user.

The main challenge here is that an image can be perceived with different semantic meanings. A user may tick banana image because he/she wants all banana images. However, it is possible that the CBIR system uses this banana image to represent a group of yellow

Figure 1.3: An example of summarizing in CBIR (Images are from *fruit* data [58]).

fruits. Therefore, a retrieving failure can happen due to the disparity between a user's query concept and a CBIR system's clustering concept. In Chapter 5, we study how to automatically discover all potential query concepts in feature subspaces. Then, the CBIR system can provide representative images under each query concept (i.e., feature subspace) as shown in Figure 1.4, where RGB implies *color* and inertial implies *shape*. Thereafter, a user can check both the feature subspace and representative images to tick the relevant image without ambiguity. For example, a feature subspace comprised by only RGB features indicates a semantic meaning of *color*. Therefore, each image represents a group of fruits with the same color.

1.2 Contributions

In this dissertation, we develop three subspace clustering methods for challenges arising from those three application scenarios discussed above and help bridge the semantic gap between low-level clustering structures and high-level understandable meanings. In particular, we make the following contributions.

- **Semi-supervised subspace clustering**: Distance metric learning (DML) [136] has been widely used to find a feature subspace, in which a preferred clustering structure is hidden. However, it has been well recognized that the DML methods suffer a lot from the overfitting problem. We exploit the dropout technique, which has been successfully applied in deep learning [54] to alleviate the overfitting problem, for DML. Different from the previous studies that only apply dropout to training data, we apply dropout to both the learned metrics and the training data.

- **Subspace hierarchical clustering**: We propose to generate a balanced hierarchy with semantics embedded in each level to help end users search friendly and efficiently. We first present a simple strategy to extract all objects with semantic meanings as candidate dimensions, and then develop an efficient unsupervised feature/dimension

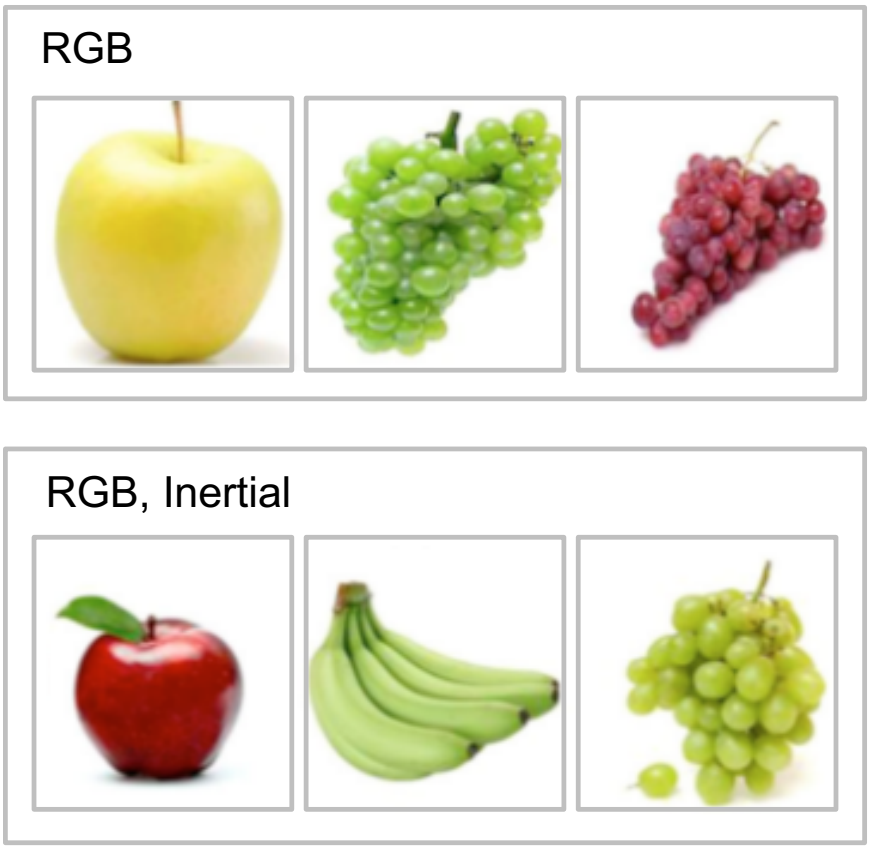


Figure 1.4: An example of summarizing in different feature subspaces in CBIR (Images are from *fruit* data [58]).

selection method to select a sufficient dimension subset to uniquely identify each image, which naturally constructs a balanced hierarchical clustering structure for images.

- **Subspace multi-clustering**: We present a novel multi-clustering method, MSC, that can automatically discover a certain number of feature subspaces, where different stable clusterings reside. An advantage of our method comparing to the state-of-the-art multi-clustering methods is that our method can provide users a feature subspace to understand each clustering solution. Another advantage is that MSC does not need users to specify the number of clusterings and their number of clusters, which is usually difficult for users without any guidance.

1.3 Organization of the Thesis

The remainder of this dissertation is structured as follows.

- In Chapter 2, we provide an overview of related work on subspace clustering methods systematically.

- In Chapter 3, we propose a regularized distance metric learning method using dropout to alleviate the overfitting problem, which is a common problem for metric learning methods. We illustrate that application of dropout to DML is essentially equivalent to matrix norm based regularization. Compared with the standard regularization scheme in DML, dropout is advantageous in simulating the structured regularizers. We verify, both empirically and theoretically, that dropout is effective in regulating the learned metric to avoid the overfitting problem.

- In Chapter 4, we develop a subspace hierarchical clustering method. To construct a balanced hierarchy as low as possible, we formulate the problem into an optimization framework that selects a minimum subset of features/dimensions with semantic meanings. We propose a greedy algorithm that naturally constructs a balanced hierarchy for searching. Our experiments on several real-world photo/image collections validate both the efficiency and effectiveness of our proposed method.

- In Chapter 5, we study two fundamental questions in subspace multi-clustering: how to model quality of clusterings and how to find multiple stable clusterings. We propose a novel method, MSC, which can heuristically determine the number of clusterings hidden in a data set and the number of clusters for each clustering. Moreover, MSC can provide each clustering a feature subspace with understandable meaning for users. We conduct an extensive empirical study that clearly demonstrates the effectiveness of our method.

- We summarize the characteristics of the proposed subspace clustering methods in Chapter 6. Some future directions are also presented.

Chapter 2

Problem Definition and Related Work

In this chapter, we first define the general problem of subspace clustering, then we provide a brief review of related work. The general subspace clustering problem discussed here includes but not limited to the well-known *subspace clustering* [123] that aims to discover multiple clusters each of which is hidden in a lower-dimensional subspace. We use *subspace clustering* to indicate this special case thereafter.

2.1 Subspace Clustering Problem

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a given data set with n objects, where each object $\mathbf{x}_i \in \mathbb{R}^d$ is represented by d attributes from the feature set $F = \{f_1, f_2, \ldots, f_d\}$. A *clustering* with k groups $C = \{c_1, c_2, \ldots, c_k\}$ is a partitioning of those objects in X such that $\cup_{i=1}^{k} c_i \subseteq X$ and $c_i \cap c_j = \emptyset$ for any $1 \leq i < j \leq k$. Each c_i $(1 \leq i \leq k)$ is called a *cluster*.

2.1.1 Subspaces

One essential challenge of subspace clustering is how to automatically discover feature subspaces which contain meaningful clustering structures. Different types of subspaces have been explored, which define a feature subspace M differently as follows. Here, M should be different from the original feature space F and $|M|$ denotes the dimensionality of feature subspace M.

- **Feature transformation** [46]: The subspace for feature transformation is defined as $M \in \mathbb{R}^{d \times d}$, where a new set of d features is generated by linear combinations of all attributes from the original feature space. Therefore, $|M| = d$. Hopefully, after transforming data X into subspace M by $M^\top X$, a hidden structure of X can be uncovered.

- **Dimensionality reduction** [109]: A subspace used to reduce dimensionality is defined as $M \in \mathbb{R}^{d \times l}$ where $l < d$ and $|M| = l$. This kind of subspaces can map the data X from the original feature space to a lower-dimensional space by $M^\top X$, from where a hidden structure of X can be discovered.

- **Feature selection** [49]: The above two kinds of subspaces both generate a new set of features, while feature selection provides subspaces that are comprised by a subset of the original feature set F. The subspace M formed by feature selection can be constructed in two different ways. The projected subspace directly uses a subset of features from F, while the weighted subspace provides each feature of F a weight to show its importance. Both feature selection strategies are special cases of feature transformation. For example, in weighted feature selection, $M \in \mathbb{R}^{d \times d}$ is formed as $M_{ii} = w_i$, where w_i is the weight assigned to the i-th feature in F, while all other elements in M are zero. We discuss these special cases in feature selection, because they essentially retain the original feature information. These two types of feature selection methods are defined as follows, respectively.

  - **Projected:** $M \subset F$, where its dimensionality $|M|$ is the number of elements in subspace M.
  - **Weighted:** $M = \{(w_i, f_i) | f_i \in F, 0 \leq w_i \leq 1, 1 \leq i \leq d, \sum_{i=1}^{d} w_i = 1\}$, where its dimensionality $|M|$ is the number of elements in subspace M with positive weights.

2.1.2 Problem Formulation

Subspace clustering methods are devoted to discovering high-quality clustering structures of X hidden in specific feature subspaces defined above. Some quality measures have been defined for a clustering $C = \{c_1, c_2, \ldots, c_k\}$, e.g., Davies-Bouldin index [35], Dunn index [40], and Silhouette coefficient [108]. We let $Q(\cdot) \in [0, 1]$ denote a clustering quality function, higher $Q(\cdot)$ indicating better quality.

Let (M, C) be a clustering C of X found in subspace M. The general problem of subspace clustering is defined as follows.

Definition 2.1.1 (Subspace Clustering)**.** *Given a data set X, generate a clustering $O = \{(M, C)\}$ of X such that $Q(C)$ is maximized.*

Many subspace clustering methods have been proposed in literature. Different approaches were presented for different application scenarios.

As an instance, in some applications, users may provide some side information (e.g., instance-level constraints like must-links or cannot-links) that can help infer the underlying instance-to-cluster assignments [132, 127]. Subspace clustering methods using side information are categorized as semi-supervised methods.

As another instance, traditional subspace clustering methods focus on finding a single way to partition data into groups. However, it has been well recognized that different outputs are possible if one uses different subsets of features in analysis. This means that assuming only a single clustering for a data set can be too strict, which has motivated the emerging area of subspace multi-clustering [57].

In the following review on related work, we categorize subspace clustering methods into two main classes, subspace single-clustering and subspace multi-clustering, respectively. Subspace single-clustering aims to find a single partitioning of X in a feature subspace, while subspace multi-clustering targets at multiple different clustering structures of X, each hidden in a feature subspace. For each category, both unsupervised and semi-supervised methods are reviewed.

2.2 Subspace Single-clustering

Subspace single-clustering is devoted to the discovery of a clustering structure hidden in an unknown feature subspace. The main challenge of subspace single-clustering is how to automatically find a specific feature subspace in which a clustering structure can be uncovered. Unsupervised methods find the subspace based only on the given data itself, while semi-supervised methods use some additional information (e.g., instance-level constraints [125]) to guide the search of the subspace.

2.2.1 Unsupervised Methods

Some unsupervised methods have been proposed in the literature. We review them with respect to different types of subspaces discovered.

Dimensionality reduction

Given X and some notion of similarity $S_{ij} \geq 0$ (e.g., RBF [65]) between all pairs of data points $\mathbf{x}_i$ and $\mathbf{x}_j$, spectral clustering [123] constructs a graph based on the similarity matrix S and tries to find a feature subspace consistent with the graph structure. Specifically, the first k eigenvectors of the graph Laplacian corresponding to the k smallest eigenvalues constitute a new data representation in a lower-dimensional feature subspace, whose dimensionality is reduced from d to k. Then, a traditional single-clustering method like k-means clustering [61] can be applied to the new data representation to obtain a clustering structure. Spectral clustering has been studied extensively. Please see [123] for a thorough treatment.

It is worth mentioning that large Laplacian eigengaps can guarantee small perturbations on the eigenvectors [117]. In Chapter 5, we extend the clustering stability based on large

eigengap in similarity learning [90] to the multi-clustering problem. We also treat the eigengap as a search criteria to determine the number of clusters [7].

Feature selection

Feature selection has also been investigated for clustering in an unsupervised manner. The primary interest is to determine which features can help to obtain better clustering results.

Law *et al.* [76] presented an expectation-maximization (EM) algorithm for unsupervised clustering with feature selection. This algorithm estimates the salience of features and the optimal number of clusters. Later, Witten and Tibshirani [130] proposed a novel framework for sparse clustering, in which a subset of features is adaptively chosen. The method uses a lasso-type penalty to select the features. Based on this framework, they developed simple algorithms for sparse k-means clustering and sparse hierarchical clustering.

At the same time, *subspace clustering* was developed to discover multiple clusters each of which is hidden in a lower-dimensional subspace (i.e., a subset of features). Some *subspace clustering* approaches aim to assign each object to a unique cluster, where clusters may exist in different subspaces. For example, PROCLUS [1] is based on the iterative processing of k-means and selects the most compact projection (subspace) based on the currently selected medoids. The projections are restricted to be the subsets of the original attributes. Later, Aggarwal and Yu [2] proposed the ORCLUS method to find arbitrarily oriented projections. DOC [101] is a Monte Carlo algorithm developed to iteratively compute projective clusters. PreDeCon [17] introduces the concept of local subspace preferences, which captures the main directions of high density. Recently, MrCC [28] adopts the multi-resolution indexing technique to extend the scalability to detect correlation clusters.

In the motivation example of scenario II (i.e., retrieving by iterative search), we also want to select a limited number of features and construct a clustering structure for search. However, the objective in our study is critically different from those in the existing work. While we will present in Chapter 4, our study focuses on finding a minimum set of features so that every image can be uniquely indexed with a combination of features.

2.2.2 Semi-supervised Methods

Semi-supervised subspace single-clustering methods use some *side information* (e.g., instance-level constraints) to guide the search of a feature subspace, in which a similar clustering structure as those constraints is observable. Apparently, the motivation problem of Scenario I (i.e., retrieving by relevance feedback) falls into this category. Most approaches proposed in this category are distance metric learning methods, whose subspaces are discovered by feature transformation.

Feature transformation

Distance metric learning has been well studied during the past years [24, 36, 112, 127, 132] and detailed investigations could be found in the survey papers [72, 136].

The early works focus on optimizing pair-wise constraints [36, 132] to make sure the distance between examples from the same class is less than a predefined threshold while that from different classes is larger than another threshold. Nowadays, triplet constraints, where the distance of examples from the same class should be marginally smaller than that of examples from different classes, are preferred due to their superior performance compared with pair-wise constraints [24, 127]. More analysis shows that triplet constraints have the large margin property and could be explained as learning a set of local SVMs [37].

However, either taking pair-wise constraints or triplet constraints increases the number of training data exponentially, which significantly results in the overfitting problem for DML. In fact, overfitting phenomenon was reported by many DML methods [24, 127], and most of them try to alleviate it by PSD constraint. PSD constraint, on one hand, is the feasible set where the optimal metric should live in, and on the other hand, restricts the learned metric in the PSD cone to reduce the complexity of the model. Given a huge number of constraints, stochastic gradient descent (SGD) is widely used to learn the metric and PSD projection occurs at every iteration [114]. Unfortunately, the computational cost of PSD projection is cubic to the dimension of data, which significantly limits the application of DML in high-dimensional datasets.

Recent empirical study [24] demonstrates that one-projection paradigm, which performs PSD projection once at the end of the algorithm, has the similar performance as projecting at every iteration. In Chapter 3, we bring dropout, a technique developed for training deep neural networks [54], to overcome overfitting in DML. We adopt triplet constraints and one-projection paradigm setting, and show that dropout significantly improves the performance of existing DML methods.

2.3 Subspace Multi-clustering

Subspace multi-clustering methods aim to discover multiple non-redundant clustering structures in different feature subspaces. In general, there are two kinds of subspace multi-clustering methods, unsupervised methods and semi-supervised methods, respectively.

2.3.1 Unsupervised Methods

Unsupervised subspace multi-clustering methods try to simultaneously find multiple clusterings that are constrained to be different from each other. We review different methods based on the types of subspaces discovered.

Dimensionality reduction

Dasgupta and Ng [32] discovered multiple subspaces utilizing different eigenvectors of the graph Laplacian. The differences between clusterings are achieved by the orthogonality between eigenvectors. They reduced the dimension from d to only 1. However, each clustering solution is restricted to have two clusters. Niu *et al.* [99] proposed a multiple spectral clustering (mSC) method. This method can simultaneously generate m (m is an input) subspaces and their corresponding cluster membership indicator matrices. Hilbert Schmidt Independence Criterion [103] is used to quantify the correlation between two subspaces, which is incorporated into the spectral clustering optimization problem. Recently, Ye *et al.* [138] adopted independent subspace analysis (ISA) [118] to find non-redundant lower-dimensional subspaces. However, the input of ISA relies on approximate solutions, and thus it limits the quality of subspaces discovered.

Feature selection

To discover multiple clusterings hidden in different subspaces (subsets of features), a brute-force way is to conduct clustering in each subspace. Due to the exponentially large number, that is $2^d - 1$, of subspaces, some researches aim to search subspaces which have potentially dense clusters in them.

CLIQUE [3] is one of the first algorithms that attempt to find subspaces based on density. It divides each dimension into fixed grid-cells by equal length intervals. Dense cells that contain more objects than a threshold η are potentially interesting clusters. It is expensive to search all dense cells in all subspaces. Based on the monotonicity that if a cell O is dense in subspace M and subspace $T \subseteq M$, O is dense in T, CLIQUE conducts a bottom-up subspace search starting from subspaces with only one dimension. The search stops when no new subspaces can be found. Then, in each subspace, connected dense cells together form a cluster and multiple clusters can be discovered in each subspace.

ENCLUS [27] follows a similar procedure as CLIQUE [3] except that it uses the entropy to select subspaces. A subspace whose entropy is below a predefined threshold is considered to have good clusters. It can be observed that the above two grid-based methods highly depend on the size of the grid cells used. The number of grid-cells determines the computational cost and the quality of the clustering results. Therefore, some methods [95, 101, 139, 111] were proposed to enhance the quality of grid cells. For example, Nagesh *et al.* [95] proposed the MAFIA method. It also has a similar procedure as CLIQUE [3]. However, it generates adaptive grid-cells based on the data distribution and does not require a user to specify the grid size.

Instead of using dense grid cells, SUBCLU [67] uses a well-known spatial clustering method DBSCAN [43] to generate clusters for each candidate subspace. Therefore, the shortcomings of grid cells can be avoided. Due to the advantage of DBSCAN, arbitrarily

shaped clusters can be discovered. However, it is highly inefficient due to repeating applying of DBSCAN in each subspace. Several techniques [71, 6, 5, 92] were proposed to speed up the step of generating candidate subspaces, which make these density-based methods scalable on high-dimensional data. At the same time, Kailing *et al.* [68] defined the *interestingness* of subspaces, where the quality of a subspace is based on the density notion of clusters. Then, only top-ranked high-quality subspaces are considered for clustering.

Most methods mentioned above in this category are *subspace clustering* methods, which aim to find multiple clusters hidden in different subspaces, although each object can be assigned to different clusters in different subspaces. Therefore, clusters formed in each subspace may not cover all data points. Moreover, many methods generate redundant clusters and the relationship between different subspaces is unknown, which is difficult to understand for users.

To overcome the specific shortcomings of *subspace clustering* methods, Günnemann *et al.* [48] assumed a generative model for given data using multiple mixture models. Each mixture describes a specific view (subspace) on the data. Recently, the cumulative mutual information (CMI) [18] method was proposed to select high contrast subspaces that potentially provide high contrast between clusters. However, users need to choose the number of clusterings and the number of clusters in each subspace.

2.3.2 Semi-supervised Methods

Semi-supervised subspace multi-clustering methods use one or more reference clusterings as the guidance to find an alternative clustering that is different from the reference clustering(s).

Feature transformation

Given a reference clustering, one straightforward way is to find an alternative clustering in a feature subspace orthogonal to the given one. Some feature transformation based methods focus on finding a transformation matrix $M \in R^{d \times d}$ that can map the data into a subspace orthogonal to the given one.

For example, Cui *et al.* [29] presented two approaches to generate an orthogonal subspace based on a given clustering. One way is to project each object onto its cluster center, and then project onto an orthogonal subspace to form a residue. The other way is to use PCA [66] to determine p ($p \leq k$) strong principle components of the cluster centers, and then calculate the orthogonal subspace. The redundancy between clustering solutions is implicitly constrained by the orthogonality between subspaces. More than one alternative clusterings can be generated sequentially. However, the orthogonality is only specified for two nearby subspaces, not between all pairs. Davidson and Qi [34] adopted the distance metric learning [136] technique to obtain an orthogonal subspace based on a given clustering.

The above two methods focus on finding an alternative subspace totally different from the given one, but cannot specify which properties of the reference clustering should or should not be retained. Qi and Davidson [102] proposed a Kullback-Leibler divergence based approach to discover an alternative subspace. In this work, a user can formally specify positive and negative feedback based on the given clustering.

It can be easily observed that all these feature transformation based methods are trying to find a full space transformation matrix that is of size $d \times d$, therefore, they are not applicable for high dimensional data due to the high time and space complexity.

Dimensionality reduction

To handle high-dimensional data, some works are devoted to discover orthogonal lower-dimensional subspaces.

Dang and Bailey [30] presented two methods in the semi-supervised scenario where there is a given (reference) clustering. The first approach, regularized PCA (RPCA), aims to discover a transformation matrix $M \in R^{d \times l}$ ($l < d$), which can map data from the original feature space into a new lower-dimensional subspace. This new subspace maximally preserves the global variance of the data and is also independent from the given clustering. They also proposed a regularized graph-based method (RegGB). Given the similarity matrix S derived from the original feature space X, RegGB learns a novel set $Y = \{\mathbf{y}_1, ...\mathbf{y}_n\}$, where each $\mathbf{y}_i \in R^l$ ($l < d$) is a new representation of $\mathbf{x}_i$. Y optimally preserves the local proximity of the objects and is independent from the given clustering.

In the motivation application of scenario III (i.e., retrieving by summarizing), we aim to automatically find multiple clustering structures, each hidden in a lower-dimensional subspace, that is, a subset of original features. Moreover, end users are not required to determine the number of clusterings or the number of clusters for each clustering. Apparently, existing subspace multi-clustering methods are not eligible for this application purpose. In Chapter 5, we propose a novel subspace multi-clustering method, MSC, which can automatically discover a certain number of clustering structures, each corresponding to a subset of features for user understanding.

2.4 Other Related Work

Besides subspace clustering methods, our work in this thesis is also well related with three other directions, namely, image annotation, multi-clustering in original full feature space, and multi-view/multi-source clustering.

2.4.1 Image Annotation

Automatic image annotation has been widely studied. Many approaches have been proposed. The earliest methods use segmentation and translation. For instance, Mori *et al.* [91] introduced a co-occurrence model that uniformly divides an image into regions with keywords and correlates each image to a set of keywords. Duygulu *et al.* [41] used a machine translation approach that translates from a vocabulary of blobs in an image to a vocabulary of words. They segmented an image into regions that are mapped to keywords.

The early image annotation approaches generally evaluate regions individually and overlook the correlation between different regions. This has been improved by some later methods. Jeon *et al.* [64] proposed the fixed annotation-based cross-media relevance model (FACMRM), which takes advantage of the joint distribution of words and blobs. The experiments show that FACMRM performs almost six times better than a word-blob co-occurrence model and two times better than a model based on machine translation.

More recently, Cao *et al.* [20] suggested to enhance the annotation performance by first finding high-confidence annotation labels for certain images and then propagating to the remaining images according to the similarity of time, location, and visual context.

Our study builds on top of some image annotation techniques. Specifically, as will be described in Chapters 4 and 5, we apply some state-of-the-art image annotation related techniques to extract features from a large set of images. However, our goal is not on image annotation. Instead, in real-world applications, more often than not, we are facing a huge image collection without any domain knowledge, let alone mentioning training annotations or image labels. Our work addresses this challenge.

2.4.2 Multi-clustering in original full feature space

In general, there are two kinds of methods for multi-clustering in original full feature space. Unsupervised methods try to simultaneously generate multiple clusterings that are constrained to be different from each other. To obtain multiple clusterings in the original full feature space, the most straightforward approaches include: (1) applying a traditional clustering algorithm multiple times with different parameter settings, (2) running different clustering algorithms, and (3) a combination of the above two strategies [21]. However, these simple approaches may generate redundant clusterings that are overwhelming for users. Therefore, meta clustering [21] further finds groups of clusterings that are similar to each other and outputs a representative clustering from each group. Jain *et. al* [62] proposed an optimization model to balance the clustering quality and the dissimilarities between clusterings, and then simultaneously generate multiple clusterings.

Semi-supervised methods use one or more reference clusterings as the guidance to find an alternative clustering that is different from the reference clustering(s). For example, COALA [8] transforms linked pairs from the reference clustering to cannot-link constraints

and then generates a good but dissimilar clustering. MAXIMUS [9] utilizes a programming model to find an alternative clustering that can maximize the dissimilarity between the new clustering and all reference clusterings.

More methods lying in the above two categories are discussed in a recent survey [10] focusing on the topic of alternative clustering. In Chapter 5, we aim to find different structures hidden in different subspaces. Apparently, our subspace multi-clustering cannot be handled by the methods limited in the original full feature space. Moreover, multi-clustering methods in original full feature space are often not applicable for high-dimensional data.

2.4.3 Multi-view/multi-source clustering

Multiple clusterings are also involved in multi-view/multi-source clustering [15, 59], but from a totally different angle. Multi-view/multi-source clustering focuses on the techniques to establish a consensus clustering by combining multiple clusterings, each from one view/source. Specifically, each object can be described in different ways using different sources, each way presenting a view on the object. For example, a web page can be represented by its text or by anchor text of inbound hyperlinks. Multiple clusterings can be obtained utilizing each view separately. In multi-view/multi-source clustering, these clusterings are assumed to be consistent to some degree and are combined to establish a consensus solution.

It has been found that multi-view/multi-source clustering can generate a better clustering than using a single view merging all available features. Bickel and Scheffer [15] considered that the available attributes can be split into two independent subsets and iterated an interleaving EM method over the two views. Kailing *et al.* [69] presented an efficient density-based approach to cluster multi-represented data from sparse or unreliable sources. Later, some researchers studied spectral clustering or fuzzy clustering with multiple views [142, 129]. Some explored the ensemble techniques on the consensus of distributed sources [63, 86] or *subspace clusterings* [44, 38]. Recently, Hua and Pei [59] studied a novel problem of mining mutual subspace clusters from multiple sources.

In summary, multi-view/multi-source clustering focuses on combining multiple similar clustering structures, while subspace multi-clustering studied in Chapter 5 is to generate multiple different clustering structures.

Chapter 3

Learning Distance Metric Using Dropout

In this chapter, we study subspace single-clustering problem in a semi-supervised setting, where users provide some side information (e.g., instance-level constraints) about their preferences on clustering structures.

3.1 Background and Overview

Learning a good distance metric is essential for distance-based clustering methods, e.g., k-means clustering [85] and spectral clustering [123]. In some applications, users may provide side information that can help infer the underlying instance-to-cluster assignments. Such knowledge has been expressed as instance-level constraints for clustering, which reveals similarity relationships among instances.

Distance metric learning (DML) [136] has been widely used to find a feature subspace, in which a preferred clustering structure is hidden. DML aims to learn a linear mapping such that in the mapped space, examples from the same class are closer to each other than those from different classes. Many methods have been developed for DML [24, 36, 84, 127, 132] in the past, and DML has been successfully applied in semi-supervised clustering [132, 23].

One problem with DML is that since the number of parameters to be determined in DML is quadratic in the dimension, it may overfit the training data and lead to a suboptimal solution [127]. Although several heuristics, such as early stopping, have been developed to alleviate the overfitting problem [127], their performance is usually sensitive to the setting of parameters (e.g. stopping criterion in early stopping), making it difficult for practitioners. Another problem with many existing DML methods is their high computational cost since they have to project intermediate solutions onto the Positive Semi-Definite (PSD) cone at every iteration to ensure that the learned metric is PSD. In [24], the authors showed that it is possible to avoid the high cost of PSD projection by an one-projection paradigm

that only needs to project the learned metric onto the PSD cone once at the end of the optimization algorithm. We adopt the one-projection paradigm in this work to alleviate the high computational cost.

Recently, dropout has been found to be helpful in alleviating the overfitting problem in training deep neural networks [54]. Dropout was initially proposed to prevent neural networks from overfitting and can be interpreted as a way of regularizing a neural network by adding noise to its hidden units [116]. The key idea is to randomly drop units (along with their connections) from a neural network during training. For example, the neural network trained in Figure 3.1(a) becomes much thinner after applying dropout in Figure 3.1(b). This

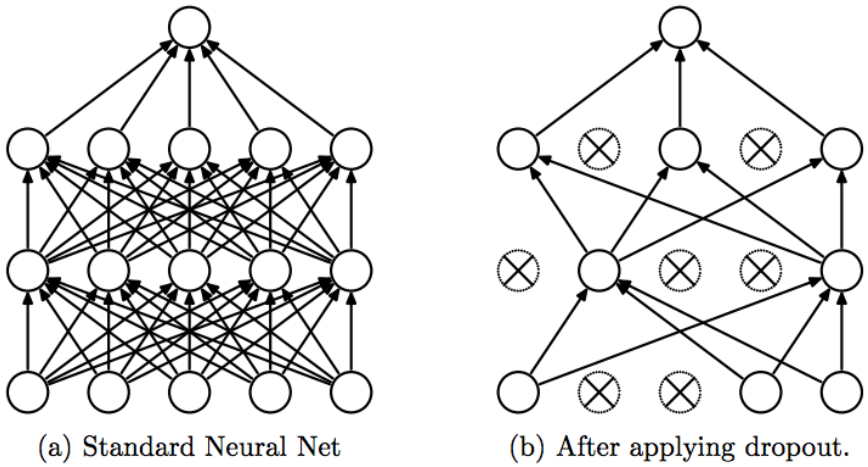


Figure 3.1: An example of dropout in neural networks [116].

prevents the units from co-adapting too much. Apparently, the number of possible thinned networks is exponential in the number of units in the network. At test time, a number of thinned networks are combined using an approximate model averaging procedure. Dropout training followed by this approximate model combination significantly reduces overfitting and gives major improvements over other regularization methods [115].

Theoretical analysis about the regularization role of dropout in neural networks can be found in [11]. Besides the success in deep learning, dropout has been applied to regression in order to obtain robust feature representations [122]. Handling features with artificial noises has been a classic topic in machine learning and data mining, and has been examined by many studies [87, 104] with the focus on additive noise that usually leads to a L_2 regularizer [16]. Wager *et al.* [124] analyzed dropout within the framework of regression and found that dropout is first-order equivalent to a L_2 regularizer for the dataset scaled by the inverse diagonal Fisher information matrix. Although dropout has received a lot of interests in machine learning and data mining community, to the best of our knowledge, this is the first study that exploits dropout for alleviating the overfitting problem in DML.

In this chapter, we, for the first time, introduce dropout to DML. Unlike previous studies on dropout that only apply dropout to training data, we apply dropout to the learned metrics. By applying appropriate dropout probabilities to the learned metric, we show that dropout can be equivalent to Frobenius norm and L_p in expectation. In addition, we develop a structured regularizer using the dropout technique. Unlike the conventional regularizer that treats diagonal and off-diagonal elements equivalently, the structured regularizer introduces different dropout probabilities for diagonal elements and off-diagonal elements.

To verify the effect of dropout in DML, we conduct a comprehensive study that compares the dropout technique to other regularization techniques used in DML. Experimental results show that dropout significantly improves the prediction performance on most datasets. In addition, we observe that dropout behaves like a trace norm based regularizer in DML when applied to training data: it controls the rank of the learned metric and leads to a skewed distribution of eigenvalues. This is in contrast to the previous studies that view dropout as a L_2 regularizer. Finally, we show that the dropout technique can be easily incorporated into the state-of-art DML methods and significantly improve their performance for most cases. The main contribution of this chapter is summarized as follows.

- We, for the first time, introduce dropout to DML and apply it to both the learned metric and the training data.
- We show that it is possible to construct structured regularizers using the dropout technique, and verify its performance, both theoretically and empirically.
- We apply the dropout to the state-of-art DML methods and show it can significantly enhance their performance.

The rest of the chapter is organized as follows. Section 3.2 defines the problem of DML. Section 3.3 describes applying dropout to the learned metric and training data, respectively. Section 3.4 summarizes the results of the empirical study. Section 3.5 concludes this chapter.

3.2 Problem Definition

In this section, we introduce the DML problem and its popular solver stochastic gradient descent.

3.2.1 Preliminaries

Given the dataset $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, distance metric learning is to learn a good Mahalanobis distance metric M, so that for each triplet constraint $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ provided, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same class and $\mathbf{x}_k$ is from a different class, we have

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_k) - \text{dist}(\mathbf{x}_i, \mathbf{x}_j) > 1$$

where $\text{dist}(\mathbf{x}_i, \mathbf{x}_j)$ is the squared Mahalanobis distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ and is measured by

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top M(\mathbf{x}_i - \mathbf{x}_j)$$

Therefore, the optimization problem based on minimizing empirical risk could be written as

$$\min_{M \in S_+^d} \sum_t \ell(\langle A_t, M \rangle)$$

where S_+^d is the $d \times d$ symmetric PSD cone, $\ell(\cdot)$ is a convex loss function, and $A_t = (\mathbf{x}_i^t - \mathbf{x}_j^t)(\mathbf{x}_i^t - \mathbf{x}_j^t)^\top - (\mathbf{x}_i^t - \mathbf{x}_k^t)(\mathbf{x}_i^t - \mathbf{x}_k^t)^\top$ is induced from the t-th triplet constraint. $\langle \cdot, \cdot \rangle$ denotes the dot product for matrix.

3.2.2 Stochastic Gradient Descent

Since the number of triplets can be very large (it can be as high as $\mathcal{O}(n^3)$), the optimization problem is usually solved by stochastic gradient descent (SGD) [36, 114]. Instead of projecting the learned metric onto PSD cone at every iteration, which can be an expensive operation, we adopt one-projection paradigm [24], which only projects the learned metric onto the PSD cone once at the end of iterations. Empirical studies have shown that one-projection-paradigm is significantly more efficient and yields the similar performance as SGD with PSD projection performed at every iteration. Therefore, in this work, we will focus on the following optimization problem without the PSD constraint.

$$\min_{M \in S^d} \sum_t \ell(\langle A_t, M \rangle) \tag{3.1}$$

Algorithm 1 gives the standard SGD algorithm for DML and dropout will be applied to perturb Step 5. We will discuss the details in the next section within this framework.

Algorithm 1 SGD for DML

1: **Input:** Dataset $X \in \mathbb{R}^{d \times n}$, #Iterations T, Stepsize η
2: Initial M_0 as an identity matrix
3: **for** $t = 1, \cdots, T$ **do**
4: Randomly sample a triplet $\{\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k\}$
5: $\boxed{M_t = M_{t-1} - \eta \nabla \ell}$
6: **end for**
7: **return** $\Pi_{psd}(\bar{M})$

3.3 The Proposed Method

In this section, we will discuss two different applications of dropout, that is, application of dropout to the learned metric and application of dropout to the training data.

3.3.1 Applying Dropout to Distance Metric

In this section, we focus on applying dropout to the learned metric. Let M be the metric learned from the previous iteration. To apply the dropout technique, we introduce a Bernoulli random matrix $\delta = [\delta_{i,j}]_{i,j=1}^{d}$, where each $\delta_{i,j}$ is a Bernoulli random variable with $\delta_{i,j} = \delta_{j,i}$. Using the random matrix δ, we compute the dropped out distance metric, denoted by $\hat{M}$ as

$$\hat{M}_{i,j} = \delta_{i,j} M_{i,j}, \quad i,j = 1,\ldots,d$$

Note that by enforcing $\delta_{i,j} = \delta_{j,i}$, $\hat{M}$ is ensured to be a symmetric matrix. In the following, we will discuss how to design the dropout probabilities for the Bernoulli random matrix δ to simulate the effect of Frobenius norm based regularization and L_1 norm based regularization, respectively.

Frobenius norm

Frobenius norm is the most widely used regularizer in DML [114, 127], and the standard DML problem with Frobenius norm is given by

$$\min_{M \in S^d} \frac{1}{T} \sum_{t=1}^{T} \ell(\langle A_t, M \rangle) + \frac{q}{2\eta} \|M\|_F^2 \tag{3.2}$$

The updating rule in SGD for Frobenius norm based regularization is

$$M_t = M_{t-1} - qM_{t-1} - \eta \nabla \ell_t(M_{t-1})$$

where $\ell_t(M) = \ell(\langle A_t, M \rangle)$.

Instead of using the regularizer directly, we could simulate the effect of Frobenius norm based regularization by applying dropout to the learned metric M_{t-1}. In particular, the Bernoulli random matrix δ is constructed by sampling each $\delta_{i,j:i\leq j}$ independently from a Bernoulli distribution with $\Pr[\delta = 0] = q$ and setting $\delta_{j,i} = \delta_{i,j}$ to ensure that δ is symmetric. It is easy to verify that

$$E[\hat{M}_{t-1}] = (1-q)M_{t-1}$$

and the updating rule becomes

$$M_t = \hat{M}_{t-1} - \eta \nabla \ell_t(M_{t-1})$$

Theorem 3.3.1. *Let M_* be the optimal solution output by Algorithm 1. Let $\bar{M}$ be the solution output by Algorithm 1 with dropout in Step 5 and q be the probability that dropout occurs in each item of the learned metric. Assume $|\mathbf{x}|_2 \leq r$ and $q = 1/T$, we have*

$$E[\ell(\bar{M})] - \ell(M_*) \leq \frac{1}{\eta T} \|M_*\|_F^2 + 8\eta r^2 (1 + \frac{1}{T})$$

Proof.

$$\begin{aligned}
&\|M_t - M_*\|_F^2 = \|\hat{M}_{t-1} - \eta A_t - M_*\|_F^2 \\
&= \|\delta M_{t-1} - (1-q)M_{t-1} + M_{t-1} - \eta A_t - M_* - qM_{t-1}\|_F^2 \\
&= (\delta - (1-q))^2\|M_{t-1}\|_F^2 + \|M_{t-1} - M_*\|_F^2 + \eta^2\|A_t\|_F^2 \\
&+q^2\|M_{t-1}\|_F^2 - 2\eta\langle A_t, M_{t-1} - M_*\rangle \\
&+2(\delta - (1-q))\langle M_{t-1}, M_{t-1} - \eta A_t - M_* - qM_{t-1}\rangle \\
&-2q\langle M_{t-1}, M_{t-1} - \eta A_t - M_*\rangle
\end{aligned}$$

Since the loss function is convex, we have

$$\begin{aligned}
&\ell(M_{t-1}) - \ell(M_*) \leq \frac{1}{2\eta}(\|M_{t-1} - M_*\|_F^2 - \|M_t - M_*\|_F^2) \\
&+\frac{\eta}{2}\|A_t\|_F^2 + \frac{1}{2\eta}\Big((\delta - (1-q))^2\|M_{t-1}\|_F^2 + q^2\|M_{t-1}\|_F^2 \\
&+2(\delta - (1-q))\langle M_{t-1}, M_{t-1} - \eta A_t - M_* - qM_{t-1}\rangle \\
&-2q\langle M_{t-1}, M_{t-1} - \eta A_t - M_*\rangle)
\end{aligned}$$

Taking expectation on δ, we have

$$\begin{aligned}
&E[\ell(M_{t-1})] - \ell(M_*) \leq \frac{1}{2\eta}(\|M_{t-1} - M_*\|_F^2 - \|M_t - M_*\|_F^2) \\
&+\frac{\eta}{2}\|A_t\|_F^2 + \frac{1}{2\eta}(q\|M_{t-1}\|_F^2 - 2q\langle M_{t-1}, M_{t-1} - \eta A_t - M_*\rangle) \\
&\leq \frac{1}{2\eta}(\|M_{t-1} - M_*\|_F^2 - \|M_t - M_*\|_F^2) + \frac{\eta}{2}\|A_t\|_F^2 \\
&+\frac{q}{2\eta}(2\langle M_{t-1}, \eta A_t + M_*\rangle - \|M_{t-1}\|_F^2) \\
&\leq \frac{1}{2\eta}(\|M_{t-1} - M_*\|_F^2 - \|M_t - M_*\|_F^2) + \frac{\eta}{2}\|A_t\|_F^2 \\
&+\frac{q}{2\eta}(\|\eta A_t + M_*\|_F^2) \\
&\leq \frac{1}{2\eta}(\|M_{t-1} - M_*\|_F^2 - \|M_t - M_*\|_F^2) \\
&+\frac{(1+q)\eta}{2}\|A_t\|_F^2 + \frac{q}{2\eta}\|M_*\|_F^2 + q\ell(M_*)
\end{aligned}$$

Since $|\mathbf{x}|_2 \leq r$, $\|A_t\|_F \leq 4r$. Adding iterations from 1 to T and setting $q = 1/T$, we have

$$E[\ell(\bar{M})] - (1 - 1/T)\ell(M_*) \leq \frac{1}{\eta T}\|M_*\|_F^2 + 8r^2(1 + 1/T)\eta$$

□

By setting the stepsize η as

$$\eta = \frac{\|M_*\|_F}{r\sqrt{8+8T}}$$

we have

$$E[\ell(\bar{M})] - \ell(M_*) \leq \frac{4r\sqrt{2+2T}}{T}\|M_*\|_F = \mathcal{O}(1/\sqrt{T})$$

It is well known that $\mathcal{O}(1/\sqrt{T})$ is the minimax convergence rate for SGD, when the loss function is Lipschitz continuous [146, 52, 53]. As a result, with appropriate choice of dropout probabilities, dropout will maintain the same convergence rate as the standard SGD method. We also notice that q is suggested to be set as $1/T$ in order to achieve $O(1/\sqrt{T})$ convergence. This result implies that dropout should not be taken too frequently, which is consistent with the analysis of other corrupted feature methods [16, 122]. Finally, since the derivation of convergence rates keep the same regardless of the sampling probabilities used in dropout, the convergence analysis for the following cases is almost identical as the above analysis. Therefore, the following cases will maintain the same convergence rate as the standard SGD method.

L_1 norm

Besides Frobenius norm, L_1 norm also could be used in DML as

$$\min_{M \in S^d} \frac{1}{T}\sum_t \ell(\langle A_t, M\rangle) + \frac{q}{\eta}\|M\|_1$$

It is known as the composite optimization problem [97] and could be solved by iterative thresholding method

$$\begin{cases} M'_t = M_{t-1} - \eta\nabla\ell_t(M_{t-1}) \\ M_{t:i,j} = \text{sign}(M'_{t:i,j})\max\{0, |M'_{t:i,j}| - q\} \end{cases}$$

With different design of sampling probabilities, we can apply dropout to the learned metric to simulate the effect of L_1 regularization. In particular, we introduce a data-dependent dropout probability as

$$\Pr[\delta_{i,j} = 0] = \min\{1, \frac{q}{|M_{i,j}|}\}$$

Now, instead of perturbing M_{t-1}, we apply dropout to M'_t, that is, the matrix after the gradient mapping. It is easy to verify that the expectation of the perturbed matrix $\hat{M'}$ is given by

$$E\left[[\hat{M'}_t]_{i,j}\right] = \begin{cases} [M'_t]_{i,j} - \text{sign}([M'_t]_{i,j})q & : \quad q \leq |[M'_t]_{i,j}| \\ 0 & : \quad q > |[M'_t]_{i,j}| \end{cases}$$

which is equivalent to the thresholding method stated above.

It is straightforward to extend the method to L_p norm

$$L_p(M) = (\sum_{i,j} |M_{i,j}|^p)^{1/p}$$

by setting the probability as

$$Pr[\delta_{i,j} = 0] = \min\{1, \frac{q|M_{i,j}|^{p-2}}{(\sum_{i,j} M_{i,j}^p)^{1-1/p}}\}$$

Note that when $p = 1$, it is equivalent to the probability for L_1 norm.

Structured regularizer

Although these conventional regularizers have been applied for DML, the performance could not be guaranteed. Considering the structure of a metric, the diagonal elements are more important than those from off diagonal. It is due to the fact that diagonal elements represent the importance of each feature rather than the interactions between different features, and they also control the trace of the learned metric. Therefore, the regularizer should be assigned for diagonal and off-diagonal elements differently. Fortunately, dropout can serve this purpose conveniently.

Given Q, which is a random matrix with each element from a uniform distribution in $[0, 1]$, we investigate the matrix

$$R = (Q + Q^\top)/2 \tag{3.3}$$

It is obvious that the diagonal elements of R are still from the same uniform distribution, while elements in off diagonal are from a triangle distribution with cumulative distribution function as

$$F(q) = \begin{cases} 2q^2 & : \quad 0 \leq q < 0.5 \\ 1 - 2(1-q)^2 & : \quad 0.5 \leq q \leq 1 \end{cases}$$

Figure 3.2 illustrates the cumulative distribution function for the diagonal elements of R and those living in off diagonal. The dropout probability based on the random matrix R is defined as

$$Pr[\delta_{i,j} = 0] = Pr[R_{i,j} \leq q]$$

First, we consider dropout with the same probability for each item of the metric as for Frobenius norm. Then, the probability of δ is

$$Pr[\delta_{i,j} = 0] = Pr[R_{i,j} \leq q] = \begin{cases} q & : \quad i = j \\ 2q^2 & : \quad i \neq j \end{cases} \tag{3.4}$$

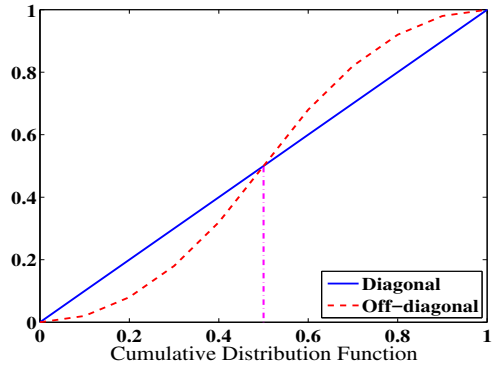


Figure 3.2: Cumulative distribution function for diagonal elements and those in off diagonal.

Algorithm 2 Dropout as Structured Frobenius Norm (SGD-M1)

Input: Dataset $X \in \mathbb{R}^{d \times n}$, #Iterations T, Stepsize η
Initial M_0 as an identity matrix
for $t = 1$ **to** T **do**
  Randomly sample a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$
  Generate random matrix R as in Equation 3.3
  Generate dropout parameters δ by Equation 3.4
  Dropout: $\hat{M}_{t-1} = \delta M_{t-1}$
  $M_t = \hat{M}_{t-1} - \eta \nabla \ell$
end for
return $\Pi_{psd}(\bar{M})$

since $q = 1/T \ll 0.5$ as indicated in Theorem 3.3.1.

Therefore, the expectation of $\hat{M}_{t-1}$ is

$$\hat{M}_{t-1} = \begin{cases} M_{i,j}^{t-1} - q M_{i,j}^{t-1} & : \quad i = j \\ M_{i,j}^{t-1} - 2q^2 M_{i,j}^{t-1} & : \quad i \neq j \end{cases}$$

which is equivalent to solving the following problem

$$\min_{M \in S^d} \frac{1}{T} \sum_t \ell(\langle A_t, M \rangle) + \frac{q}{2\eta} \sum_i^d M_{i,i}^2 + \frac{q^2}{\eta} \sum_{i,j:i \neq j} M_{i,j}^2$$

It is obvious that the L_2 norm of diagonal elements in the metric is penalized quadratically more than those from off diagonal. This regularizer seems complex but the implementation by dropout is quite straightforward and Algorithm 2 summarizes the method.

Algorithm 3 Dropout as Structured L_1 Norm (SGD-M2)

Input: Dataset $X \in \mathbb{R}^{d\times n}$, #Iterations T, Stepsize η
Initial M_0 as an identity matrix
for $t = 1$ **to** T **do**
    Randomly sample a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$
    $M_t' = M_{t-1} - \eta\nabla\ell$
    Generate random matrix R as in Equation 3.3
    Generate dropout parameters δ by Equation 3.5
    Dropout: $M_t = \delta M_t'$
end for
return $\Pi_{psd}(\bar{M})$

Then, we consider dropout with the probability based on the elements as

$$Pr[\delta_{i,j} = 0] = Pr[R_{i,j} \leq \min\{1, \frac{q}{|M_{i,j}|}\}] \tag{3.5}$$
$$= \begin{cases} \min\{1, q/|M_{i,j}|\} & : \quad i = j \\ 2(q/|M_{i,j}|)^2 & : \quad i \neq j, q < 0.5|M_{i,j}| \\ 1 - 2(1 - q/|M_{i,j}|)^2 & : \quad i \neq j, 0.5 \leq q/|M_{i,j}| \leq 1 \\ 1 & : \quad q > |M_{i,j}| \end{cases}$$

It seems too complicated to analyze at the first glance, but Figure 3.2 could help us to understand the dropout strategy here. For the diagonal elements, they are actually shrunk by q as the L_1 regularizer. For the off-diagonal elements, if $|M_{i,j}| > 2q$, the red dashed curve is under the blue solid one, which means the shrinkage is less than q. When $q \leq |M_{i,j}| \leq 2q$, the red dashed curve stands above the blue solid one and the shrinkage on these elements is much faster than the standard L_1 norm. Since q is very small, most of the off-diagonal elements have relatively larger values and will be shrunk slower than those with extremely small values. Algorithm 3 summarizes this method. Unlike Algorithm 2, dropout in Algorithm 3 is performed after updating with gradient.

3.3.2 Applying Dropout to Training Data

Besides dropout within the learned metric, in this section we apply dropout to the training data as many pervious studies [122, 124]. Since the analysis for data highly dependents on the loss function, we take the hinge loss, which is the most widely used loss function in DML [24, 36, 114, 127], as an example.

Hinge loss is defined as $\ell(z) = [1 + z]_+$, where $z = \langle A, M\rangle$ and

$$A = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)\top - (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)\top$$

Apparently, the loss function penalizes A rather than an individual example, so dropout is taken according to the structure of A. To avoid affecting the decision of hinge loss, we perturb A after calculating the hinge loss.

We begin with additive noise as

$$\hat{A} = (\mathbf{x}_i - \mathbf{x}_j + \epsilon)(\mathbf{x}_i - \mathbf{x}_j + \epsilon)\top - (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)\top \tag{3.6}$$

where $\epsilon \sim \mathcal{N}(0, qI_{d\times d})$. So the expectation of $\hat{A}$ is

$$\begin{aligned} E[\hat{A}] &= E[(\mathbf{x}_i - \mathbf{x}_j + \epsilon)(\mathbf{x}_i - \mathbf{x}_j + \epsilon)\top] - (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)\top \\ &= A + qI \end{aligned}$$

By replacing A in Problem 3.1 with $\hat{A}$, the expectation of the problem becomes

$$\min_{M\in S^d} \sum_t \ell(\langle A_t, M\rangle) + \sum_{t:\ell_t>0} q\|M\|_{tr} \tag{3.7}$$

Note that the trace norm stands outside of the hinge loss, since the noise is added only after computing the hinge loss and only active constraints will contribute to the trace norm. We use the trace norm rather than the trace of the metric, because the final metric will be projected onto the PSD cone, where the trace of metric is equivalent to the trace norm. Algorithm 4 describes the details of the method.

Although the Guassian noise could perform as the trace norm, the external noise may affect the solution. Therefore, we consider dropout instead by

$$\hat{\mathbf{x}_i}^s = \mathbf{x}_i^s * \delta_s, \quad \hat{\mathbf{x}_j}^s = \mathbf{x}_j^s * \delta_s$$

where δ_s is a binary value random variable and

$$Pr[\delta_s = 1 + q/(\mathbf{x}_i^s - \mathbf{x}_j^s)^2] = 1/(1 + q/(\mathbf{x}_i^s - \mathbf{x}_j^s)^2)$$

It is obvious that $E[\delta_s] = 1$ and $V[\delta_s] = 1 + q/(\mathbf{x}_i^s - \mathbf{x}_j^s)^2$. Similar to the additive noise, we only apply dropout for the first item of A as

$$\hat{A} = (\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)\top - (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)\top \tag{3.8}$$

Note that when we perform dropout to the training data according to this strategy, we actually drop the rows and the corresponding columns in the first component $(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)\top$ of A. Since the expectation of random variables in diagonal is the variance and it is

Algorithm 4 Additive Noise as Trace Norm (SGD-D1)

Input: Dataset $X \in \mathbb{R}^{d\times n}$, #Iterations T, Stepsize η
Initial M_0 as an identity matrix
for $t = 1, \cdots, T$ **do**
  Randomly sample a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$
  if $\ell(A_t, M_{t-1}) > 0$ **then**
    Generate a Guassian noise vector ϵ
    Add noise as in Equation 3.6
    $M_t = M_{t-1} - \eta \hat{A}_t$
  end if
end for
return $\Pi_{psd}(\bar{M})$

Algorithm 5 Dropout as Trace Norm (SGD-D2)

Input: Dataset $X \in \mathbb{R}^{d\times n}$, #Iterations T, Stepsize η
Initial M_0 as an identity matrix
for $t = 1, \cdots, T$ **do**
  Randomly sample a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$
  if $\ell(A_t, M_{t-1}) > 0$ **then**
    Dropout as in Equation 3.8
    $M_t = M_{t-1} - \eta \hat{A}_t$
  end if
end for
return $\Pi_{psd}(\bar{M})$

1 in off diagonal, the expectation of $\hat{A}$ is

$$\begin{aligned} E[\hat{A}] &= E[(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j)\top] - (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)\top \\ &= A + qI \end{aligned}$$

By taking $\hat{A}$ back to Problem 3.1, we obtain the same problem as Problem 3.7. Algorithm 5 summarizes this dropout strategy for training data.

Theorem 3.3.2. *Let M_* be the optimal solution output by Algorithm 1. Let $\bar{M}$ be the solution output by Algorithm 5 and q be the probability that dropout occurs in each feature of the dataset. Assume $|\mathbf{x}|_2 \leq r$ and $q = 1/T$, we have*

$$E[\ell(\bar{M})] - \ell(M_*) \leq \frac{\|M_*\|_F^2}{2\eta T} + 8\eta r^2 + \frac{1}{T}(8\eta d r^2 + 4\eta r^2 + \|M_*\|_{tr})$$

Proof.

$$\begin{aligned} &\|M_t - M_*\|_F^2 = \|M_{t-1} - \eta\hat{A} - M_*\|_F^2 \\ &= \|M_{t-1} - M_*\|_F^2 + \eta^2\|\hat{A}\|_F^2 - 2\eta\langle\hat{A}, M_{t-1} - M_*\rangle \end{aligned}$$

Taking expectation on $\hat{A}$, we have

$$\begin{aligned}
&\|M_t - M_*\|_F^2 \\
&= \|M_{t-1} - M_*\|_F^2 + \eta^2 E[\|\hat{A}\|_F^2] - 2\eta\langle A + qI, M_{t-1} - M_*\rangle
\end{aligned}$$

Since the loss function is convex, we also have

$$\begin{aligned}
&E[\ell(M_{t-1})] - \ell(M_*) \leq \frac{1}{2\eta}(\|M_{t-1} - M_*\|_F^2 - \|M_t - M_*\|_F^2) \\
&+\frac{\eta}{2}E[\|\hat{A}\|_F^2] + q(tr(M_*) - tr(M_{t-1})) \\
&\leq \frac{1}{2\eta}(\|M_{t-1} - M_*\|_F^2 - \|M_t - M_*\|_F^2) \\
&+\frac{\eta r^2}{2}(16 + 16dq + 8q) + q\|M_*\|_{tr}
\end{aligned}$$

where q's high-order items are omitted since q is a small number. After a summation over the iteration from 1 to T, we have

$$E[\ell(\bar{M})] - \ell(M_*) \leq \frac{\|M_*\|_F^2}{2\eta T} + \eta r^2(8 + 8dq + 4q) + q\|M_*\|_{tr}$$

The proof is finished by setting $q = 1/T$. □

If we set the stepsize η as

$$\eta = \frac{\|M_*\|_F}{2r\sqrt{4T + 4d + 2}}$$

we have

$$\ell(\bar{M}) - \ell(M_*) \leq \frac{1}{T}\left(2r\sqrt{(4T + 4d + 2)}\|M_*\|_F + \|M_*\|_{tr}\right)$$

where $\mathcal{O}(1/\sqrt{T})$ convergence rate, the well known result for standard SGD, is also observed as in Theorem 3.3.1.

According to Theorem 3.3.2, applying dropout to training data with appropriate component and dropout probability does not hurt the convergence performance of standard SGD method either. Furthermore, q is required to be sufficiently small to avoid the suboptimal solution, which is also consistent with the analysis in Theorem 3.3.1.

3.4 Experimental Results

Although we aim to learn a good distance metric for clustering, we verify our proposed DML method using the standard experimental setting for DML. Specifically, six datasets from different application scenarios are used to verify the effectiveness of the proposed method.

Table 3.1: Statistics for the datasets used in the empirical study. #C is the number of classes. #F is the number of features. #Train and #Test represent the number of training data and test data, respectively.

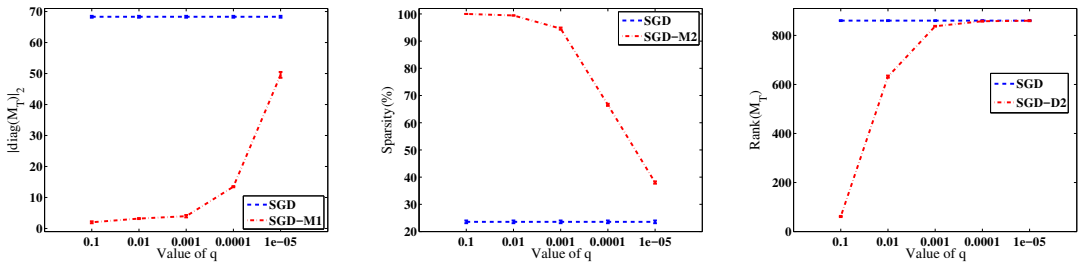
	# C	# F	#Train	#Test
ta	6	106	902	391
semeion	10	256	1,115	478
dna	3	180	2,000	1,186
caltech10	10	1,000	3,151	1,338
protein	3	357	17,766	6,621
sensit	3	100	78,823	19,705

Table 3.1 summarizes the information of these datasets. *ta* is a social network dataset with 6 different categories of terrorist attacks [110]. *semeion* is a handwritten digit dataset downloaded directly from the UCI repository [83]. *caltech10* is a subset of Caltech256 image dataset [47] with 10 most popular categories and we use the version pre-processed by a previous study [24], where each image is represented by a $1,000$-dimensional vector. The other datasets are directly downloaded from LIBSVM database [22]. For *dna*, *protein* and *sensit*, we use the standard training/testing split provided by the original dataset. For the rest datasets, we randomly select 70% of data for training and use the remaining 30% for testing. For each dataset, we randomly select $T = 100,000$ active triplets (e.g., incur the positive hinge loss by Euclidean distance) within the range of 3-nearest same class neighbors as suggested by the study [127]. Instead of using clustering algorithms to cluster the whole data set, k-nearest neighbor (k=3) classifier is more meaningful in this scenario and is applied after obtaining the metric, since we optimize the triplets from 3-nearest neighbors. All experiments are repeated by 5 trials on different randomly generated triplet sets and the average result with standard deviation is reported.

3.4.1 Comparison with SGD Methods

In the first experiment, we compare the standard SGD for DML to five SGD variants including our proposed methods (i.e., SGD-M1, SGD-M2, SGD-D1, and SGD-D2). The methods are summarized as follows.

- **SGD**: stochastic gradient descent method as described in Algorithm 1.
- **SGD-PSD**: SGD with PSD projection at every iteration.
- **SGD-M1**: SGD with dropout for learned metric as structured Frobenius norm (Algorithm 2).
- **SGD-M2**: SGD with dropout for learned metric as structured L_1 norm (Algorithm 3).
- **SGD-D1**: SGD with additive Guassian noise in training data as trace norm (Algorithm 4).
- **SGD-D2**: SGD with dropout for training data as trace norm (Algorithm 5).



(a) *SGD-M1* (b) *SGD-M2* (c) *SGD-D2*

Figure 3.3: Trend of effect for dropout as regularizers on dataset *caltech*. Fig.(a) is the L_2 norm of the diagonal elements in the metric learned by SGD-M1. Fig.(b) is the sparsity of the metric learned by SGD-M2. Fig.(c) is the rank of the metric learned by SGD-D2.

Euclidean distance is also included as a baseline method and denoted as "**Euclid**".

All of these SGD methods are applied on the same triplet set and take one-projection paradigm except SGD-PSD, which projects the learned metric onto the PSD cone at every iteration. We search the stepsize η in $\{0.1, 1, 10\}$ by cross validation and $\eta = 1$ shows the best performance, so we fix it for all experiments. The dropout probability parameter q for the proposed methods is searched in $\{10^{-i} : i = 1, \cdots, 5\}$. All SGD methods are started with an identity matrix in the experiment.

Table 3.2 shows the classification accuracy of different SGD methods. First, it is not surprising to observe that all the DML algorithms improve the performance compared to the Euclidean distance. Second, for all datasets, we observe that the proposed SGD methods with dropout (i.e., SGD-M1, SGD-M2, SGD-D1, and SGD-D2) significantly outperform the baseline SGD methods (i.e., SGD and SGD-PSD), which is also demonstrated by the statistical significance examined via pairwise t-tests at the 5% significance level. Concretely, on most datasets, the accuracy of SGD with dropout is about 2% improved compared with that of SGD and it is even 4% on *protein*.

Furthermore, we observe that SGD-M1 shows the best performance on *semeion*, *caltech10*, and *protein*, while SGD-M2 outperforms other methods on *ta* and *dna*, and SGD-D2 is the best on *sensit*. It is because dropout in the learned metric and dropout in the training data represent different regularizers, and different dataset prefers different regularizer. SGD-D1 and SGD-D2 have the similar performance because they optimize the same trace norm. However, SGD-D2 is a little bit better than SGD-D1 due to the reason that no additional Guassian noise is introduced by SGD-D2. Finally, SGD-PSD performs same as if not worse than SGD, which is consistent with the observation in a previous study [24].

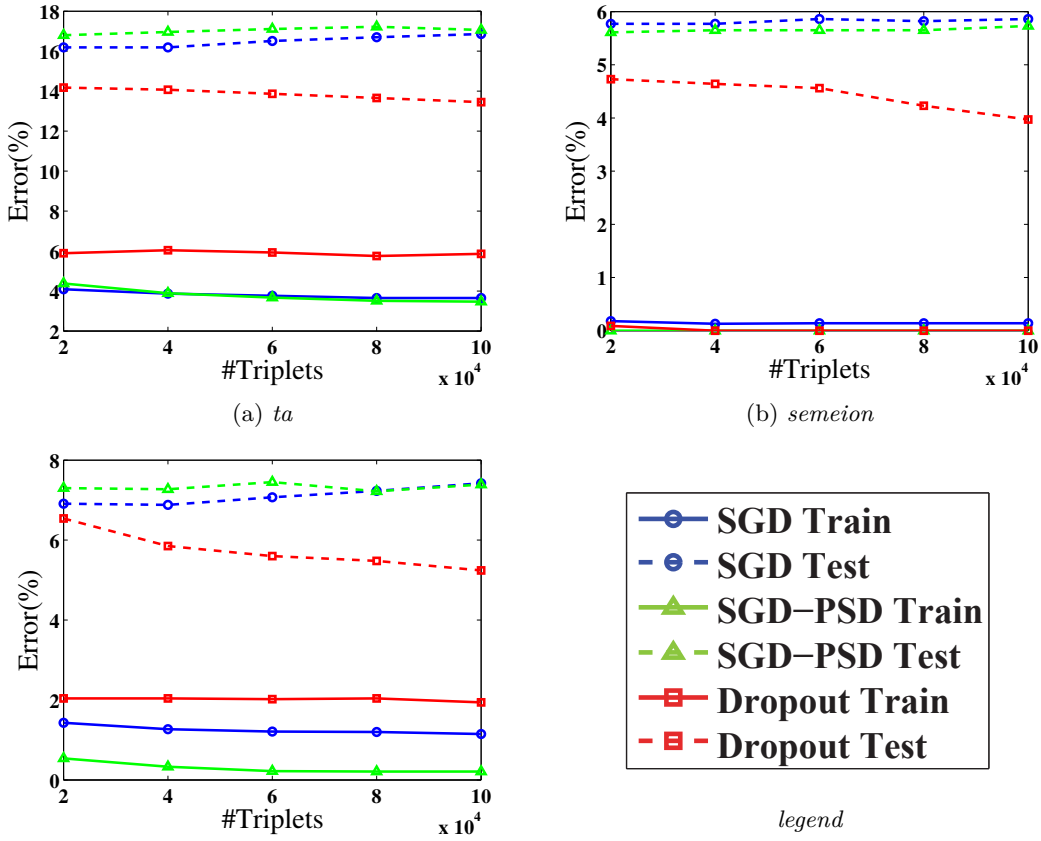
Then, we investigate if dropout can perform as the regularizers as we expected. Figure 3.3 compares the effect of different norms with different parameters to that of SGD, where the parameter q of dropout changes and the others are kept the same. First, since SGD-M1 puts more aggressive penalty on the diagonal, Figure 3.3(a) shows how L_2 norm

Table 3.2: Comparison of classification accuracy (%) for different SGD methods, and the best result is bolded (statistical significance examined via pairwise t-tests at the 5% significance level between baselines and the proposed methods).

	ta	*semeion*	*dna*	*caltech10*	*protein*	*sensit*
Euclid	80.15	91.63	80.10	62.36	50.05	72.72
SGD	83.14±0.39	94.14±0.36	92.58±0.39	65.29±0.29	60.19±0.16	74.92±0.09
SGD-PSD	82.94±0.56	94.27±0.11	92.61±0.35	64.88±0.39	58.15±0.47	74.54±0.08
SGD-M1	85.77±0.42	**96.03±0.39**	93.86±0.33	**67.32±0.29**	**64.05±0.31**	76.33±0.10
SGD-M2	**86.55±0.22**	95.61±0.42	**94.76±0.59**	66.04±0.24	62.76±0.47	76.10±0.08
SGD-D1	84.33±0.69	95.31±0.55	93.36±0.41	65.85±0.46	61.36±0.57	76.82±0.05
SGD-D2	84.74±0.50	95.40±0.39	93.66±0.39	66.07±0.37	62.86±0.46	**76.89±0.07**
SPML	83.56±0.50	94.60±0.27	92.85±0.37	65.46±0.50	59.15±0.37	74.99±0.10
OASIS	83.20±0.95	94.06±0.19	88.57±0.28	65.06±0.40	58.83±0.38	73.50±0.15
LMNN	84.79±0.65	93.77±0.48	**95.13±0.26**	**67.17±0.49**	60.73±0.12	76.47±0.04

Table 3.3: Comparison of classification accuracy (%) with state-of-art DML methods. "Dropout" refers to the best result of dropout from Table 3.2. Note that LMNN is a batch learning algorithm, and there is no limitation for the triplets it uses and the number of PSD projections. The best result is bolded (statistical significance examined via pairwise t-tests at the 5% significance level).

	ta	*semeion*	*dna*	*caltech10*	*protein*	*sensit*
SPML	83.56±0.50	94.60±0.27	92.85±0.37	65.46±0.50	59.15±0.37	74.99±0.10
OASIS	83.20±0.95	94.06±0.19	88.57±0.28	65.06±0.40	58.83±0.38	73.50±0.15
LMNN	84.79±0.65	93.77±0.48	**95.13±0.26**	**67.17±0.49**	60.73±0.12	76.47±0.04
Dropout	**86.55±0.22**	**96.03±0.39**	**94.76±0.59**	**67.32±0.29**	**64.05±0.31**	**76.89±0.07**



(a) *ta* (b) *semeion*

(c) *dna* *legend*

Figure 3.4: Comparison of training error and test error of different SGD methods with different size of triplets. There is no overfitting observed for SGD method with dropout.

of the diagonal elements in the metric learned by SGD-M1 varies as q changes. We observe that the looser the parameter is, the larger the L_2 norm is, and even when $q = 10^{-5}$, the L_2 norm is still less than that of SGD. It demonstrates that dropout as structured Frobenius norm restricts the size of diagonal elements well. Second, Figure 3.3(b) compares the sparsity of the learned metric, where sparsity is defined as

$$\text{Sparsity} = \frac{\#(M_{i,j} = 0)}{d^2}$$

Without the constraint of L_1 norm, the sparsity of the metric learned by SGD is small as shown by the blue dashed line. However, in SGD-M2 as plotted by the red dash dotted line, with an increasing of the structured L_1 penalty as the probability of dropout, the learned metric becomes more sparse, which confirms the effectiveness of SGD-M2. Finally, trace norm constraint usually leads to a low-rank metric [19], so we study the rank of the learned metric by SGD-D2 in Figure 3.3(c). As expected, when q becomes larger, more stress is put on the trace norm and lower-rank metric is induced.

Since dropout is found to be helpful to overcome overfitting in deep learning [47], we empirically study the role of dropout for alleviating overfitting problem in DML. Overfitting [51] is especially likely in cases where learning was performed too long, that is, too many iterations of learning is conducted in iterative methods. SGD methods in this chapter sample one triplet in each iteration. Therefore, more triplets sampled indicates longer learning. Morevoer, an overfitted model will likely perform worse on validation data outside the training data, even though the model performs as well, or perhaps even better, on the training data [51]. We fix all parameters as above except the number of sampled triplets, to study how training error and test error on the same training data and test data change, respectively, as the number of triplets used increases. Figure 3.4 shows the training error and test error of SGD, SGD-PSD and SGD with best dropout strategy on three small datasets, while the number of sampled triplets increases from $20,000$ to $100,000$. First, we observe that the training error of SGD with dropout is a little bit larger than those conventional SGD methods. This gap is from the large dropout probability q as we indicate in Theorem 3.3.1. However, overfitting is observed for SGD and SGD-PSD when the number of triplets is up to $40,000$, while there is no overfitting phenomenon for SGD with dropout. It further demonstrates the overwhelming performance of dropout technique in Table 3.2 and shows that dropout is also helpful to overcome the overfitting problem in DML.

3.4.2 Comparison with State-of-Art Methods

Besides the comparison with various SGD methods, we also compare our proposed dropout methods to three state-of-art DML algorithms as follows.

- **SPML** [114]: a mini-batch stochastic gradient descent method for DML to optimize the hinge loss with Frobenius norm as the regularizer.
- **OASIS** [24]: an online learning algorithm for DML and the symmetric version is adopted in the comparison.
- **LMNN** [127]: a batch learning algorithm with Frobenius norm for DML.

SPML and OASIS use the same triplet set as SGD methods and also adopt one-projection paradigm. LMNN is a batch learning method, and thus there is no limitation for the type and the number of triplets that it could use for each iteration. Specifically, LMNN is not restricted to the set of triplets used by other SGD methods. There is also no constraint for PSD projection in LMNN and it can perform PSD projection whenever it requires. All codes for these methods are from the authors and the recommended parameters are used. Since SPML is a stochastic method, it shares the same setting as the proposed methods, where the parameter for Frobenius norm is searched within the same range as q to choose the best performance. SPML and OASIS are both initialized with an identity matrix, while LMNN starts with the matrix from PCA without dimension reduction, which usually gives a better performance than the identity matrix [127].

Table 3.3 summarizes the classification accuracy of different DML algorithms. "**Dropout**" denotes the best result of dropout methods adopted from Table 3.2. It can be easily observed that, although LMNN is a batch learning method and could utilize much more information than our method, LMNN only has the similar performance on *dna* and *caltech10*, while SGD method with dropout significantly outperforms on all other datasets. It further demonstrates the effectiveness of the proposed methods. SPML and OASIS are slightly better than the standard SGD method, but significantly worse than SGD method with dropout technique. The performance of OASIS could be explained by the fact that it does not include any regularizer and overfitting could be easily induced. Although SPML combines the Frobenius norm as the regularizer, it is worse than SGD-M1 and SGD-M2 shown in Table 3.2, which implies that the proposed structured norm by dropout is more effective than the standard norm.

3.4.3 Wrap Dropout in Existing DML Methods

In this section, we show that dropout can be easily wrapped in the existing DML methods and help improve the performance. First, we wrap dropout in SPML, which is a state-of-art mini-batch SGD method for DML. Note that SPML has Frobenius norm as the regularizer, so we drop it to make sure that there is only one regularizer at one time. Since it is a SGD method, the dropout on M is the same as Algorithm 2 and Algorithm 3. We denote dropout as structured Frobenius norm on SPML as "SPML-M1" and dropout as structured L_1 norm as "SPML-M2". Instead of randomly selecting one triplet at each iteration, SPML samples b triplets at one time and updates according to the batch of the gradient. Therefore, for the dropout to the training data, we simply perform the dropout on different matrix A in the mini-batch as in Algorithm 5 and the method is denoted as "SPML-D".

Table 3.4 summarizes the results for wrapped SPML methods. First, it is not surprising to observe that all dropout strategies improve the performance of SPML. On almost all datasets, the improvement on accuracy is more than 1% and it is even about 3% on *protein*, which is also consistent with the observation in the comparison for various SGD methods. Although SPML applies the standard Frobenius norm as the regularizer, SPML with different dropout strategies outperforms it significantly according to the statistical significance examined via pairwise t-tests at the 5% significance level, which shows the superior performance of the proposed structured regularizer.

Then, we wrap dropout in OASIS, which is a state-of-art online learning method for DML. Since online learning has the similar process as stochastic gradient descent method, wrapping dropout in is pretty straightforward. Figure 3.5 illustrates the procedures of wrapping different dropout strategies in OASIS. Let "OASIS-M1", "OASIS-M2", and "OASIS-D" denote dropout as structured Frobeniuse norm, structured L_1 norm, and trace norm in OASIS, respectively. The comparison of classification accuracy applied by 3-NN is summarized in Table 3.5. The similar phenomenon as for SPML is observed, that is, dropout always

Table 3.4: Comparison of classification accuracy (%) on SPML and its variants by wrapping different dropout strategies in. The best result is bolded.

	ta	*semeion*	*dna*	*caltech10*	*protein*	*sensit*
SPML	83.56±0.50	94.60±0.27	92.85±0.37	65.46±0.50	59.15±0.37	74.99±0.10
SPML-M1	84.59±0.38	**95.48±0.41**	93.27±0.16	**66.68±0.70**	61.38±0.36	75.63±0.08
SPML-M2	84.46±0.44	95.27±0.35	**93.91±0.32**	66.15±0.40	61.03±0.48	75.90±0.23
SPML-D	**84.74±0.56**	95.44±0.27	93.41±0.23	66.27±0.36	**62.14±0.68**	**76.84±0.13**

Table 3.5: Comparison of classification accuracy (%) on OASIS and its variants by wrapping different dropout strategies in. The best result is bolded.

	ta	*semeion*	*dna*	*caltech10*	*protein*	*sensit*
OASIS	83.20±0.95	94.06±0.19	88.57±0.28	65.06±0.40	58.83±0.38	73.50±0.15
OASIS-M1	84.74±0.86	**95.69±0.43**	93.35±0.33	**67.11±0.31**	62.44±0.55	75.47±0.15
OASIS-M2	**84.79±0.79**	94.73±0.38	**94.33±0.54**	66.12±0.49	62.61±0.60	75.47±0.12
OASIS-D	84.38±0.59	95.23±0.48	93.15±0.28	66.71±0.62	**63.06±0.55**	**76.56±0.15**

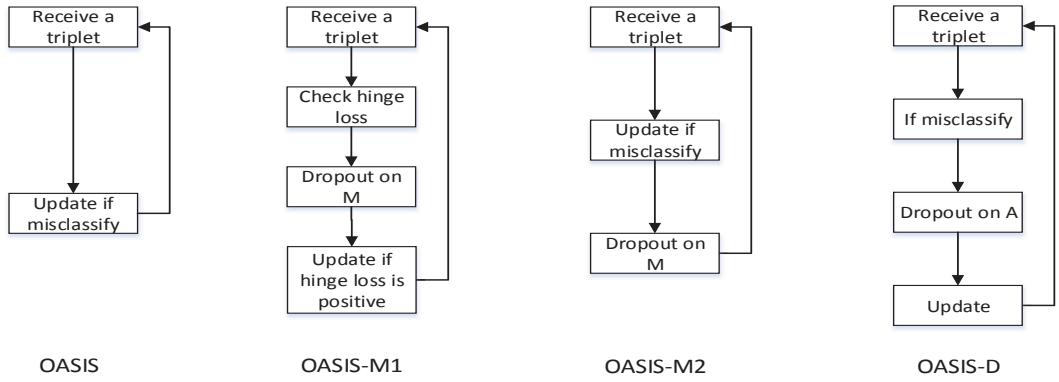


Figure 3.5: Procedure of wrapping dropout in OASIS.

helps to improve the performance of OASIS significantly according to pairwise t-test at the 5% significance level.

In summary, wrapping dropout in existing DML methods is not only convenient but also very helpful for performance improvement.

3.5 Summary

In this chapter, we propose two strategies to perform dropout for DML, i.e., dropout in the learned metric and dropout in the training data. For dropout in the metric, we propose the structured regularizer, which is simulated with dropout by assigning different dropout probabilities for the diagonal elements and those living in off diagonal. For dropout in the training data, the data-dependent dropout probability is adopted to mimic the trace norm. We develop the theoretical guarantees for both dropout scenarios to show that dropout will not affect the convergence rate of SGD. Furthermore, we demonstrate that the proposed strategies are very convenient to wrap in the existing DML methods. Our empirical study confirms that the proposed methods have the overwhelming performance compared with the baseline methods, and can significantly improve the classification accuracy for the state-of-art DML methods.

Chapter 4

Mining Balanced Hierarchical Clustering

In this chapter, we aim to construct a balanced hierarchical clustering structure containing semantics to help end users efficiently and effectively organize and search in large amount of data. We study this problem for the specific application of scenario II, that is, retrieving relevant images from a huge collection of images by iterative search.

4.1 Background and Overview

You just came back from a wonderful vacation at your favorite place. Your digital camera, equipped with only a 64GB flash memory card, records your lovely memory in thousands of pictures. In your computer, you have tens of thousands of photos taken in the last 5 years. You want to share those photos online with your friends, but you face a challenge. Many of your friends unlikely have the time and patience to browse thousands of photos. Ideally, you would like to organize your thousands of photo using a small number of meaningful features, such as "places", "my kids", "our pets", and "classmate reunion", so that every photo can be uniquely identified and retrieved using a combination of those features. In other words, the features have to be discriminative and independent. We call such a small number of meaningful features a *multidimensional index* (or *index* for short) of the photos. Here, an index is for human users instead of for software search engines. It should be easy to understand and manipulate.

Creating an index of thousands of photos manually is time-consuming. Moreover, a manually created index without a careful design may not be able to facilitate search and retrieval effectively. For example, using too many features may overwhelm users. Some photos may need a combination of many features to be retrieved, and thus are deeply hidden. Many photos may not be uniquely identified by feature combinations.

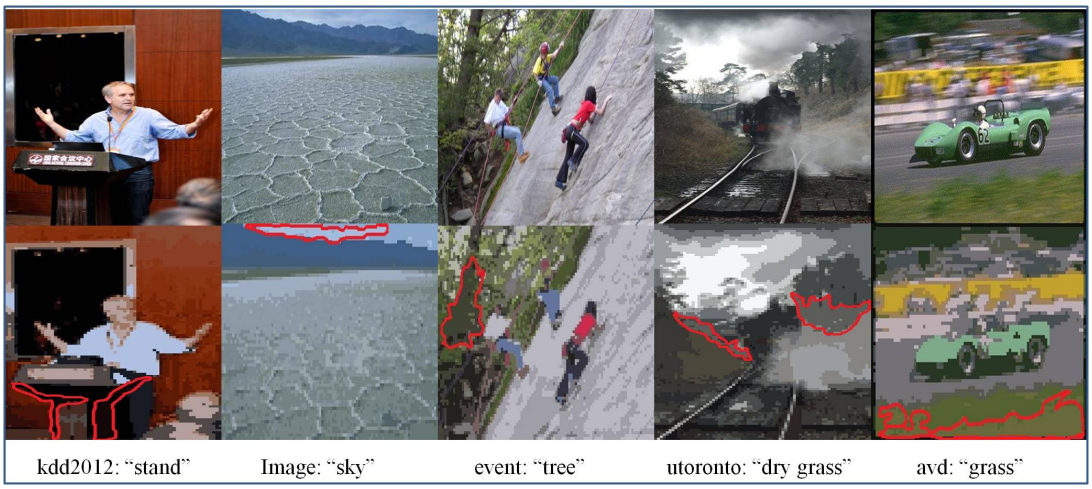


Figure 4.1: The 1st dimension selected with manual label.

Most existing image retrieval techniques [105] rely heavily on images manually or algorithmically annotated, which is often expensive and unavailable in our problem setting. Fortunately, by applying the state-of-art methods, we can extract a large set of meaningful semantics from the original photos as "dimensions". Concretely, each photo may contain multiple semantic dimensions, such as "chair", "computer", "sunset", and "ocean". Inspired by the idea of multi-instance multi-label learning [82, 144], where each data object is represented by multiple instances, all meaningful regions for each image can be roughly detected by image segmentation, and then each region can be presented as an instance. Consequently, each instance can be treated as a semantically meaningful dimension for people to search, view, and organize these images. For example, given the photo collection from the KDD2012 conference, the dimension "stand" as red line circled in Figure 4.1 could be used to indicate whether an image has the content of "stand".

However, such semantic annotation methods typically generate many features. One may wonder whether the more features, the better a large set of photos are indexed. If one uses all features as dimensions, many dimensions may be redundant, since different images may contain similar content. A dimension existing in all or many images are not discriminative and not useful for indexing photos. For example, image feature "football" for a collection of football game photos is not informative, since most photos contain footballs.

Thus, although extracting semantic features from photos is highly feasible, the real challenge of selecting effective features to efficiently search in a large number of objects largely remains. Ideally, given a collection of images, we want to find a minimum set of dimensions such that each image in the collection can be uniquely identified by some of those selected dimensions. Moreover, any sub-group of images that contain the same subset of features should be easily located together for search by keywords in mind as "sunset" and "mountains".

One may wonder if this is just an instance of the well studied feature selection problem [45]. However, the lack of label or class information makes some popular feature selection methods, such as Information Gain, Relief, Fisher Score, and Lasso, not applicable here. Some unsupervised feature selection methods have been proposed for clustering [4]. However, those methods usually try to select features that can improve the quality of clustering that groups similar objects having similar feature values together. Since our goal is to uniquely index each image or any sub-group of similar images, the unsupervised feature selection methods for clustering are not reliable for our problem, either. Therefore, although our problem is a kind of feature selection, it cannot be tackled well using the existing methods.

Hierarchical clustering structures generated by traditional hierarchical clustering methods [94] can provide unique paths to locate each identical object or sub-group of objects. However, this kind of hierarchies do not contain any semantic meanings. Therefore, it is impractical to induce understandable questions from the hierarchy to help users search in the hierarchy. Moreover, traditional hierarchical clustering methods often generate very unbalanced hierarchies which make search inefficient.

In this chapter, we formulate the problem of subspace hierarchical clustering that aims to construct a balanced hierarchical clustering structure using a subset of dimensions. Moreover, the hierarchy contains semantics for humans to efficiently search desired objects. Specifically, this problem can be solved by finding a minimum subset of dimensions such that each object is uniquely identified. This problem is very challenging, and we prove that this optimization problem is submodular [77]. Therefore, we propose an efficient greedy algorithm to naturally construct a balanced hierarchy with semantics. Our experiments on various real-world image collections validate both the efficiency and effectiveness of our proposed method.

In the following of this chapter, Section 4.2 formulates our problem. Section 4.3 presents our method. Section 4.4 reports an empirical study. Section 4.5 summarizes this chapter.

4.2 Problem Definition

In this section, we first describe how we can extract candidate features from a set of raw images without any domain knowledge. Then, we present our problem of feature selection.

4.2.1 Candidate Feature Extraction

Given a large set of images, without any additional information (i.e., in an unsupervised manner), how can we extract meaningful features automatically?

As mentioned above, each image may contain multiple meaningful semantics. Each semantically meaningful region can be detected for each image, which can be roughly done by the existing image segmentation techniques. For instance, Wang *et al.* [126] proposed a

segmentation method that partitions each image into blocks with 4×4 pixels and extracts a feature vector for each block (color feature and texture feature). Then the k-means algorithm [50] is used to cluster the feature vectors into several clusters such that every cluster contains multiple blocks and form a meaningful region.

We borrow the idea of MIML (Multi-Instance Multi-Label learning) [82, 144], which present each data object by multiple instances. After segmentation, we present each image as a bag with multiple regions, that is, $I_i = \{\mathbf{x}_{ij} | j = 1, \ldots, N_i\}$, where each region is described as an *instance* $\mathbf{x}_{ij}$ which is the average feature vector over all its blocks, and N_i is the total number of instances in image I_i. $X = \{\mathbf{x}_{11}, \ldots, \mathbf{x}_{1N_1}, \ldots, \mathbf{x}_{n1}, \ldots, \mathbf{x}_{nN_n}\}$ denotes the set of instances collected from all n images. Therefore, X can be treated as the set of extracted candidate dimensions, further denoted as $D = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_{|X|}\}$, where each $\mathbf{d}_j$ has a meaningful semantic.

We then map each image onto the extracted dimension set D by finding the minimum Euclidean distance from its instances $I_i = \{\mathbf{x}_{ij} | j = 1, \cdots, N_i\}$ to each dimension in D. Formally, each image is represented by the distance feature vector as $[\phi_1, \cdots, \phi_{|D|}]$, where the q-th feature value is calculated as

$$\phi_q(I_i) = \min_{j=1,\cdots,N_i} ((\mathbf{x}_{ij} - \mathbf{d}_q)^\top (\mathbf{x}_{ij} - \mathbf{d}_q))^{\frac{1}{2}} \tag{4.1}$$

Thereafter, by comparing the feature value with a predefined distance threshold θ, we determine if an image contains the corresponding dimension. Specifically, if $\phi_{iq} \leq \theta$, then we set $\mathbf{O}_{iq} = 1$, which means image I_i has the q-th semantic content, otherwise 0. As such, a new feature space $\mathbf{O} \in \{0, 1\}^{n \times |D|}$ is generated. The whole process is illustrated in Figure 4.2.

4.2.2 Problem Formulation

To construct a balanced hierarchical clustering structure in a minimum subset of dimensions, we propose to find a minimum subset of dimensions such that each image can be uniquely indexed. Thereafter, a decision tree [106] for efficient retrieving can be constructed by choosing the most discriminative dimension for each level, although this step will be naturally embedded into the feature selection step as discussed later.

Specifically, given the candidate dimenion set D and the corresponding presentation for all images $\mathbf{O} \in \{0, 1\}^{n \times |D|}$, suppose that each image can be uniquely represented by $\mathbf{O} \in \{0, 1\}^{n \times |D|}$ (cases beyond this assumption will be discussed separately in Section 4.3.2), which means $\forall p, q \in [1, n], p \neq q, \mathbf{o}_p \neq \mathbf{o}_q$, our goal is to select a minimum subset of dimensions $D' \subseteq D$, such that each image can still be uniquely represented by $\mathbf{O}' \in \{0, 1\}^{n \times |D'|}$. The problem can be formulated as the following optimization problem.

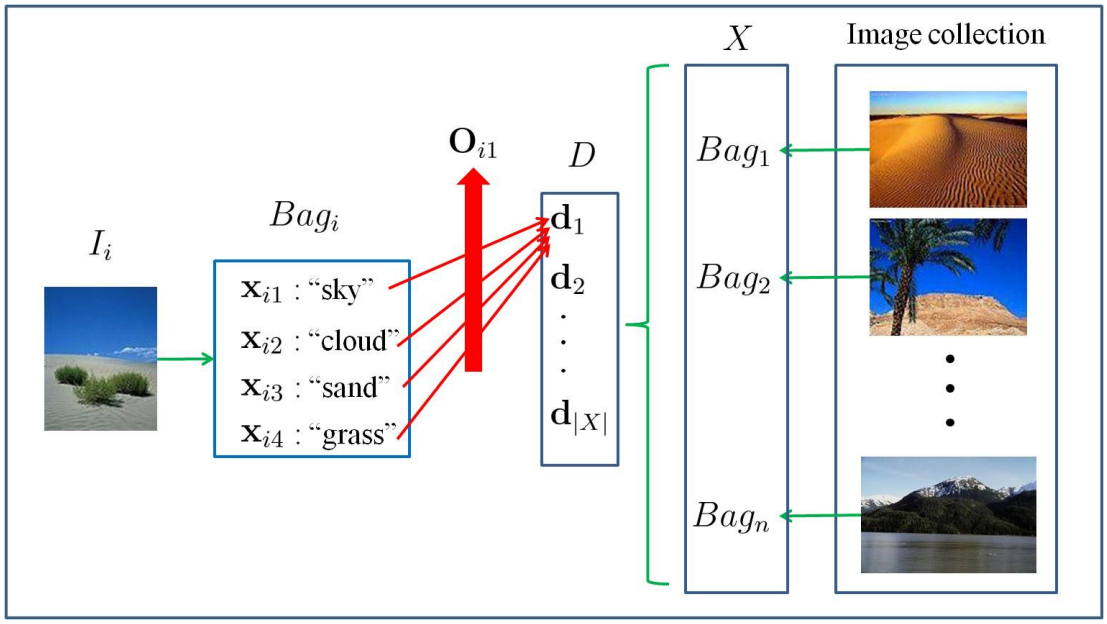


Figure 4.2: Feature extraction.

$$\begin{aligned} \arg\min_{D' \subseteq D} \quad & |D'| \\ s.t. \quad & \mathbf{o}'_p \neq \mathbf{o}'_q, \forall p, q \in [1, n], p \neq q \\ & \mathbf{o}'_i \neq \mathbf{0}, \forall i \in [1, n] \end{aligned} \tag{4.2}$$

where the second constraint requires that every image should be covered by selected dimensions.

The optimization problem in Equation 4.2 is equivalent to the following optimization problem.

$$\begin{aligned} \arg\max_{D' \subseteq D} \quad & N_{dif}^{D'} \\ s.t. \quad & N_{dif}^{D^0} < N_{dif}^{D'}, \forall D^0 \subseteq D, |D^0| < |D'|, \\ & \mathbf{o}'_i \neq \mathbf{0}, \forall i \in [1, n] \end{aligned} \tag{4.3}$$

where $N_{dif}^{D'}$ indicates the number of different pairs of images generated upon the dimension set D'. Taking the image collection in Table 4.1 as an example, dimension set $\{d_1, d_2\}$ generates $N_{dif}^{\{d_1,d_2\}} = 2$ different pairs $\langle \mathbf{o}_1, \mathbf{o}_2 \rangle$ and $\langle \mathbf{o}_2, \mathbf{o}_3 \rangle$. Both $\{d_1, d_4\}$ and $\{d_1, d_3, d_4\}$ generate 3 different pairs which is the maximum number, however $\mathbf{o}_3$ is not covered by any dimension selected in $\{d_1, d_4\}$.

Theorem 4.2.1. *Optimization problem in Equation 4.2 is equivalent to the optimization problem in Equation 4.3.*

Table 4.1: An image collection with 3 images on 4 dimensions.

	d_1	d_2	d_3	d_4
$\mathbf{o}_1$	0	1	1	1
$\mathbf{o}_2$	1	1	0	0
$\mathbf{o}_3$	0	1	1	0

Proof. As stated, we assume that each image is uniquely represented by $\mathbf{O} \in \{0,1\}^{n\times|D|}$, which means $\forall p, q \in [1,n], p \neq q, \mathbf{o}_p \neq \mathbf{o}_q$. Suppose the solution set for Equation 4.2 is $\mathcal{A}$ and the solution set for Equation 4.3 is $\mathcal{B}$.

1. For any solution $A \in \mathcal{A}$, it provides a minimum subset that every image is uniquely represented. The number of different pairs generated by A is $\binom{n}{2}$, which is maximal. The size of $|A|$ is minimum, and thus $A \in \mathcal{B}$. Therefore, $\mathcal{A} \subseteq \mathcal{B}$.

2. For any solution $B \in \mathcal{B}$, since we assume that each image is uniquely represented by $\mathbf{O} \in \{0,1\}^{n\times|D|}$, $N_{dif}^{B} = \binom{n}{2}$. Moreover, there exists no other solution B' that $|B'| < |B|$. That means B is a minimum subset of D that can uniquely identify each image, and thus $B \in \mathcal{A}$. Therefore, $\mathcal{B} \subseteq \mathcal{A}$.

In summary, $\mathcal{A} = \mathcal{B}$. □

4.3 The Proposed Method

Now we prove that the optimization problem in Equation 4.3 is *submodular* [77], that is, it exhibits a diminishing returns property: adding a dimension when there are only a few dimensions adds more different pairs than adding it after gathering many dimensions. Let the number of different pairs generated upon dimension set D be $\mathcal{R}(D) = N_{dif}^{D}$. We first give the following lemma.

Lemma 4.3.1. *For all dimension sets $A \subseteq B \subseteq D$ and dimension $d \in D \setminus B$,*

$$\mathcal{R}(A \cup \{d\}) - \mathcal{R}(A) \geq \mathcal{R}(B \cup \{d\}) - \mathcal{R}(B).$$

Proof. Given a new dimension d, we can calculate the number of different pairs generated by it. For instance, d_3 in Table 4.1 will generate two different pairs: $\langle \mathbf{o}_1, \mathbf{o}_2 \rangle$ and $\langle \mathbf{o}_2, \mathbf{o}_3 \rangle$.

Let the set of different pairs generated by dimension set $\{d\}$ be $S_{\{d\}}$ and that of D be S_D. Obviously, $\mathcal{R}(A \cup \{d\}) - \mathcal{R}(A) = |S_{\{d\}} - S_A \cap S_{\{d\}}|$ and $\mathcal{R}(B \cup \{d\}) - \mathcal{R}(B) = |S_{\{d\}} - S_B \cap S_{\{d\}}| = |S_{\{d\}} - (S_A \cup S_{B-A}) \cap S_{\{d\}}| = |S_{\{d\}} - S_A \cap S_{\{d\}} - S_{B-A} \cap S_{\{d\}}| \leq |S_{\{d\}} - S_A \cap S_{\{d\}}| = \mathcal{R}(A \cup \{d\}) - \mathcal{R}(A)$. □

Using Lemma 4.3.1, we have the following result.

Theorem 4.3.1. *$\mathcal{R}(D)$ is a submodular.*

Proof. 1. $\mathcal{R}(\emptyset) = 0$, which is trivial because all images carry the same dimension values without adding any dimension;

2. Since adding more dimensions does not reduce the number of different pairs, $\mathcal{R}$ is non-decreasing, i.e., $\mathcal{R}(A) \leq \mathcal{R}(B)$ for all $A \subseteq B \subseteq D$;

3. Follow Lemma 4.3.1.

Therefore, $\mathcal{R}(D) = N_{dif}^{D}$ is a submodular. □

Maximizing submodular in general is NP-hard [70]. A commonly used heuristic is *greedy algorithm*. In our case, the greedy algorithm begins with the empty dimension set $D_0 = \emptyset$, and iteratively, in step t, adds dimension d_t which maximizes the increased number of different pairs as

$$d_t = \arg \max_{d \in D \setminus D'_{t-1}} \mathcal{R}(D'_{t-1} \cup \{d_t\}) - \mathcal{R}(D'_{t-1})$$

Since all dimensions are equally considered, according to theorem stated in [96], if the greedy algorithm stops when M dimensions are selected,

$$\mathcal{R}(D_M) \geq (1 - 1/e) \max_{D' \subseteq D, |D'| = M} \mathcal{R}(D')$$

which means this simple greedy algorithm is near-optimal.

Therefore, we propose a greedy algorithm summarized in Algorithm 6. It can be observed that this algorithm naturally produces a hierarchical clustering structure in a style of binary decision tree, where the root is the whole data set, all interior nodes have two children (i.e., containing a corresponding semantic meaning or not), and each leaf uniquely identifies each image. Since each step chooses the dimension that generates the largest number of different pairs, the hierarchy provides efficient paths (i.e., the length of each path is upper bounded by the height of the tree) to retrieve either a specific image or any sub-group of similar images. We call this hierarchy is most balanced because for each node in the tree, the number of images in the leaf child and that in the right child are most balanced.

4.3.1 Analysis

Let the height of the binary tree formed by Algorithm 6 be T, we have $\log_2 n \leq T \ll n$, where n is the total number of images. For each level h, there will be at most $|D|$ dimensions to be selected and for each dimension, we can calculate how many pairs will be added for each tree node in this level within one step. As we know the total number of tree nodes in the tree that should be calculated is at most $2^0 + 2^1 + 2^2 + \cdots + 2^{(T-1)} = 2^T - 1$. Therefore, in the worst case, the total running time is $O(|D| \cdot 2^T)$ that is between $O(n|D|)$ and $O(n^2|D|)$, while in fact, the running time is very close to $O(n|D|)$.

Algorithm 6 Subspace Hierarchical Clustering

Input: D: Candidate dimension set
$\mathbf{O} \in \{0,1\}^{n \times |D|}$: Image representations over D
Output: D': Selected dimensions
1: $D' = \emptyset$
2: Initialize root as $\{1, 2, \cdots, n\}$
3: $t = 1$
4: **while** any leaf has more than one image **do**
5: **for** each $d \in D \setminus D'$ **do**
6: calculate the number of added pairs by $\mathcal{R}(D' \cup \{d\}) - \mathcal{R}(D')$
7: **end for**
8: $d_t = \arg \max_{d \in D \setminus D'} \mathcal{R}(D' \cup \{d\}) - \mathcal{R}(D')$
9: **for** each leaf **do**
10: move each image i with $\mathbf{o}_{it} = 0$ to the left child
11: move each image j with $\mathbf{o}_{jt} = 1$ to the right child
12: **end for**
13: $D' \leftarrow D' \cup \{d_t\}$
14: $t = t + 1$
15: **end while**
16: **if** the most left image i is represented as $\mathbf{o}' = \mathbf{0}$ **then**
17: Randomly pick $d_t \in D \setminus D'$ with $\mathbf{o}_{iq} = 1$
18: $D' \leftarrow D' \cup \{d_t\}$
19: **end if**

4.3.2 Extensions

For real-world image collections, it is possible that all extracted dimensions cannot uniquely identify each image. In such a case, there is no way to distinguish images with the same dimension values. If α images have the same dimension values in the original image collection, we can change the objective to finding a minimum subset of dimensions that at most α images have the same dimension values, which is formulated as

$$\begin{aligned} \arg \min_{D' \subseteq D} \quad & |D'| \\ s.t. \quad & |Leaf|_{max} \leq \alpha, Leaf = \{\mathbf{o}' | \forall i, j, \mathbf{o}'_i = \mathbf{o}'_j\} \\ & \mathbf{o}'_i \neq \mathbf{0}, \forall i \in [1, n] \end{aligned}$$

Algorithm 6 can be adopted while Line 4 changes to "any leaf has more than α images".

Moreover, during the search, it is possible that a user may stop the search at the tree root. For example, a user may stop the search after he/she ticks the "sky" region as shown in Figure 4.1. It means that the user wants images about "sky". In the best case, $n/2$ images have the corresponding dimension value. It is still not convenient for people to view all output images, while n is very large. In such a scenario, a ranked list of images are output, where the top-ranked image is with the smallest distance to the ticked dimension using Equation 4.1, where distances are already stored when calculating $\mathbf{O}$ in Section 4.2.1.

Figure 4.3: Segmentation examples on the *kdd2012* data set.

If a user ticks several dimensions, the image with the smallest summation of distances to all ticked dimensions can be top-ranked.

4.4 Experimental Results

To validate the effectiveness and efficiency of our proposed method, we conduct experiments on several real-world photo/image collections: the *kdd2012* conference photo collection[1], which has 1,503 images in total, the *Image* [140] data set with 2,000 images, the *event* [80] data set with 1,579 images, *utoronto* [88], a subset of the ImageNet data set with 1,998 useful images, and *avd*, a subset of COREL data set with animals, vehicles and distractors [79] including 3,979 images.

All our experiments are conducted on a 64bit-Windows sever with a 2.5Hz CPU and 8G main memory. First, we resize all original images into smaller ones with size no larger than 400×400 pixels. It takes about ten minutes to resize 2,000 images. Following the image segmentation technique described in [126], each image is segmented into at least 2 and at most 16 regions. Note that 6 features are used for segmentation: 3 LUV color features averaged over the 4×4 block and 3 wavelet texture features in the block. It takes about 1.5 hours to segment 2,000 images. Samples are shown in Figure 4.3, where the upper row shows the original images and the lower row shows the segmented images with each region represented by the same color.

After segmentation, each region may contain multiple 4×4 blocks. We represent each region as an instance by 25 features, 6 features of the cluster center found in the segmentation process, 3 RGB features averaged over all its blocks, 3 averaged HSV color features, 6 averaged color moment features, 3 averaged wavelet texture features, and the total number

[1] `http://www.flickr.com/photos/sigkdd2012`

of blocks within this region. Therefore, each image is represented as a bag of at most 16 instances, each with 25 features.

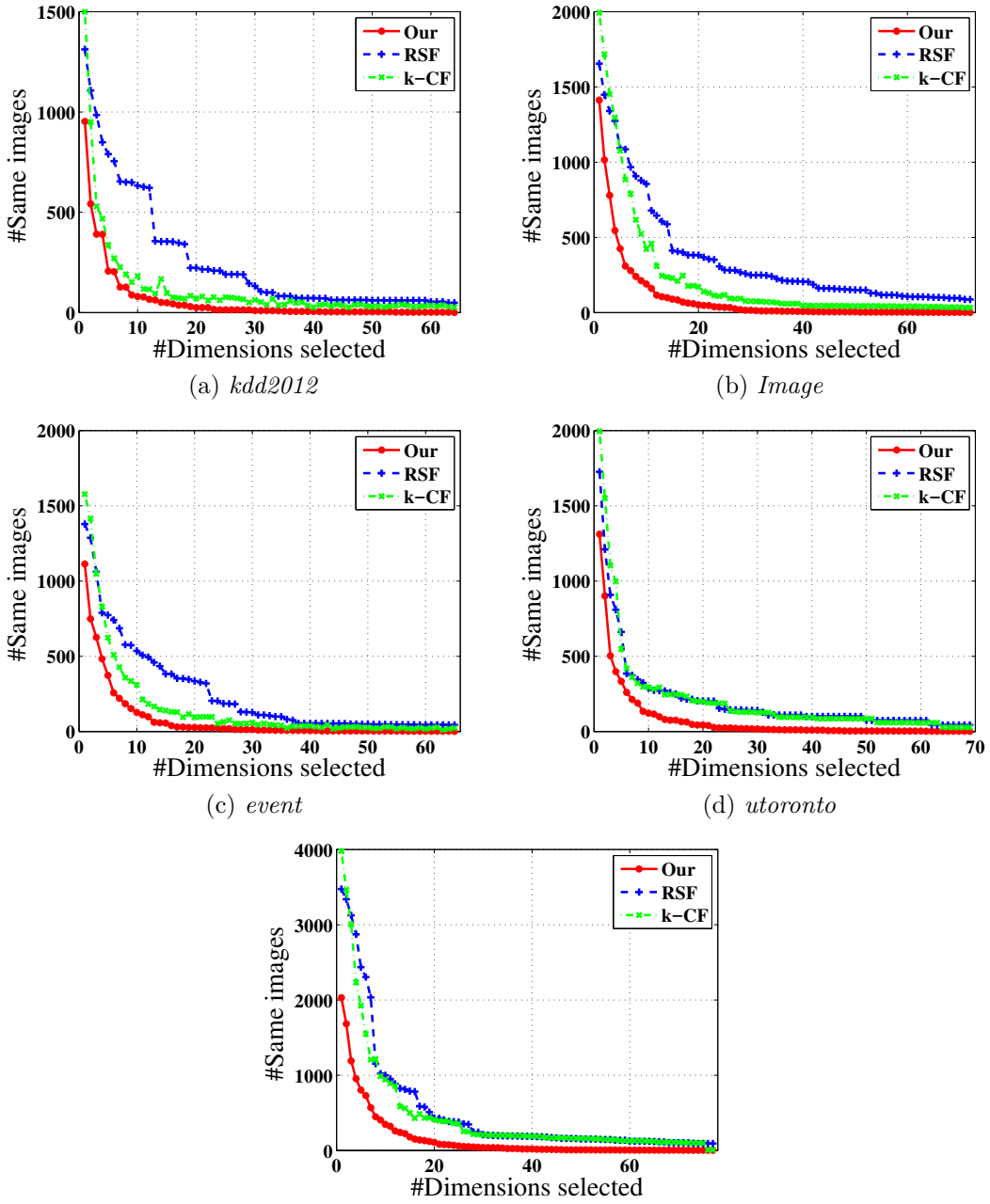
Then, we map each image onto dimensions collected from all images by finding the minimum Euclidean distance from its own instances to each dimension. For convenience, instead of using the distance value, each image is represented by a weight vector $[\psi_1, \cdots, \psi_{|D|}]$, where the q-th value is calculated by $\psi_q = \exp(-\phi_q(I_i)^2/\sigma^2)$, and σ is the mean distance following the method in [145]. In the experiments, if $\psi_q \geq \theta$, where θ is a threshold parameter and $\theta \in \{0.95, 0.90, 0.85, 0.80, 0.75, 0.70\}$, the image contains the q-th dimension and we set $\mathbf{o}_{iq} = 1$, otherwise 0.

To the best of our knowledge, there is no existing method that is exactly for our problem. We propose some simple baselines for comparison. Two baselines are searching by random selection. RSF (Random Selection Forward searching) begins with an empty set and randomly selects a feature at each iteration until each image is uniquely represented and RSB (Random Selection Backward searching) begins with the whole set and randomly removes a feature at each iteration until at least one image cannot be uniquely identified. The other baseline is k-CF (k-means Clustering Forward searching) that adopts the k-means clustering algorithm [50] to do clustering over all features. Then each cluster centroid as a dimension is selected. However, it is impractical to determine the number of k which can exactly identify each image uniquely. Therefore, k-means clustering is repeatedly applied and k is increased by 1 in each iteration from $k_0 = 1$ until each image can be uniquely identified.

4.4.1 Performance

In this section, we set the parameter $\theta = 0.70$ and compare the performance of our method to all proposed baselines in four aspects 1) the total number of features selected; 2) efficiency (CPU time); 3) the maximum number of images that have the same dimension values as the number of dimensions selected increases (only for forward searching method); 4) the number of different image sets (a set contains images that have the same dimension values on current selected dimensions) generated as the number of dimensions selected increases (only for forward searching method). Specifically, 3rd and 4th aspects are comparing different methods on their convergence speed. Note that k-means clustering is very slow especially when k increases to a large number due to the huge number of candidate features as the CPU time shown in Table 4.2. For example, it takes about 37.5 hours for k-CF to select only up to 75 features for data set *event*. Therefore, we compare to k-CF only in the 3rd and 4th aspets. For RSF and RSB, the best result of 10 trials for each data set is reported.

The total number of candidate dimensions extracted for each data set is summarized in Table 4.2, from which we can see that the total number of features selected by our method is much less than RSF and RSB. It indicates that the proposed method can provide much more efficient hierarchy for search, because the number of selected dimensions determines

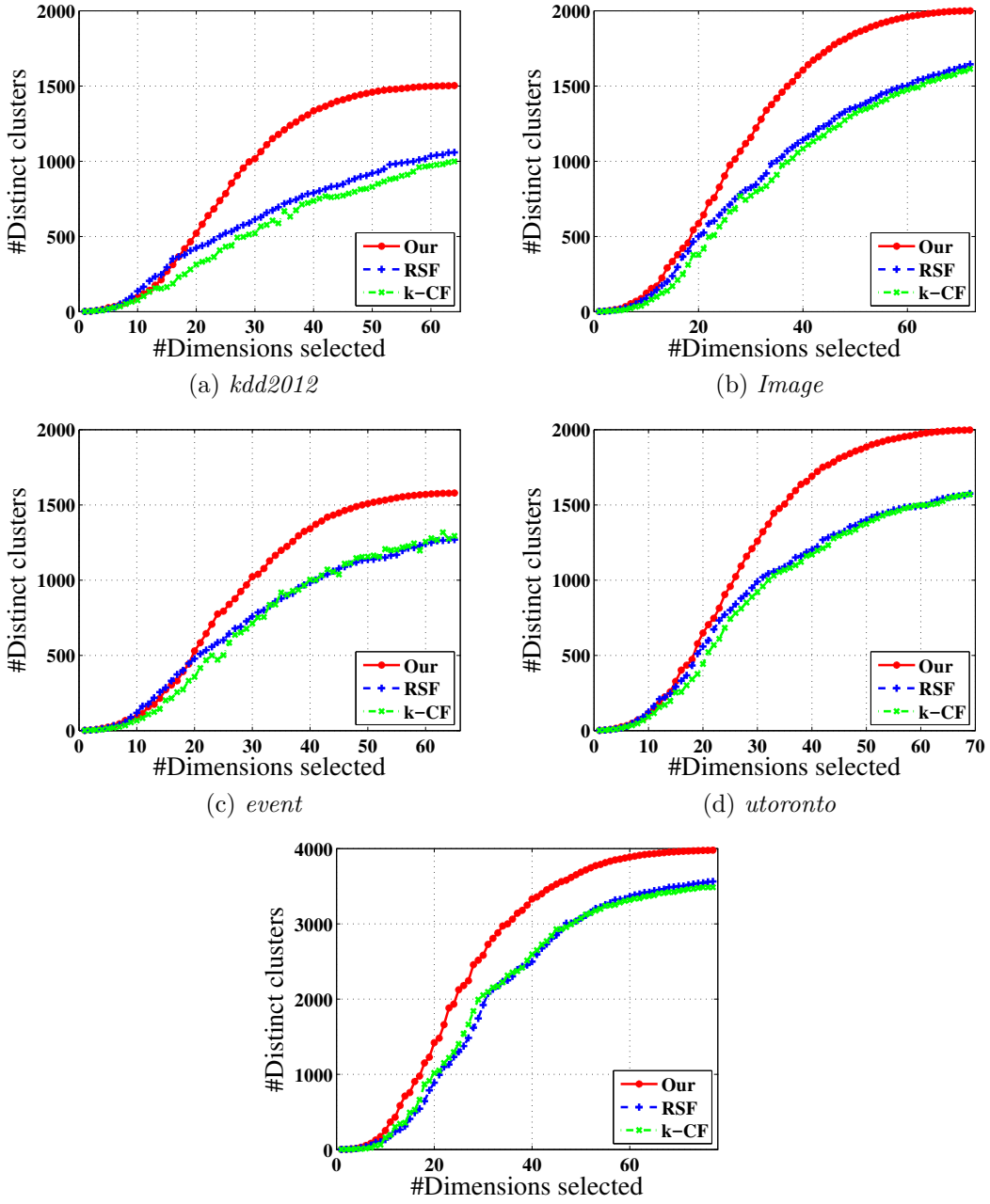


(a) *kdd2012* (b) *Image*

(c) *event* (d) *utoronto*

(d) *avd*

Figure 4.4: Maximum #images with same dimension values as selected dimensions increases when $\theta = 0.70$.



(a) *kdd2012* (b) *Image*

(c) *event* (d) *utoronto*

(d) *avd*

Figure 4.5: #Distinct image sets as selected dimensions increases when $\theta = 0.70$.

Table 4.2: Performance comparison when $\theta = 0.70$.

Data set	#Images	#Candidates	Method			CPU time (sec)		
			Our	RSF	RSB	Our	RSF	RSB
kdd2012	1,503	20,251	64	935	20,098	54.08	61.55	211.17
Image	2,000	24,923	72	612	24,757	105.28	119.25	420.07
event	1,579	21,992	65	463	21,829	64.89	70.18	293.73
utoronto	1,998	25,116	69	812	24,976	125.37	203.34	314.07
avd	3,979	56,900	77	1,834	56,561	569.43	613.27	2,021.5

Table 4.3: Effect of θ ("-" means no solution).

Data	*kdd2012*		*Image*		*event*		*utoronto*		*avd*	
Method	Our	RSF	Our	RSF	Our	RSF	Our	RSF	Our	RSF
$\theta = 0.75$	71	675	87	1,735	76	531	79	876	100	3,230
$\theta = 0.80$	81	936	110	3,529	98	1,344	96	1,255	133	4,612
$\theta = 0.85$	100	1,899	175	10,002	144	4,022	129	4,455	206	8,112
$\theta = 0.90$	158	3,385	-	-	264	6,605	-	-	391	16,433
$\theta = 0.95$	367	8344	862	15,750	659	11,014	540	14,003	1,062	38,319

the height of the clustering hierarchy. At the same time, Table 4.2 demonstrates that the efficiency of our proposed feature selection method is even slightly better than random searching, due to the relatively smaller number of dimensions selected.

Figure 4.4 shows the maximum number of images that have the same dimension values on current selected dimensions as the number of dimensions selected increases and Figure 4.5 shows the number of different image sets generated as the number of selected dimensions increases. The best result for RSF is plotted too. Figure 4.4 indicates that our method converges much faster than those baselines, while Figure 4.5 implies that with the same amount of selected dimensions, the dimensions selected by our method are more discriminative. Therefore, our method clearly outperforms the baselines.

4.4.2 Effect of Parameter θ

In this section, we study the effect of parameter θ. Since RSB is much worse than RBF due to the possible reason that it may randomly remove some essential features at the very beginning, we only compare our method with RSF in this subsection. The larger θ, the less 1's generated in $\mathbf{O}$. When θ increases, it is also possible that some images cannot be uniquely identified using all available dimensions. In this case, there will be no solution for comparison as shown by "-" in Table 4.3. At the same time, a larger number of features need to be selected as θ increases, as shown in Table 4.3. Our method is much better than RSF for different values of θ.

4.4.3 Illustration of Selected Dimensions

For selected dimensions, we can easily map them back to their specific bag instances and then locate the corresponding regions in the original images, which, for example, can be shown as circled in red line in Figure 4.1. With less than 100 dimensions selected, it is much easier for people to give manual labels as "stand" and "sky" shown in Figure 4.1, compared with labeling millions of images.

4.5 Summary

To automatically construct a hierarchical clustering structure with semantics for a large image collection without domain knowledge, we first propose a simple strategy to automatically extract meaningful semantics from original images as dimensions. Upon an even larger set of dimensions collected, we then propose an efficient unsupervised feature/dimension selection method to select a sufficient dimension subset to represent each photo within the collection uniquely for indexing, which naturally constructs a most balanced binary tree structure for efficient search. Experiments on a board of variety real-world image collections demonstrate both the efficiency and effectiveness of our proposed method.

Chapter 5

Finding Multiple Stable Clusterings

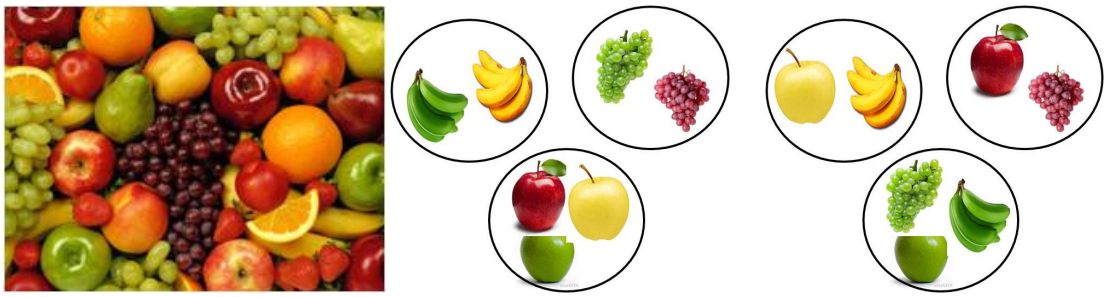
Considering that orthogonal ways may exist to partition a given data set, we study how to automatically discover multiple clustering structures hidden in a given data and provide understandable feature subspaces for them to cover users' search interests in this chapter.

5.1 Background and Overview

Many clustering methods were proposed in literature. Some recent surveys on the subject include [61, 134, 13, 133]. Traditional clustering methods [61] focus on finding a single way to partition data into groups. However, it has been well recognized that different outputs are possible if one varies the parameter setting, changes the clustering algorithm, or uses different subsets of features in analysis. This means that assuming only a single clustering for a data set can be too strict, which has motivated the emerging area of multi-clustering [93].

In many situations, different orthogonal ways may exist to partition a given data set, each way presents a unique aspect to understand the structure of the data. For example, fruits can be clustered by species or by color. An illustrative example is shown in Figure 5.1. They can even be clustered by nutrition components and in some other ways. In customer relationship management, customers can be clustered by gender, region, job, age, religion, purchase behavior, credit history and many other ways. As another example, mining phenotypes [120] is very useful in bioinformatics and healthcare informations, and is essentially a multi-clustering problem.

To obtain multiple good and different clusterings, where a specific way of partitioning the data is called a *clustering*, two general strategies were proposed. Semi-supervised multi-clustering methods [10, 31, 137] focus on finding one or more alternative clusterings with respect to a given clustering. They generate multiple clusterings in a greedy way such that multiple clusterings are produced sequentially and a new clustering is required to be



(a) Fruits (b) Clustering by species (c) Clustering by color

Figure 5.1: Fruits can be clustered in different ways (Images are from Internet).

different from the previous ones. For example, COALA [8] aims to find an alternative clustering of high quality and high dissimilarity comparing to the given clustering. The major idea is to add instance level constraints, such as linked pairs in the given clustering being transformed to cannot-link constraints.

Semi-supervised multi-clustering methods highly rely on a given clustering as the input. Consequently, the alternative clustering result may not capture a user's interest exactly. To handle this problem, unsupervised multi-clustering methods [21, 62, 32] try to simultaneously generate multiple clusterings that are constrained to be different from each other. However, to cover users' interest, unsupervised multi-clustering methods tend to generate many alternative clusterings. It is hard to constrain the differences between the clusterings computed. It is also overwhelming for users to absorb and understand the results.

Apparently, multi-clustering methods lead a challenge to end users: how to efficiently and effectively understand many clusterings. Subspace multi-clustering methods discover multiple clusterings by considering different feature subspaces. Then, each clustering corresponds to a feature subspace. Users can interpret and understand each clustering structure using its feature subspace, such as species or color in the example of Figure 5.1. For example, CLIQUE [3] divides a multidimensional data space into grid-cells, each dimension being partitioned into equal-width intervals. Then, dense cells in each subspace are identified using a density threshold. A group of connected dense cells in a subspace is regarded as a cluster. A clustering can be produced accordingly within a subspace. Obviously, CLIQUE has to search an exponential number of subspaces with respect to the number of attributes. Although some fast heuristic variants, such as INSCY [5], were proposed, the scalability remains a challenge. Another drawback is that these subspace multi-clustering approaches cannot explicitly consider the dissimilarity between different clusterings. Such methods tend to produce many clusterings in order to cover some interesting ones, which may likely overwhelm users.

In this chapter, we address two fundamental questions: how can we model quality of clusterings and how can we find multiple stable clusterings in a given data set? We make a few contributions.

First, we extend the idea of clustering stability based on Laplacian eigengap, recently used by Meilă and Shortreed in the regularized spectral learning method for similarity matrix learning [90], to multi-clustering. The intuition is that a clustering is stable if small distortions on the attribute values do not affect the discoverability of the clustering. Mathematically, we show that the larger the eigengap, the more stable the clustering.

Second, based on the notion of stability of clusterings and the underlying analysis on the Laplacian eigengap, we propose a novel subspace multi-clustering method, named multiple stable clustering (MSC) to obtain a certain number of stable clusterings in different feature subspaces. We model the problem of finding a stable clustering as an optimization problem that maximizes the eigengap. The problem is unfortunately non-convex. We propose a heuristic randomized method using iterative gradient ascent. In order to find multiple stable clusterings, we introduce to the optimization problem a constraint on the difference from the clusterings found so far. An advantage of our method comparing to the existing multi-clustering methods is that our method can provide a feature subspace (i.e., a subset of original features) to help users understand each clustering solution. Another advantage is that it does not require users to specify the number of clusters and the number of alternative clusterings in the data set, which is usually difficult for users without any guidance. Our method can heuristically determine the number of clusters, and then estimate the number of meaningful clusterings in a data set. We also discuss techniques to make MSC scalable on large-scale data. Last, we report an extensive empirical study on synthetic and real data sets that clearly demonstrates the effectiveness of our method.

This chapter is organized as follows. Section 5.2 defines the problem and models the stability of clusterings. In Section 5.3, we develop MSC, an algorithm to find multiple stable clusterings including a practical method to speedup MSC. Section 5.4 reports the results of an empirical study. Section 5.5 concludes this work.

5.2 Problem Definition

In this section, we model the stability of a clustering. Let us start with some preliminaries and the intuition.

5.2.1 Preliminaries

We consider a data set $X \in \mathbb{R}^{d \times n}$, that is, X contains n instances, each of d features. We do not assume any knowledge about how the instances in X are partitioned into groups.

A *clustering* of k clusters $C = \{c_1, c_2, \ldots, c_k\}$ is a partitioning of the instances in X such that $\cup_{m=1}^{k} c_i = X$ and $c_i \cap c_j = \emptyset$ for $1 \leq i < j \leq k$. Each c_i $(1 \leq i \leq k)$ is called a *cluster*. In clustering analysis, we are interested in the clusterings where objects in a cluster are similar and objects in different clusters are dissimilar. Here, similarity can be defined

in many different ways and thus lead to various clustering methods for different application purposes in the literature [133]. This is not a concern of our work in this chapter.

Multi-clustering aims to find multiple clusterings that are independent and thus different from each other. How to measure the degree of independency or differences among clusterings is critical in multi-clustering design. In this paper, we explore stability of clusterings as the measure. A clustering is stable if any small distortions on the attribute values do not affect the quality of the clustering. Stable clusterings are essential in the multi-clustering setting. Given an unstable clustering, a small perturbation on the data may change this clustering to another.

Take data set "donuts" in Figure 5.2 as an example, which contains 2 clusters. If we do not take the clustering stability into account, both "Clustering 1" and "Clustering 2" contain 2 clusters, and they can be considered as two different clusterings for output. However, "Clustering 2" is an unstable clustering. Because if we rotate two donuts by five degrees at the same time, "Clustering 3" can be obtained by doing a cut on $x = 0$. Obviously, "donuts" contains many unstable clusterings, such as "Clustering 2" and "Clustering 3".

Now, the technical question is how we can model the stability of a clustering.

5.2.2 Stability of a Clustering

We consider clusterings in different subspaces by exploring weighting designs of the features. We use a simplex Δ^d to denote all possible feature subspaces, which can be represented as a set of points

$$\Delta^d = \{w_1\mathbf{q}_1 + w_2\mathbf{q}_2 \cdots + w_d\mathbf{q}_d | w_m \geq 0, \sum_{m=1}^{d} w_m = 1\}$$

where $\mathbf{q}_m$ is a unit vector corresponding to the m-th feature, that is,

$$\mathbf{q}_1 = (1, 0, 0, \cdots, 0)$$

$$\mathbf{q}_2 = (0, 1, 0, \cdots, 0)$$

$$\cdots$$

$$\mathbf{q}_d = (0, 0, 0, \cdots, 1)$$

and w_m is the weight assigned to the m-th feature. Denote by $\mathbf{w} = [w_1, w_2, \cdots, w_d]$ the feature weight vector. When all weights are set to $1/d$, it is the conventional full feature space.

For each data point $\mathbf{x}_i$, we can obtain the weighted vector representation by multiplying each weight w_m with the m-th feature, that is,

$$\mathbf{x}'_i = \mathbf{w} \odot \mathbf{x}_i$$

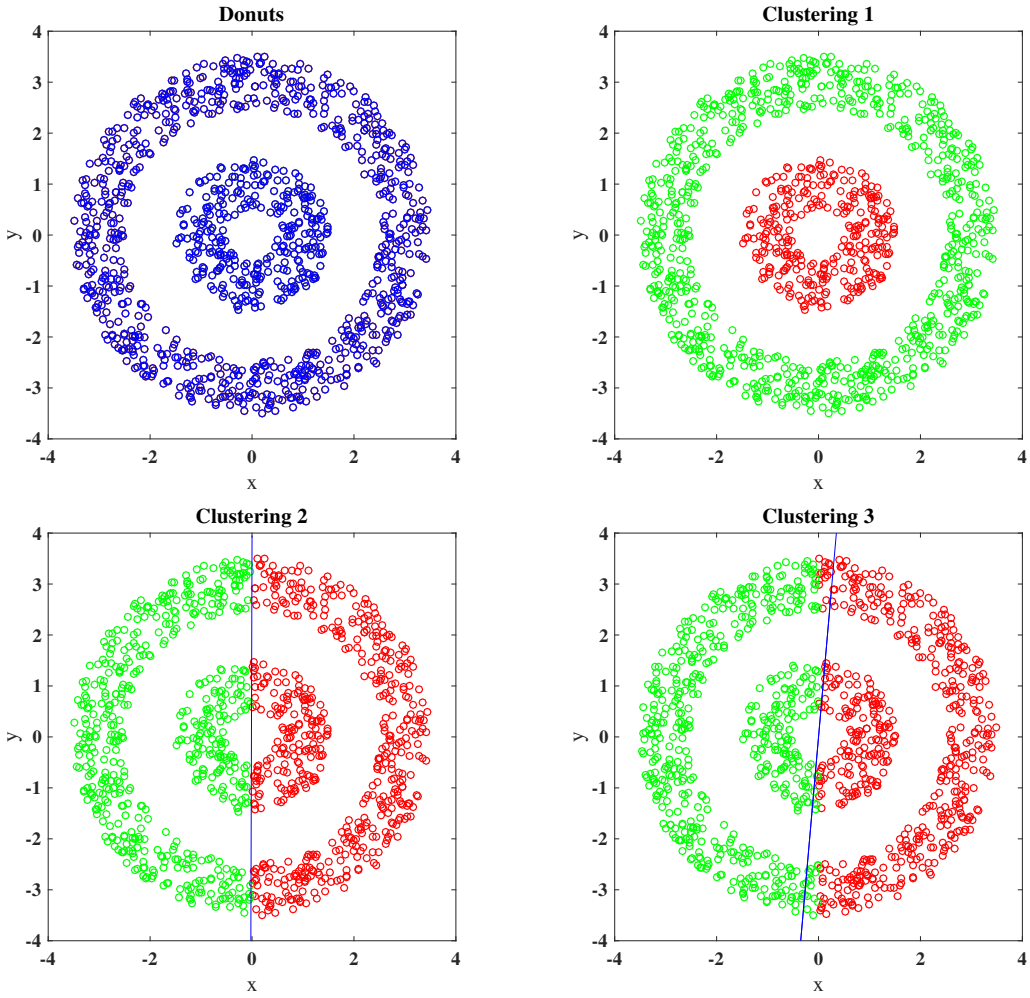


Figure 5.2: An example of Donuts data and its potential clusterings with 2 clusters (Clustering 1 is stable while Clusterings 2 and 3 are not).

where $\odot$ is the operation multiplying each element of $\mathbf{w}$ with the corresponding feature in $\mathbf{x}_i$. Then, the similarity between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ can be written as

$$S_{i,j} = e^{-\|\mathbf{x}'_i - \mathbf{x}'_j\|_2^2} \tag{5.1}$$

Now we apply spectral clustering to the similarity matrix S. Note that, although we discuss only spectral clustering here, any clustering method, such as k-means, can be applied to the similarity matrix. We calculate the *normalized Laplacian* by

$$L = D^{-1/2} S D^{-1/2} \tag{5.2}$$

where D is a diagonal matrix formed by

$$D_i = \sum_{j=1}^{n} S_{i,j}, \qquad i = 1, 2, \ldots, n$$

It is easy to verify that the eigenvectors of the normalized Laplacian here having the largest eigenvalues are identical to the eigenvectors of $I - D^{-1/2} S D^{-1/2}$ having the smallest eigenvalues, as stated in [98].

In spectral clustering, we conduct eigen-decomposition for the Laplacian matrix L and conduct clustering based on the top k eigenvectors, where k is the number of clusters we want. Now we show the main theoretical result of this paper, which indicates a nice property of stable clusterings.

Denote by $\lambda_k(L)$ the k-th largest eigenvalue of L. Essentially, we show that, if the eigengap $\lambda_k(L) - \lambda_{k+1}(L)$ is sufficiently large, a small perturbation on the similarity matrix S or on the weight vector $\mathbf{w}$ will not affect the top k eigenvectors, and thus the clusterings of k clusters obtained are stable.

Theorem 5.2.1 (Stability)**.** *Given a Laplacian matrix L, if the eigengap $\lambda_k(L) - \lambda_{k+1}(L)$ is large enough, the top k eigenvectors of $L_{perb} = L + \epsilon$ are the same as those of L, where ϵ is a symmetric perturbation matrix of small spectral norm $\|\epsilon\|_2$.*

Proof. We prove that, for any ϵ such that

$$\|\epsilon\|_2 < \frac{\lambda_k(L) - \lambda_{k+1}(L)}{2}$$

the top k eigenvectors of $L_{perb} = L + \epsilon$ are the same as those of L.

Since L is positive semi-definite, L has n non-negative real-valued eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = 0$ with the corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. Denote by $\beta = \lambda_k(L) - \lambda_{k+1}(L)$. ϵ must be in one of the following two cases.

- Case 1. ϵ is from the space spanned by the eigenvectors of L. Then, $L_{perb} = L + \epsilon$ can be represented as $\sum_{i=1}^{n} (\lambda_i \mathbf{v}_i \mathbf{v}_i^\top + \lambda_i^\epsilon \mathbf{v}_i \mathbf{v}_i^\top)$, where each $\mathbf{v}_i$ is an eigenvector of L and

$\epsilon = \sum_{i=1}^{n} \lambda_i^{\epsilon} \mathbf{v}_i \mathbf{v}_i^{\top}$. In this case, ϵ only affects the eigenvalues of L_{perb} comparing to those of L by a factor λ_i^{ϵ}, and the eigenvectors remain the same. Furthermore, since $\|\epsilon\|_2 < \beta/2$, $\max_i |\lambda_i^{\epsilon}| < \beta/2$. Therefore,

$$\lambda_k + \lambda_k^{\epsilon} - (\lambda_{k+1} + \lambda_{k+1}^{\epsilon}) > \lambda_k - \beta/2 - (\lambda_{k+1} + \beta/2) = \lambda_k - \lambda_{k+1} - \beta = 0$$

Thus, the top k eigenvectors of L_{perb} remain the same, though the ordering of the top k eigenvectors may not be identical.

- Case 2. ϵ is not solely from the space spanned by the eigenvectors of L. Then, $L_{perb} = L + \epsilon$ can be decomposed into two parts, one from the space spanned by the eigenvectors of L and the other from the orthogonal space. That is, $L + \epsilon = \sum_{i=1}^{n} (\lambda_i \mathbf{v}_i \mathbf{v}_i^{\top} + \lambda_i^{\epsilon} \mathbf{v}_i \mathbf{v}_i^{\top} + \lambda_i^{\perp} \mathbf{u}_i \mathbf{u}_i^{\top})$, where $\sum_{i=1}^{n} \lambda_i^{\perp} \mathbf{u}_i \mathbf{u}_i^{\top}$ is part of ϵ from the orthogonal space. The first part is similar to the first case. For the second part, since $\|\epsilon\|_2 < \beta/2$, $\max_i \lambda_i^{\perp} <= \max_i |\lambda_i^{\perp}| < \beta/2$. Therefore,

$$\lambda_k = \lambda_{k+1} + \beta \geq \beta > \max_i \lambda_i^{\perp}$$

Thus, the top k eigenvectors of L_{perb} remain the same as L, though the ordering of the top k eigenvectors may, again, not be identical.

□

Remark 5.2.1. *In spectral clustering, if the top k eigenvectors are the same for L and L_{perb}, the clusterings with k clusters based on the same eigenvectors are the same.*

According to Theorem 5.2.1 and Remark 5.2.1, the larger the eigengap between the k-th and the $(k+1)$-th eigenvalues of L, the more stable the clustering with k clusters obtained from L.

5.3 The Proposed Method

In this section, we first propose how to find one stable clustering and determine its number of clusters. Then, we present our novel subspace multi-clustering method: MSC.

5.3.1 Finding a Stable Clustering

According to Theorem 5.2.1, given the number of clusters k, the most stable clustering can be obtained by maximizing the eigengap, that is,

$$\arg \max_{\mathbf{w} \in \Delta^d} \lambda_k(L) - \lambda_{k+1}(L) \tag{5.3}$$

where n is the number of instances in the data set, and the *stable weight vector* $\mathbf{w}$ is searched in the simplex Δ^d. Note that the simplex constraint has a good sparsity property – it automatically eliminates those features of too low weights [107, 56]. This property is desirable since sparse feature selection has been demonstrated effective by many previous studies [121, 141, 14, 131].

Although Equation 5.3 models the most stable clustering with k clusters precisely, apparently the optimization problem is non-convex. Thus, it is computationally expensive or even prohibitive to find the optimal solution.

Moreover, in real-world applications, the number of clusters hidden in data is usually unknown in advance. We determine the number of clusters by maximizing the eigengap, that is,

$$\arg\max_k \lambda_k(L^*) - \lambda_{k+1}(L^*) \tag{5.4}$$

where $2 \leq k \leq n-1$, and L^* is the Laplacian matrix constructed by Equations 5.1 and 5.2 when $\mathbf{w} = [1/d, 1/d, \cdots, 1/d]$. Specifically, we will only study the case of k that the original data can provide the most stable clustering. This k can help discover at least one stable clustering whose perturbation tolerance is comparatively large. Moreover, if all stable clusterings with k clusters can be discovered, all other stable clusterings with different number of clusters are either combing or splitting clusters from the clusterings obtained.

Now given the number of clusters k targeted, as a heuristic solution, we can initialize $\mathbf{w}$ by setting

$$\mathbf{w}_0 = [1/d, 1/d, \cdots, 1/d]$$

in the simplex Δ^d and solve the optimization problem in Equation 5.3 by iterative gradient ascent. Specifically, in the t-th iteration ($t \geq 1$) of gradient ascent, we set

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \eta \mathbf{G}$$

where the m-th element of vector $\mathbf{G}$ ($1 \leq m \leq d$) is

$$G_m = \langle \mathbf{v}_k \mathbf{v}_k^\top, \frac{\partial L}{\partial w_m} \rangle - \langle \mathbf{v}_{k+1} \mathbf{v}_{k+1}^\top, \frac{\partial L}{\partial w_m} \rangle \tag{5.5}$$

and $\mathbf{v}_k$ is the k-th eigenvector. The derivation of the gradient is calculated as follows. Since

$$L\mathbf{v}_k = \lambda_k(L)\mathbf{v}_k$$

we have

$$\mathbf{v}_k^\top L \mathbf{v}_k = \mathbf{v}_k^\top \lambda_k(L) \mathbf{v}_k$$

Therefore,

$$\lambda_k(L) = \frac{\mathbf{v}_k^\top L \mathbf{v}_k}{\|\mathbf{v}_k\|_2^2} = \mathbf{v}_k^\top L \mathbf{v}_k = \langle \mathbf{v}_k \mathbf{v}_k^\top, L \rangle$$

Consequently, we have

$$\frac{\partial \lambda_k(L)}{\partial w_m} = \langle \mathbf{v}_k \mathbf{v}_k^\top, \frac{\partial L}{\partial w_m} \rangle$$

where

$$\frac{\partial L}{\partial w_m} = D^{-3/2} \frac{\partial D}{\partial w_m} S D^{-1/2} + D^{-1/2} \frac{\partial S}{\partial w_m} D^{-1/2} + D^{-1/2} S D^{-3/2} \frac{\partial D}{\partial w_m}$$

$$\frac{\partial D_i}{\partial w_m} = \sum_{j=1}^{n} \frac{\partial S_{i,j}}{\partial w_m}, \quad i = 1, 2, \ldots, n$$

and

$$\frac{\partial S_{i,j}}{\partial w_m} = -2 S_{i,j} (x_{i,m} - x_{j,m})^2 w_m$$

since

$$S_{i,j} = e^{-\|\mathbf{x}'_i - \mathbf{x}'_j\|_2^2} = e^{-\sum_m w_m^2 (x_{i,m} - x_{j,m})^2}$$

To constrain $\mathbf{w}$ within the simplex Δ^d, in each gradient ascent step, we adopt the projection algorithm proposed by Kyrillidis *et al.* [75]. Concretely, we project $\mathbf{w}_t$ obtained in the t-th iteration onto the simplex by

$$(\mathcal{P}_{1^+}(\mathbf{w}_t))_m = [(\mathbf{w}_t)_m - \alpha]_+ \tag{5.6}$$

where α is a threshold that is set to

$$\alpha = \frac{1}{\rho} (\sum_{i=1}^{\rho} (\mathbf{w}_t)_m - 1)$$

where

$$\rho = \max\{m : (\mathbf{w}_t)_m > \frac{1}{m} (\sum_{i=1}^{m} (\mathbf{w}_t)_m - 1)\}$$

Here, α and ρ are calculated by sorting the elements in $\mathbf{w}_t$ in descending order. Instead of directly calculating the gradient under the simplex constraint, we first calculate the gradient without the constraint, conduct the gradient ascent, and then project $\mathbf{w}$ back to the simplex. Kyrillidis *et al.* [75] proved that the projection algorithm still obtains the same $\mathbf{w}$ as considering the simplex constraint in the gradient calculation.

Algorithm 7 shows the pseudo-code of finding a stable state (FSS), which first determines the number of clusters we can discover on the data set, and then finds a local optima for the problem defined in Equation 5.3. Note that Step 4 conducts full eigen-decomposition for L^* because almost all eigenvalues are required to determine the number of clusters (maximizing the eigengap), while Step 15 does partial eigen-decomposition on L because only the top $k+1$ eigenvalues and their corresponding eigenvectors are useful for the gradient calculation.

Algorithm 7 Finding a Stable State (FSS)

Input: data set $X \in \mathbb{R}^{d \times n}$, the number of iterations T, and step size η
Initialize weight vector $\mathbf{w}_0 = [1/d, 1/d, \cdots, 1/d]$ in the simplex Δ^d
Compute L^* by Equations 5.1 and 5.2
Conduct eigen-decomposition for L^*
for $k = 2$ **to** $n - 1$ **do**
    Calculate $\lambda_k(L^*) - \lambda_{k+1}(L^*)$
end for
Set $k = \arg\max_k \lambda_k(L^*) - \lambda_{k+1}(L^*)$
for $t = 1$ **to** T **do**
    Compute $\mathbf{G}$ using Equation 5.5
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta\mathbf{G}$
    Project $\mathbf{w}_t$ onto the simplex Δ^d using $\mathcal{P}_{1^+}$ (Equation 5.6)
    Compute S according to Equation 5.1
    Calculate the normalized Laplacian L according to Equation 5.2
    Conduct partial eigen-decomposition for L
end for
return $\mathbf{w}_T$

5.3.2 Finding Multiple Stable Clusterings

In this section, we develop MSC, an algorithm to find multiple stable clusterings. We first present MSC, and then discuss the techniques to speed up the algorithm.

Finding Multiple Stable States

To find various stable clusterings, we need to search all stable states in the simplex. Although random initialization can give a good start point, two random initialization values may converge to the same local optimal state. To overcome this problem, we introduce a constraint on the difference between the current weight vector and those previously obtained.

Let W be the set of weight vectors obtained so far. We enhance the optimization problem in Equation 5.3 to

$$\arg\max_{\mathbf{w} \in \Delta^d} \lambda_k(L) - \lambda_{k+1}(L) + \frac{\delta}{2} \frac{1}{|W|} \sum_{\mathbf{w}_p \in W} \|\mathbf{w} - \mathbf{w}_p\|_2^2 \tag{5.7}$$

In other words, we want to maximize the sum of the distances between the current weight vector and those weight vectors obtained previously so as to keep the current clustering far away from all previous ones. Although we choose the simplest way to constrain the difference between weight vectors, some alternative ways can be adopted and convex functions such as Kullback-Leibler divergence [73] are more desired. Convex functions will not add too much computational burden to our already non-convex optimization problem. Here, $\delta \geq$

0 is a parameter controlling the tradeoff between the maximization of eigengap and the dissimilarity of the new stable state. Please note that, the difference between clusterings can be implicitly constrained by the difference between weight vectors due to the stability. Specifically, if two different stable weight vectors lead to two different clusterings, they must be sufficiently different or independent. If two different stable weight vectors provide similar partitions, they imply the same clustering meaningful for users, especially in multi-view clustering [44, 38], which aims to combine results from different views to generate a consensus clustering.

Algorithm 8 describes how to find a new stable state (weight vector), given the set W of stable states found so far. It solves the optimization problem in Equation 5.7 using gradient ascent. The derivation of the regularization term can be easily calculated as

$$\frac{\partial(\frac{\delta}{2}\frac{1}{|W|}\sum_{\mathbf{w}_p\in W}\|\mathbf{w}-\mathbf{w}_p\|_2^2)}{\partial w_m} = \frac{\delta}{|W|}\sum_{\mathbf{w}_p\in W}(w_m - w_{p,m})$$

as stated in Step 8 in Algorithm 8. The regularization term incurs only very light computational cost.

Algorithm 8 Finding Alternative Stable State (FASS)

1: **Input:** data set $X \in \mathbb{R}^{d\times n}$, the number of clusters k, the number of iterations T, step size η, the set of previously found stable states W, and the tradeoff parameter δ
2: Randomly initialize weight vector $\mathbf{w}_0$ in the simplex
3: **for** $t = 1$ **to** T **do**
4: Compute S by Equation 5.1
5: Calculate the normalized Laplacian L by Equation 5.2
6: Conduct eigen-decomposition for L
7: Compute $\mathbf{G}$ by Equation 5.5
8: $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta(\mathbf{G} + \delta\frac{1}{|W|}\sum_{\mathbf{w}_p\in W}(\mathbf{w}_{t-1} - \mathbf{w}_p))$
9: Project $\mathbf{w}_t$ onto the simplex by $\mathcal{P}_{1^+}$
10: **end for**
11: **return** $\mathbf{w}_T$

We can run Algorithm 7 to find the first stable state, and then run Algorithm 8 repeatedly with different initialization values to find more stable states. Unfortunately, so far there is no theoretical guarantee on finding all significantly different clusterings in Equation 5.7. Heuristically, if Algorithm 8 does not lead to any new stable state in the last l runs, then we can terminate the process, where $l > 0$ is a parameter.

After gathering a set of stable weight vectors, we can compute the similarity matrix S for each stable weight vector $\mathbf{w}$ and apply spectral clustering to obtain the corresponding stable clustering. Each stable clustering obtained as such has a corresponding sparse feature subspace $\mathbf{w}$ for user understanding. The whole algorithm is summarized in Algorithm 9.

Algorithm 9 Multiple Stable Clustering (MSC)

Input: data set $X \in \mathbb{R}^{d \times n}$, the number of iterations T, step size η, the tradeoff parameter δ, and the stopping threshold τ
Initialize the stable state set $W = \emptyset$ and clustering solution set $R = \emptyset$
Determine the number of clusters k and obtain the 1st stable state $\mathbf{w}$ using Alg. 7
$W = W \cup \mathbf{w}$
repeat
    Obtain a new stable state $\mathbf{w}$ using Alg. 8
    $W = W \cup \mathbf{w}$
until $\min_{\mathbf{w}_i, \mathbf{w}_j \in W, i \neq j} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2 \leq \tau$
Delete the $\mathbf{w} \in W$ that is last obtained
for each $\mathbf{w} \in W$ **do**
    Compute S by Equation 5.1
    Calculate the normalized Laplacian L by Equation 5.2
    Conduct partial eigen-decomposition for L
    Get the top k eigenvectors
    Obtain the clustering C using k-means on the eigenvectors
    $R = R \cup \{(\mathbf{w}, C)\}$
end for
return R

A unique advantage of our method is that our method can provide feature subspaces for clustering understanding, and it does not need users to specify the number of clusters and the number of clusterings in the data set, which is usually difficult for users without any guidance. The optimization on both the eigengap and the weight vector dissimilarity leads to stable clusterings discovered in the iterations. The mining method terminates when no new stable and substantially different clustering can be found. As demonstrated in the experimental results in Section 5.4, our method can heuristically estimate the number of meaningful clusterings in a data set, which is infeasible in the existing multi-clustering methods.

Speedup

To find a stable state vector $\mathbf{w}$, we conduct gradient ascent for T iterations. In each iteration, we have to compute the full similarity matrix S based on the weight vector obtained in the previous iteration. Thus, the time complexity is $\mathcal{O}(n^2)$. We also have to conduct eigen-decomposition for the Laplacian matrix L, whose computational cost is $\mathcal{O}(n^3)$. To compute the gradient, we need to compute $\frac{\partial L}{\partial w_m}$ for each element in $\mathbf{w}$, whose computational cost is $\mathcal{O}(n^2)$. All other computation steps are cheaper comparing to these two. This heavy computational cost limits the application of MSC to large-scale data sets that have many instances.

Algorithm 10 Speedy MSC (SMSC)

1: **Input:** data set $X \in \mathbb{R}^{d\times n}$, the number of iterations T, step size η, the tradeoff parameter δ, the stopping threshold τ, and the number of representatives r
2: Initialize the stable state set $W = \emptyset$ and clustering solution set $C = \emptyset$
3: Perform k-means clustering method on X, and generate r clusters
4: Generate $X^{(r)}$ that are nearest to each cluster center
5: Determine the number of clusters k and obtain the 1st stable state $\mathbf{w}$ using Algorithm 7 based on $X^{(r)}$
6: $W = W \cup \mathbf{w}$
7: **repeat**
8: Obtain a new stable state $\mathbf{w}$ using Algorithm 8 based on $X^{(r)}$
9: $W = W \cup \mathbf{w}$
10: **until** $\min_{\mathbf{w}_i,\mathbf{w}_j\in W, i\neq j} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2 \leq \tau$
11: Delete the $\mathbf{w} \in W$ that is last obtained
12: **for each** $\mathbf{w} \in W$ **do**
13: Conduct spectral clustering on $S^{(r)}$ computed by Equation 5.1 on $X^{(r)}$
14: Obtain clustering c by assigning each point in X to the nearest center within subspace $\mathbf{w}$
15: $C = C \cup \{(\mathbf{w}, c)\}$
16: **end for**
17: **return** C

Although some techniques, such as Nyström method [128, 39] and column sampling [81], can be adopted to approximate the similarity matrix and the corresponding top eigenvectors, those methods cannot be easily extended to approximate the full eigen-decomposition on L^* and the gradient calculation of $\frac{\partial L}{\partial w_m}$. Therefore, we adopt a strategy that can benefit all above heavy steps.

We select an affordable size r of representative data points from the original data as some existing fast spectral clustering methods [135, 25] do. Here we can randomly selecting r data points or apply the k-means clustering method on the original large-scale data and generate r clusters. Then, we choose the data points nearest to the cluster centers as the representative data points $X^{(r)} \in R^{d\times r}$. The stable weight vectors are produced by MSC method based on $X^{(r)}$. Thereafter, we extend the k-means-based approximate spectral clustering [135] to conduct spectral clustering for each obtained subspace $\mathbf{w}$. Specifically, we first apply the standard spectral clustering on those representatives within the subspace discovered by $\mathbf{w}$. Then, the cluster memberships for the rest data points are determined by assigning them to the nearest cluster centers within subspace $\mathbf{w}$.

In summary, Algorithm 10 shows how we can speed up the proposed MSC algorithm. Please note that, Steps 3 and 4 can be replaced by randomly select r data points as needed. It can be easily observed that the computational complexity of generating r representatives using k-means is $\mathcal{O}(rnt)$, where t is the number of iterations set by k-means and is usually a constant as the maximum number of iterations allowed. The time complexity can be further

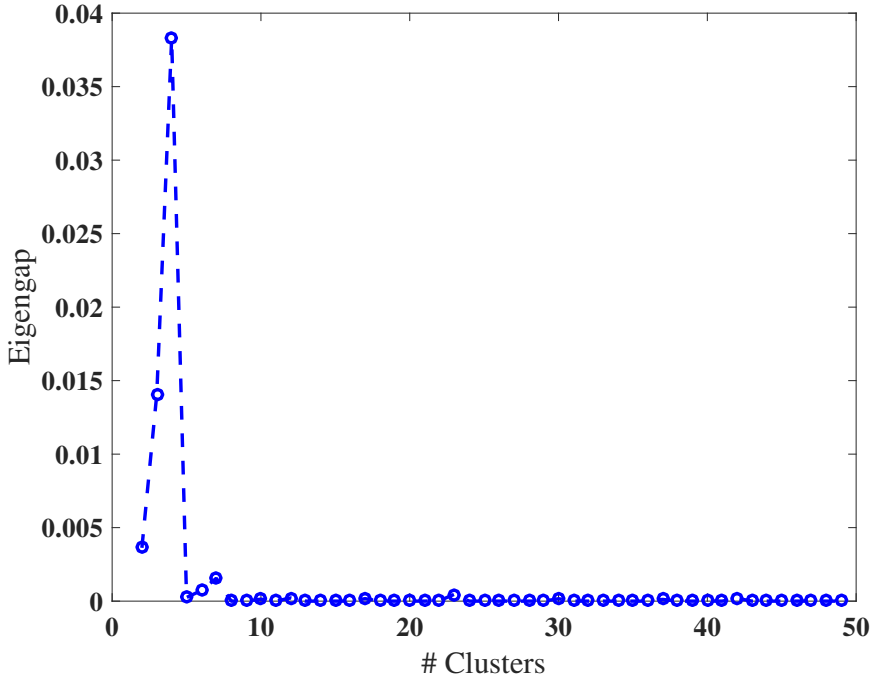


Figure 5.3: Eigengap distribution over #Clusters on the synthetic data $\{0,1\}^{50\times 3}$.

reduced to $\mathcal{O}(nt)$ by applying the approximate k-means algorithm [74]. Meanwhile, the time complexity of calculating the similarity matrix, conducting eigen-decomposition, and computing the gradient can be reduced to $\mathcal{O}(r^2)$, $\mathcal{O}(r^3)$, and $\mathcal{O}(r^2)$, respectively. Finally, the step to obtain clustering solutions incur a computational cost of $\mathcal{O}(r^3+nr)$. Therefore, the overall computational complexity of SMSC is $\mathcal{O}(nr+r^3)$. This makes our MSC method applicable on large-scale data.

5.4 Experimental Results

To test our proposed MSC method, we conduct an empirical study on both synthetic and real data sets. We report the results in this section.

5.4.1 A Case Study Using Synthetic Data

In this subsection, we report a comprehensive empirical study on synthetic data. We randomly generate 50 data points with 3 binary features, that is, the synthetic data $X \in \{0,1\}^{50\times 3}$. By checking the eigengap distribution over different number of clusters on the original data in Figure 5.3, we can determine that the number of clusters desired on this data set is $k = 4$.

Settings

We randomly initialize $\mathbf{w}_0$ in the simplex Δ^3 during the calling of Algorithms 7 and 8. We set the number of iterations $T = 30$. The step size starts with $\eta = 1$, and is decreased according to $1 \to 1/2 \to 1/3 \to \cdots$ when necessary, guided by the norm of the gradient during the process. MSC stops when no more sufficiently different stable state can be discovered. We consider two weight vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ are sufficiently different when the difference between any corresponding element is larger than 0.05 (note that all weight vectors are within the simplex), which means if $\|\mathbf{w}_i - \mathbf{w}_j\|_2^2 \leq 0.0025d$, $\mathbf{w}_i$ and $\mathbf{w}_j$ are considered similar and thus the stopping threshold is set as $\tau = 0.0025d$. Since we have $d = 3$ on this synthetic data, $\tau = 0.0075$. If in $l = 3$ successive calling of Algorithm 8, no sufficiently different stable state can be discovered, MSC stops.

The Tradeoff Effect Controlled by Parameter δ.

Next we study how the tradeoff parameter δ can affect the performance of MSC. Due to the reason that the eigengap is between 0 and 1 exclusive (usually less than 0.1), and for any i, j, $0 \leq \|\mathbf{w}_i - \mathbf{w}_j\|_2^2 \leq 2$, we show the effect of δ by setting it to 0.0001, 0.001,0.01, 0.1, and 1. We find that if we pay too much attention to the dissimilarities of weight vectors, the stability of the clustering is sacrificed. It leads to unstable clusterings that are not meaningful for users. For example, when $\delta = 1$, we obtain four weight vectors, that is, $\mathbf{w}_1 = [0.4881, 0.5119, 0]$, $\mathbf{w}_2 = [1, 0, 0]$, $\mathbf{w}_3 = [0, 0, 1]$, and $\mathbf{w}_4 = [0, 1, 0]$, where the first weight vector is obtained only considering the eigengap. It is obvious that $\mathbf{w}_2$ to $\mathbf{w}_4$ cannot provide stable clustering for $k = 4$, whose low stability can also be observed from Figure 5.4. We can observe from Figure 5.4 that states 2 to 4 (i.e., $\mathbf{w}_2$, $\mathbf{w}_3$, and $\mathbf{w}_4$) converge to unstable clusterings with too small eigengaps, while only state 1 ($\mathbf{w}_1$) that pays attention only on the eigengap can converge to a stable clustering with relatively large eigengap. When $\delta = 0.1$, four out of ten states discovered are unstable as $\mathbf{w}_2$ to $\mathbf{w}_4$.

When we set $\delta = 0.01$, 0.001, or 0.0001, MSC can discover the same set of four stable states as shown in Table 5.1. Figure 5.5 shows how the eigengap of each stable state converges as the number of iterations increases. It can be easily observed that gradient ascent converges quickly within 10 iterations. By comparing each stable clustering obtained, we find that clustering solutions obtained in the discovered subspaces exactly match the ground truth as expected. Meanwhile, MSC discovers three stable states (i.e., $\mathbf{w}_1$, $\mathbf{w}_3$, and $\mathbf{w}_4$) that each is composed by almost equivalent weighting on two features. It is as expected since any combination of two features on this synthetic data can identify 4 clusters as shown by the cluster patterns in Table 5.1. On the other hand, $\mathbf{w}_2$ equally weights three features, which tells 4 clusters in another interesting independent way as shown in Table 5.1. Therefore, to determine the tradeoff parameter δ, we propose to check the eigengap when determining the number of clusters. Generally, if eigengap is around 10^{-m}, δ can be set to

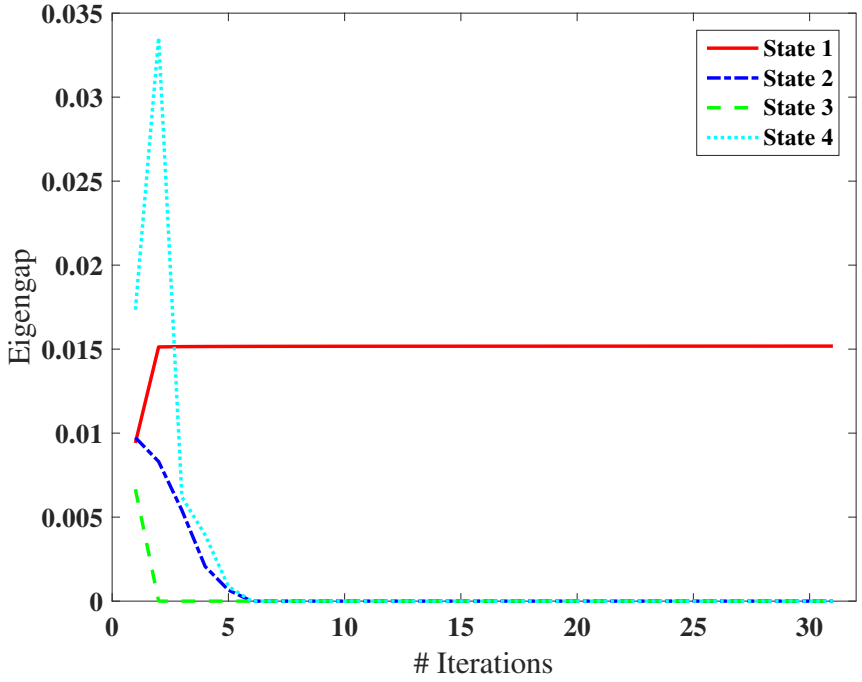


Figure 5.4: The changes of eigengap in iterations on the synthetic data $\{0,1\}^{50\times 3}$ when $\delta = 1$.

Table 5.1: Results on the synthetic data $\{0,1\}^{50\times 3}$.

Stable States $\mathbf{w}$	Eigengap	Cluster Patterns
$\mathbf{w}_1 = [.4881, .5119, .0000]$	.0152	$(00*, 11*, 10*, 01*)$
$\mathbf{w}_2 = [.3208, .3348, .3444]$	.0414	$(11*, *00, 01*, *01)$
$\mathbf{w}_3 = [.5051, .0000, .4949]$	.0137	$(0*1, 0*0, 1*0, 1*1)$
$\mathbf{w}_4 = [.0000, .4900, .5100]$	.0129	$(*01, *10, *00, *11)$

$10^{-(m+1)}$. Apparently, setting the parameter δ is easier for users comparing to setting the number of clusterings if no guidance is given.

Performance Comparison

To better interpret the relationships between the stable states and clustering solutions, and compare the clustering performance, we further set the number of clusters $k = 2$. According to the above study in Section 5.4.1, we set $\delta = 0.001$.

Baselines. To the best of our knowledge, existing unsupervised subspace multi-clustering methods are mainly *subspace clustering* methods. Although *subspace clustering* methods can provide a subset of features for each clustering, a clustering in a feature subspace often covers only a subset of data points which is totally different from our setting. At the

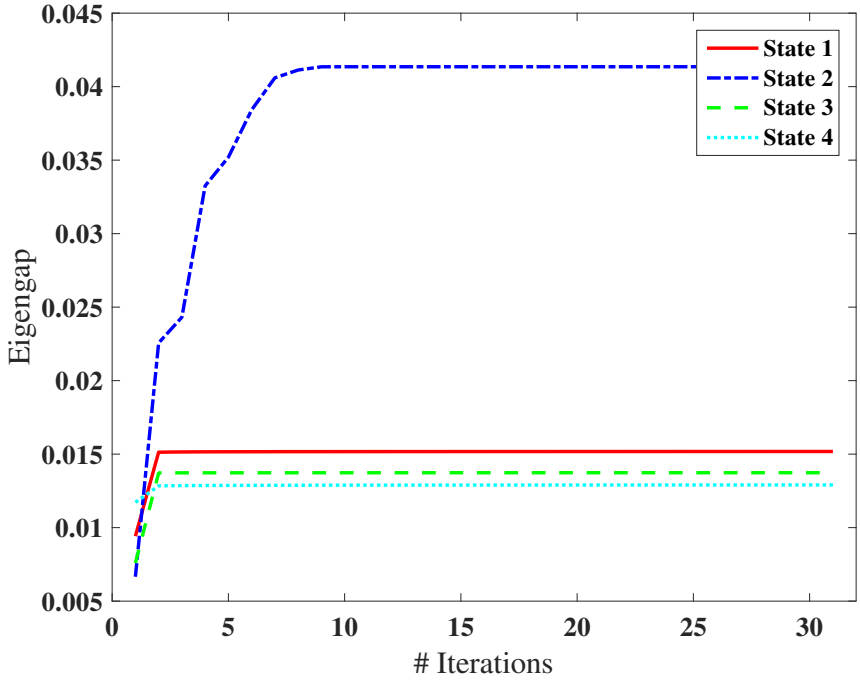


Figure 5.5: The changes of eigengap in iterations on the synthetic data $\{0,1\}^{50\times 3}$ when $\delta = 0.001$.

same time, existing semi-supervised subspace multi-clustering methods are usually based on feature transformation or dimensionality reduction, and thus their subspaces are not intuitively understandable as feature selection in our method. We thus focus on comparing the clusterings and the ground truth here.

Moreover, the existing semi-supervised multi-clustering methods, such as COALA [8], often require a given clustering. Although one can use some traditional clustering method, such as k-means clustering and spectral clustering method, to generate a clustering solution as an input, many methods may still provide only one alternative clustering. MSC can discover a certain number of clusterings. It is unfair and difficult to compare methods that generate different numbers of clusterings. Therefore, we compare with the most effective unsupervised multi-clustering method, meta-clustering [21], in which the number of clusterings generated can be specified.

Specifically, we use k-means clustering with different initializations to generate 50 clusterings at first. Then, we use the Rand Index (RI) [60] to measure the similarities between clusterings. Spectral clustering [123] is further applied on the similarity matrix to group those 50 clusterings. Finally, we choose the clustering nearest to each group center as the representative clustering to output. The number of groups for clusterings is an input item that can be varied for the purpose of comparison.

In addition, we compare the clusterings produced by MSC with the results from two well known clustering methods: k-means [85] and normalized spectral clustering [123], denoted by "k-means" and "Spectral" hereafter, respectively.

When applying above mentioned baselines, we project the data to subspace

$$\mathbf{w} = [1/d, 1/d, \cdots, 1/d]$$

because those methods cannot find the weights of features automatically.

We report the matching quality between two clusterings using five popularly used measures: Normalized Mutual Information (NMI), Rand Index (RI), Adjusted Rand Index (AR), Mirkin's Index Distance (MI), and Hubert's Index (HI) [60]. The smaller MI, the more similar the two clusterings under comparison. For the other four measures, the larger the values, the more similar the two clusterings.

Results. Due to the binary features, each feature solely can group the data points into 2 clusters, where the feature itself is a subspace. Therefore, we have 3 clusterings in the ground truth denoted by "Clustering 1", "Clustering 2", and "Clustering 3", respectively. MSC produces 3 stable clusterings as shown in Table 5.2.

The clustering produced by k-means matches Clustering 1 in the ground truth, manifested by the first feature, though all attributes are equivalently weighted in k-means. However, k-means can only produce a single partitioning, and cannot produce the other two clusterings in the ground truth, which is also the case for the normalized spectral clustering method. The spectral clustering method produces a clustering solution relatively consistent with but not perfectly matching Clustering 2 in the ground truth, manifested by the second feature.

As shown in Table 5.2, MSC finds the three clusterings perfectly in the ground truth. Moreover, MSC also discovers the three stable states (weight vectors) exactly as expected – each feature providing a perfect feature subspace for a 2-way clustering, as shown in the last column of Table 5.2. These weight vectors obtained confirm the sparse property of the simplex constraint. The eigengaps for those three stable states are much larger than the eigengap in the clustering produced by the spectral clustering method. This result clearly demonstrates that, by maximizing the eigengap, MSC can find stable clusterings.

Although our extension of meta-clustering also discovers three clusterings on this data set, where each one exactly matches one of the ground truth, the meta-clustering method first generates 50 clusterings, which is computationally expensive. Moreover, generally it is hard to determine how many groups of clusterings should be generated in advance. More importantly, those clusterings cannot provide the additional subspace information of each clustering for user understanding.

Table 5.2: Clusterings on the Synthetic Data $\{0,1\}^{50\times 3}$ when $k = 2$. The best cases are highlighted in bold. The AR values are not provided here because the denominator in the AR formula is zero in all cases here.

Clustering produced by methods	Clustering in ground truth	NMI	RI	AR	MI	HI	Eigengap	Weight vector **w**
k-means	Clustering 1	**1.000**	**1.000**	-	**.0000**	**1.000**	-	-
	Clustering 2	.0043	.4783	-	.5217	-.043		
	Clustering 3	.0039	.4737	-	.5263	-.053		
Spectral	Clustering 1	.0283	.5569	-	.4431	.1138	.0039	-
	Clustering 2	**.7263**	**.7628**	-	**.2372**	**.5257**		
	Clustering 3	.0718	.6491	-	.3509	.2983		
Meta-clustering 1	Clustering 1	.0039	.5292	-	.4708	.0585	-	-
	Clustering 2	.0154	.5573	-	.4427	.1146		
	Clustering 3	**1.000**	**1.000**	-	**.0000**	**1.000**		
Meta-clustering 2	Clustering 1	**1.000**	**1.000**	-	**.0000**	**1.000**	-	-
	Clustering 2	.0043	.4783	-	.5217	-.044		
	Clustering 3	.0039	.4737	-	.5263	-.053		
Meta-clustering 3	Clustering 1	.0043	.4923	-	.5077	-.015	-	-
	Clustering 2	**1.000**	**1.000**	-	**.0000**	**1.000**		
	Clustering 3	.0154	.5088	-	.4912	.0175		
MSC: Clustering 1	Clustering 1	**1.000**	**1.000**	-	**.0000**	**1.000**	.2951	[1.000, .0000, .0000]
	Clustering 2	.0043	.4923	-	.5077	-.015		
	Clustering 3	.0039	.5292	-	.4708	.0585		
MSC: Clustering 2	Clustering 1	.0043	.4783	-	.5217	-.044	.2931	[.0000, 1.000, .0000]
	Clustering 2	**1.000**	**1.000**	-	**.0000**	**1.000**		
	Clustering 3	.0154	.5573	-	.4427	.1146		
MSC: Clustering 3	Clustering 1	.0039	.4737	-	.5263	-.053	.2711	[.0000, .0000, 1.000]
	Clustering 2	.0154	.5088	-	.4105	.4912		
	Clustering 3	**1.000**	**1.000**	-	**.0000**	**1.000**		

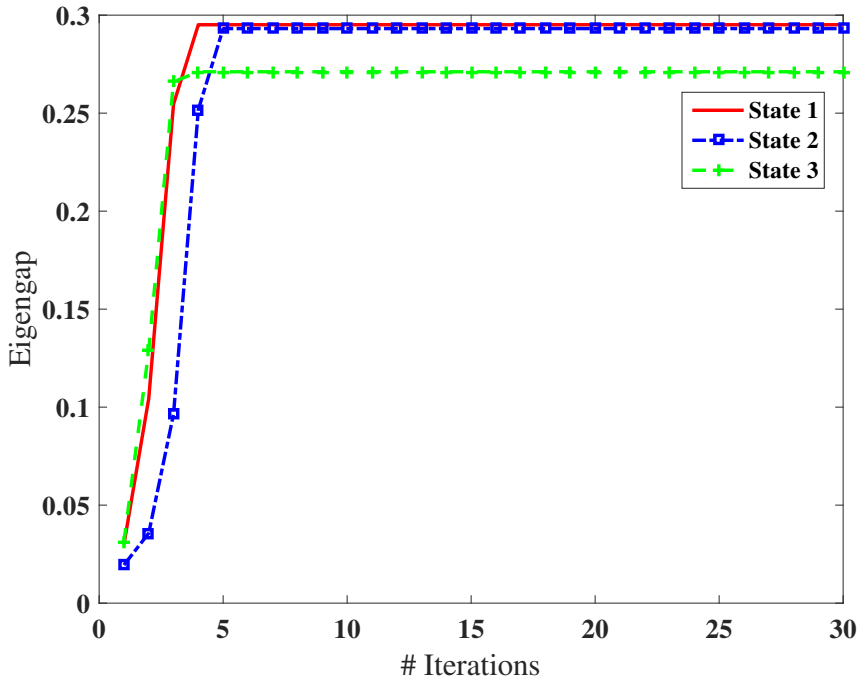


Figure 5.6: The changes of eigengap in iterations on the synthetic data $\{0,1\}^{50\times 3}$ when $k = 2$.

Figure 5.6 shows how the eigengap of each clustering solution converges as the number of iterations increases. Our gradient ascent method converges very quickly. This result also demonstrates the effectiveness of our method.

Results on Large Data Sets

We show that our scalable version of MSC, SMSC, is capable of processing large-scale data. We randomly generate 200,000 data points as $X \in \{0,1\}^{200{,}000\times 3}$. We set the number of representatives $r = 2000$, that is, 1% of the original data. Instead of using k-means clustering to generate r representatives, we randomly select r data points from X as the representative data points.

This experiment is conducted on a desktop computer with an Intel Core i7-2600 3.40GHz CPU and running Windows Operating System. SMSC stops after calling Algorithm 8 10 times. It discovers 4 stable states as shown in Table 5.3, where each clustering solution exactly matches a specific cluster pattern. Although the cluster pattern for $\mathbf{w}_2$ is not the same as above findings, it is also a different partition comparing to other three clusterings.

Table 5.4 shows the CPU time required for each essential step in SMSC. It indicates that SMSC is capable on large-scale data.

Table 5.3: Results on the synthetic data $\{0,1\}^{200,000\times 3}$

Stable States $\mathbf{w}$	Cluster Patterns
$\mathbf{w}_1 = [.4869, .5131, .0000]$	$(00*, 10*, 11*, 01*)$
$\mathbf{w}_2 = [.3356, .3293, .3251]$	$(110, *00, 1**-110, 00*)$
$\mathbf{w}_3 = [.4993, .0000, .5007]$	$(0*0, 1*1, 0*1, 1*0)$
$\mathbf{w}_4 = [.0000, .4861, .5139]$	$(*00, *11, *01, *10)$

Table 5.4: CPU time on the synthetic data $\{0,1\}^{200,000\times 3}$ when $r = 2,000$.

Steps	CPU time (sec.)
Select representatives	.0156
Discover stable states	2.7333×10^3
Clustering	31.5590

5.4.2 A Case Study on an Image Data Set

Besides synthetic data, we conduct a case study on a set of images of three types of fruits, namely apples, bananas, and grapes, in different colors, namely red, yellow, and green for apples, yellow and green for bananas, and red and green for grapes. There are two different clusterings in the ground truth: the clustering by fruit category, denoted by Clustering-by-Category, and the clustering by color, denoted by Clustering-by-Color.

Data Preparation

We collect from Internet 15 images for each sub-group, that is, red apples, yellow apples, green apples, yellow bananas, and so on. In total, there are $15 \times 7 = 105$ images in the data set, 45 about apples, 30 about bananas, and another 30 about grapes. Orthogonally, there are 30 images about red fruits, 30 about yellow fruits, and 45 about green fruits.

For each image, we first partition it into blocks of 4 pixels by 4 pixels each, and then extract the color and texture features for each block. For the color features, we transform the RGB images into the HSI color space, which is appropriate for object recognition as suggested by the study in [113]. Thereafter, the color feature of each block is represented by the average value of each channel. Thus, each block has three color features, corresponding to the channels of Hue, Saturation, and Intensity, respectively. For the texture features, we employ the one-level two-dimensional Daubechies-4 wavelet transformation [33] to decompose each block to four frequency bands. Each band has four parameters $\{c_{k,l}, c_{k,l+1}, c_{k+1,l}, c_{k+1,l+1}\}$. We then calculate the features by

$$f = (\frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1} c_{k+i,l+j}^2)^{\frac{1}{2}}$$

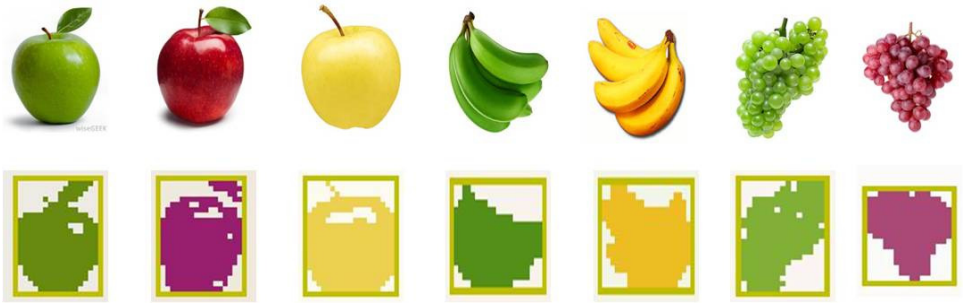


Figure 5.7: Exemplar images in the fruit data set.

with the bands about activities in horizontal, vertical and diagonal directions [126], which also lead to three features for a block.

For an image with multiple six dimensional blocks, we apply k-means for segmentation as recommended by the study in [126]. After segmentation, we then use raster scanning [113] to get all connected components. A region whose number of blocks is below a threshold is set to be background. We erase background regions. Then, each fruit region is represented by the average color features and the normalized inertia as the shape features [126]

$$l(H, \gamma) = \frac{\sum_x \in H \|x - \hat{x}\|^\gamma}{V(H)^{1+\gamma/k}}$$

where H is the fruit region, $V(H)$ is the number of blocks in the region and $\hat{x}$ is the center of the region. We respectively set $\gamma = 1, 2, 3$ to calculate the shape features, and thus the total number of features for each fruit region is 6. Figure 5.7 shows some exemplar images. The first row is the original images collected from Internet sources, and the second row is the recognized fruit regions bounded and represented by the average RGB colors.

Results

MSC determines $k = 3$ based on the original full feature space. The eigengap distribution over the number of clusters is shown in Figure 5.8. According to the parameter studied in the above section, we set $\tau = 0.0025d = 0.015$ and $\delta = 0.0001$.

Table 5.5 shows the results of k-means, spectral clustering, meta-clustering, and MSC on this image data set. Interestingly, the clustering results produced by k-means and spectral clustering are almost identical, which are more consistent with the ground truth clustering by color than that by category.

In this data, users may expect the method to generate only two clusterings. This is a good guidance for meta-clustering. Meta-clustering can generate more than two clusterings to let the users to choose. However, user may feel overwhelming. Therefore, we set the number of clustering groups to 2 and it generates two clusterings as the result. Although 50 clusterings are generated at first, the two output clusterings are quite similar. In addition

Table 5.5: Clusterings on the fruit image data set. The best cases are highlighted in bold.

Clustering produced by methods	Clusterings in ground truth etc.	NMI	RI	AR	MI	HI	Weight vector $\mathbf{w}$
k-means	Clustering-by-Category	.1486	.5659	.0684	.4341	.1319	-
Spectral		.1432	.5650	.0611	.4350	.1300	-
Meta-clustering 1		.1486	.5659	.0684	.4341	.1319	-
Meta-clustering 2		**.1870**	.5172	.0633	.4828	.0344	-
MSC: Clustering 1		.1394	.5857	.0818	.4143	.1714	[.2538,.0011,.0765,.0655,.1004,.5027]
MSC: Clustering 2		.1627	**.6045**	**.1289**	**.3954**	**.2092**	[.3222,.0000,.0000,.0000,.6778,.0000]
MSC: Clustering 3		.1449	.5886	.0883	.4114	.1773	[.4468,.0000,.5532,.0000,.0000,.0000]
MSC: Clustering 4		.1151	.5716	.0465	.4284	.1432	[.4012,.0000,.0000,.5988,.0000,.0000]
k-means	Clustering-by-Color	.5905	.7626	.4905	.2374	.5253	-
Spectral		.5522	.7559	.4730	.2441	.5117	-
Meta-clustering 1		.5905	.7626	.4905	.2374	.5253	-
Meta-clustering 2		.6478	.7132	.4435	.2868	.4264	-
MSC: Clustering 1		.6160	.7711	.4926	.2289	.5241	[.2538,.0011,.0765,.0655,.1004,.5027]
MSC: Clustering 2		.5564	.7474	.4436	.2526	.4949	[.3222,.0000,.0000,.0000,.6778,.0000]
MSC: Clustering 3		**.6886**	**.8051**	**.5681**	**.1949**	**.6103**	[.4468,.0000,.5532,.0000,.0000,.0000]
MSC: Clustering 4		.5124	.7291	.3971	.2709	.4582	[.4012,.0000,.0000,.5988,.0000,.0000]
k-means	Spectral	.8839	.9581	.9118	.0419	.9161	-
Meta-clustering 1	Meta-clustering 2	.6720	.7656	.5406	.2344	.5311	-

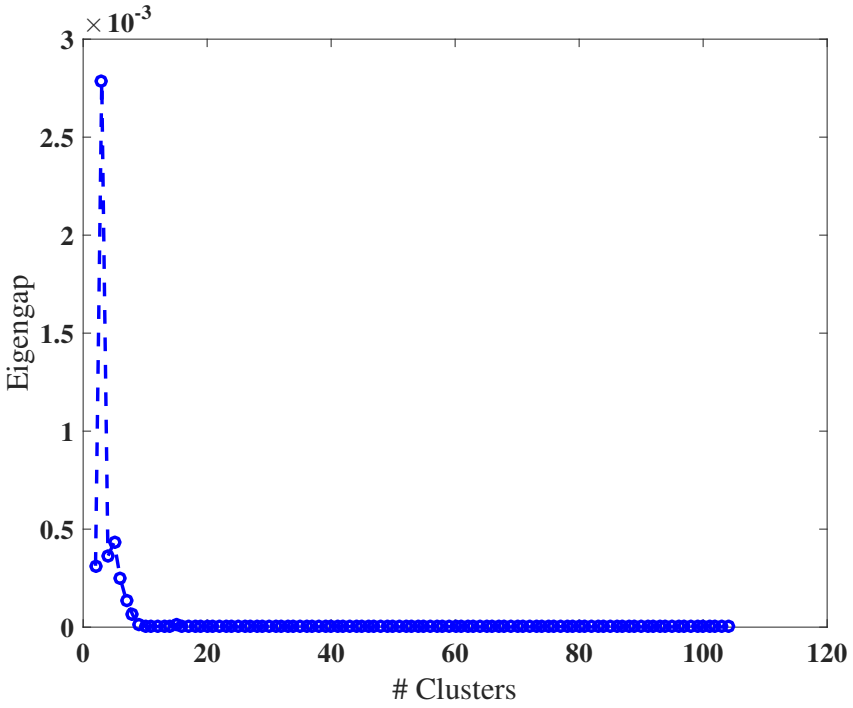


Figure 5.8: Eigengap distribution over #Clusters on the fruit data set.

to the high computational cost, it is difficult to determine the proper number of groups for those clusterings in general. Moreover, meta-clustering cannot provide additional subspace information of each clustering for user understanding.

MSC produces 4 stable clusterings on this data set. It is out of the users' expectation (two different clusterings). Each clustering solution of MSC is manifested by a feature subspace, so that users can understand each clustering by exploring its feature subspace and choose what they want. For instance, Clustering 3 produced by MSC corresponds to a feature subspace that only considers the color features. Consequently, Clustering 3 is very consistent with Clustering-by-Color in the ground truth, and is also more consistent than the clusterings generated by the baselines with respect to Clustering-by-Color. Simultaneously, Clustering 2 focuses on both the color and the shape features and puts more weight on the shape feature. Clustering 2 is substantially closer to Clustering-by-Category than those produced by the baselines. A clustering using both the shape and color features is highly reasonable. To distinguish between apples, bananas and grapes, only the shape features may not be enough. For example, a bunch of bananas may happen to have a shape similar to that of a bunch of grapes. Thus, we need to get help from the color attributes. For example, it happens that this data set does not have red bananas.

In addition to those two clusterings, MSC also produces two other stable clusterings, Clustering 1 and Clustering 4 in Table 5.5. Those two stable clusterings put weights on totally different subspaces. The 4 stable clusterings produced by MSC have low redundancy

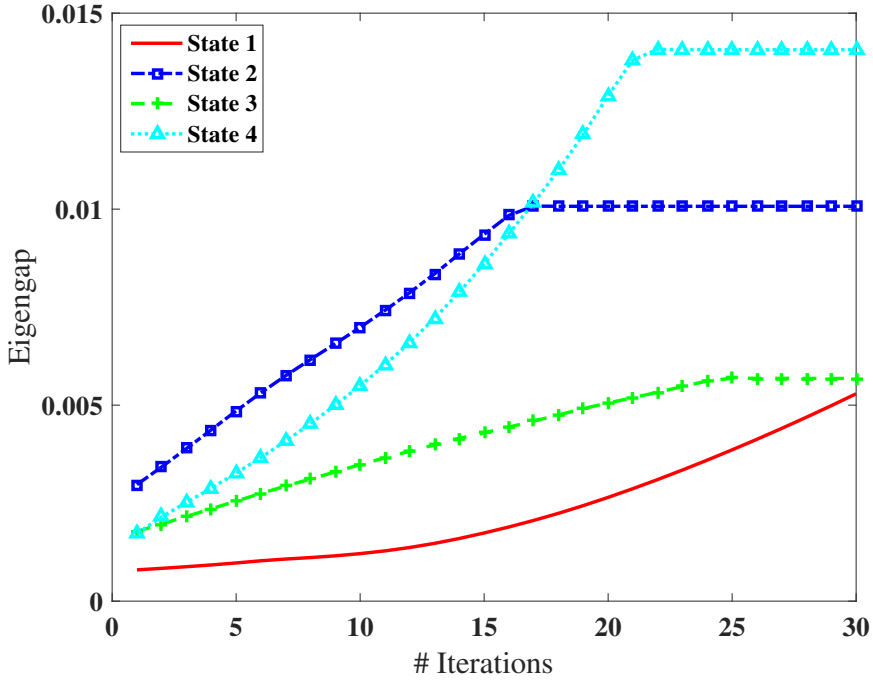


Figure 5.9: The changes of eigengap in iterations on the fruit data set.

– all pairwise NMIs between them are smaller than 0.55. Moreover, MSC provides different subspaces for the stable clusterings, which is helpful for user understanding. Interestingly, two different subspaces lead to two somehow similar clusterings on the data, such as Clustering 2 and Clustering 4 in Table 5.5. An important application of different weight vectors leading to similar clusterings is multi-view clustering [44, 38] which aims to combine results from different views to generate a consensus clustering.

Figure 5.9 shows how the eigengap of each clustering solution converges as the number of iterations increases. Figures 5.10(a) and 5.10(b) show the images that are closest to the cluster centers of Clustering 2 and Clustering 3, respectively. The cluster centers for Clustering 2 are apple, banana and grape, respectively, which is consistent with Clustering-by-Category in the ground truth. The cluster centers of Clustering 3 are yellow, green, and red, respectively, which matches the ground truth of Clustering-by-Color. Please note that colors green and red here are both represented by grapes. This further verifies that MSC can find meaningful weight vectors (subspaces) corresponding to different stable clusterings hidden in data.

5.4.3 More Results on the UCI Data Sets

We also test MSC on some real data sets from the UCI Machine Learning Data Repository [83]. We report the results on 4 data sets here as examples. The information about the data sets used in this subsection is summarized in Table 5.6.

(a) MSC: Clustering 2 (Category)

(b) MSC: Clustering 3 (Color)

Figure 5.10: The images closest to the cluster centers.

Table 5.6: Statistics of UCI data sets.

Data	# Instances	# Features	# Clusters
balance	625	4	3
iris	150	4	3
letter	20,000	16	26
skin	245,057	3	2

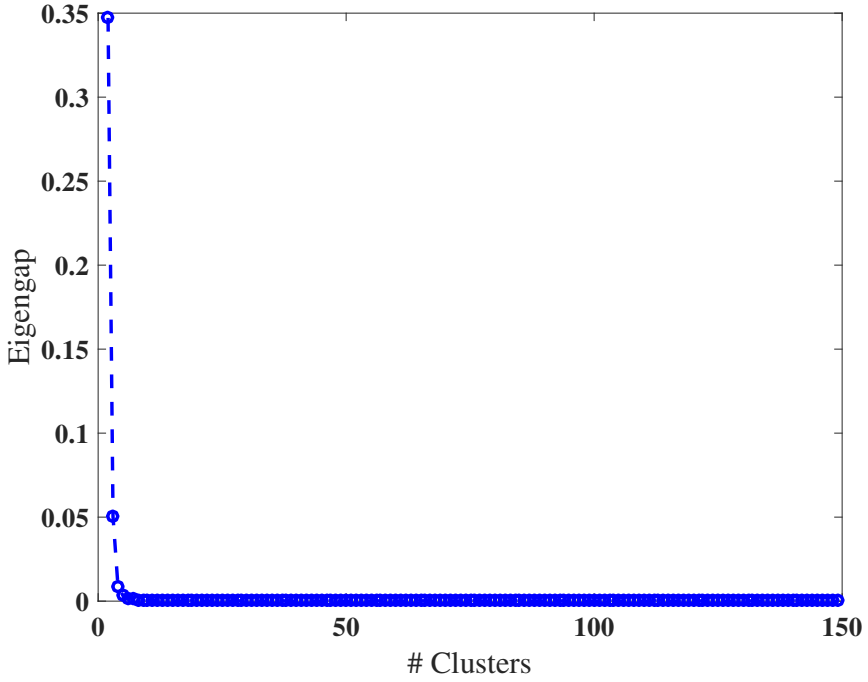


Figure 5.11: Eigengap distribution over #Clusters on the *iris* data set.

Note that each data set in this subsection provides only one ground-truth clustering with a specific number of clusters. We find that if MSC is used to determine the number of clusters based on the original data, it is not surprising to find that the value of k determined by MSC is different from the number of clusters in the ground truth. Because more stable clusterings may be hidden in the data besides the ground-truth clustering. For example, the eigengap distribution in Figure 5.11 over the number of clusters on the original *iris* data determines that $k = 2$. By studying its features, we find that the third feature (petal length) has a large gap that can clearly separate the iris plants into two clusters as plotted by the histograms in Figure 5.12.

To facilitate the comparison of clustering performance, we set k as the number of clusters in the ground truth for each data except the largest data set *skin*. For meta-clustering, we set the number of groups of clusterings in the second step as the number of clusterings output by our method MSC, which is generally not available for real tasks.

Results on Data Sets *Balance* and *Iris*

The parameters are set accordingly as $\tau = 0.0025d = 0.01$ and $\delta = 0.001$.

Table 5.7 compares the clusterings produced by the methods tested against the ground truth. MSC obtains two stable states on the *balance* data set. One of the stable states weights more on the first and second features, while the other stable state puts more weights

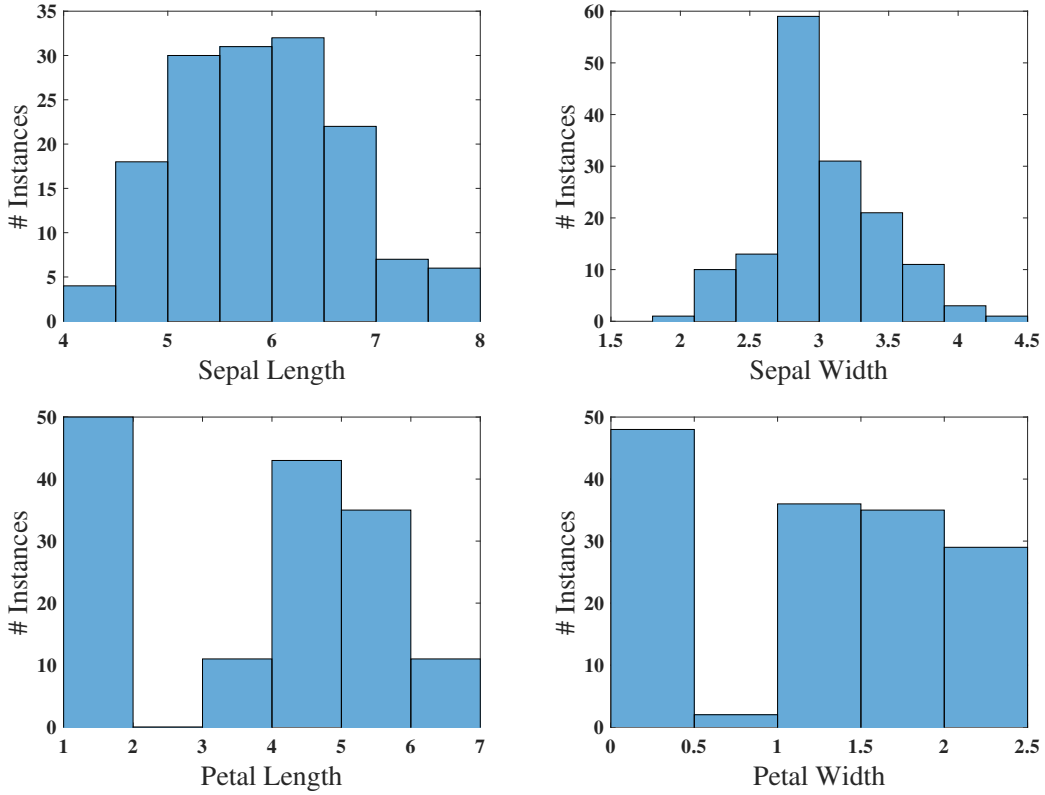


Figure 5.12: The histograms on each feature of the *iris* data set.

Table 5.7: Clusterings on the UCI data sets *Balance* and *Iris*. All clusterings produced by the methods tested are compared with the ground truth. "Meta-clustering 1 and 2" in the table means meta-clustering 1 and meta-clustering 2 are the same.

Clusterings produced by methods tested	NMI	RI	AR	MI	HI	Eigengap	Weight vector **w**
Data set *balance*							
k-means	.2702	.6551	.2765	.3449	.3102	-	-
Spectral	.2769	.6636	.2942	.3364	.3271	.0000	-
Meta-clustering 1	.2716	.6551	.3177	.3255	.3490	-	-
Meta-clustering 2	.1397	.6636	.1684	.3964	.2072	-	-
MSC: Clustering 1	**.3215**	**.6928**	**.3556**	**.3072**	**.3856**	.0126	[.5000,.5000,.0000,.0000]
MSC: Clustering 2	.1238	.5895	.1389	.4105	.1791	.0126	[.0000,.5006,.4994,.0000]
Data set iris							
k-means	.7582	.8797	.7302	.1203	.7595	-	-
Spectral	.7347	.8679	.7037	.1321	.7358	.0519	-
Meta-clustering 1 and 2	.7582	.8797	.7302	.1203	.7595	-	-
MSC: Clustering 1	**.8366**	**.9341**	**.8510**	**.0659**	**.8683**	.2733	[.0000,.0000,.8658,.1342]

Table 5.8: Clustering redundancy.

Clusterings compared	NMI	RI	AR	MI	HI
Data set *balance*					
k-means vs. Spectral	.1598	.6103	.1233	.3897	.2206
Meta-clustering 1 vs. Meta-clustering 2	.4706	.6438	.4697	.2362	.5276
MSC: Clustering 1 vs. MSC$_k$: Clustering 2	.2714	.6438	.1988	.3562	.2875
Data set *iris*					
k-means vs. Spectral	.9398	.9825	.9610	.0175	.9649

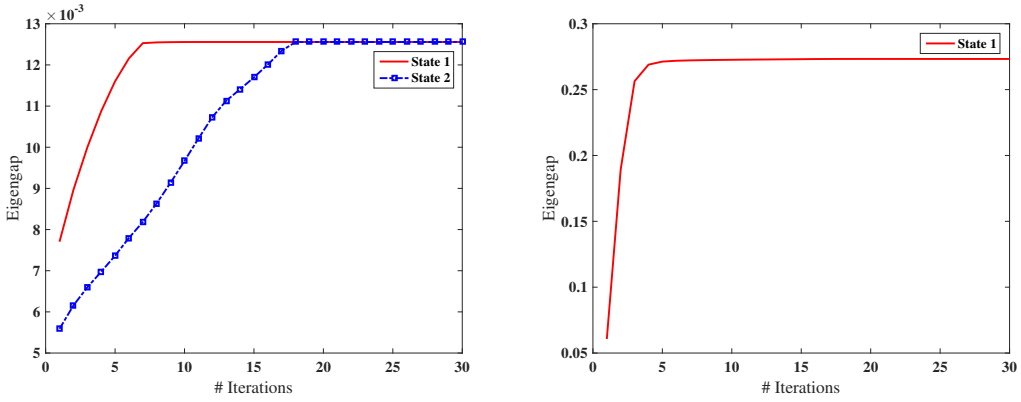
on the second and third features. The sparse property of the simplex constraint can also be observed from the values of these two obtained weight vectors.

The clustering generated by the first stable state is much more consistent with the provided ground truth than the clusterings produced by the baselines. The clustering produced by MSC has a larger eigengap than that produced by the spectral clustering method.

To understand the redundancy between clusterings, Table 5.8 compares the clusterings produced by k-means and Spectral, and the clusterings produced by meta-clustering. The redundancy between the clusterings produced by k-means and Spectral on the *balance* data set is low. This confirms that using different clustering methods may generate substantially different alternative clusterings. The second clustering produced by MSC is substantially different from the first one, which confirms the effectiveness of the stability measure. Although meta-clustering generates two different clusterings, the two clusterings are very similar.

On the *iris* data set, MSC finds a stable state that weights more on the third and the fourth features. The clustering is much more consistent with the ground truth comparing to the clusterings produced by the baselines, especially Spectral that has much smaller eigengap. These two methods generate very similar clusterings on this data set and thus fail to provide different alternative clusterings. Remarkably, MSC produces only one stable clustering on this data set. The result from MSC heuristically indicates that the data set may not contain multiple interesting clusterings. This information is very helpful in practice, since existing multi-clustering methods cannot determine the number of alternative clusterings in a data set exactly or heuristically. We let meta-clustering generate two clusterings. However, the two clusterings are the same.

Figure 5.13 shows how the eigengap converges as the number of iterations increases. On both data sets, the eigengaps converge in less than 20 iterations.



(a) *balance* (b) *iris*

Figure 5.13: The changes of eigengap in iterations on the *balance* and *iris* data sets.

Table 5.9: CPU time (sec.) when $r = 2,000$.

Steps	*letter*	*skin*
Select representatives	57.2500	0.0400
Discover stable states	46.6200	186.5370
Clustering	2.3500	20.3500

Results on Large Data Sets Letter and Skin

Letter and *skin* are larger data sets. We test whether SMSC can handle these large data sets.

For *letter*, the parameters are set accordingly to $\tau = 0.0025d = 0.0025 * 16 = 0.04$ and $\delta = 0.001$. To compare the clustering performance, we set $k = 26$ as the ground truth, since there are 26 different letters. We run k-means clustering to select $r = n * 1\% = 200$ representatives at first. SMSC produces 8 stable states by calling Algorithm 8 14 times. CPU time costs shown in Table 5.9 indicate that SMSC is capable on this large data set. Meanwhile, we compare the first clustering generated by SMSC with the ground truth, and the RI is 0.9272. We also apply 10 times of k-means clustering on the whole data set without using any sampling strategy and choose the best result to report, whose RI is 0.9299. This further demonstrates the scalability of our SMSC method.

Skin has 3 color features: B(lue), G(reen), and R(ed). These three values were collected by randomly sampling from RGB values of each face image. It contains 245,057 samples, out of which 50,859 is the skin samples and 194,198 is non-skin samples. The parameters are set accordingly to $\tau = 0.0025d = 0.0025 \times 3 = 0.0075$ and $\delta = 0.001$.

We randomly select $r = 2,000$ representatives at first. Then, SMSC determines the number of clusters $k = 2$ following the eigengap distribution in Figure 5.14. SMSC produces 6 stable states by calling Algorithm 8 14 times. The most similar one compared to the ground truth is with RI 0.8609, while the best k-means result without using any

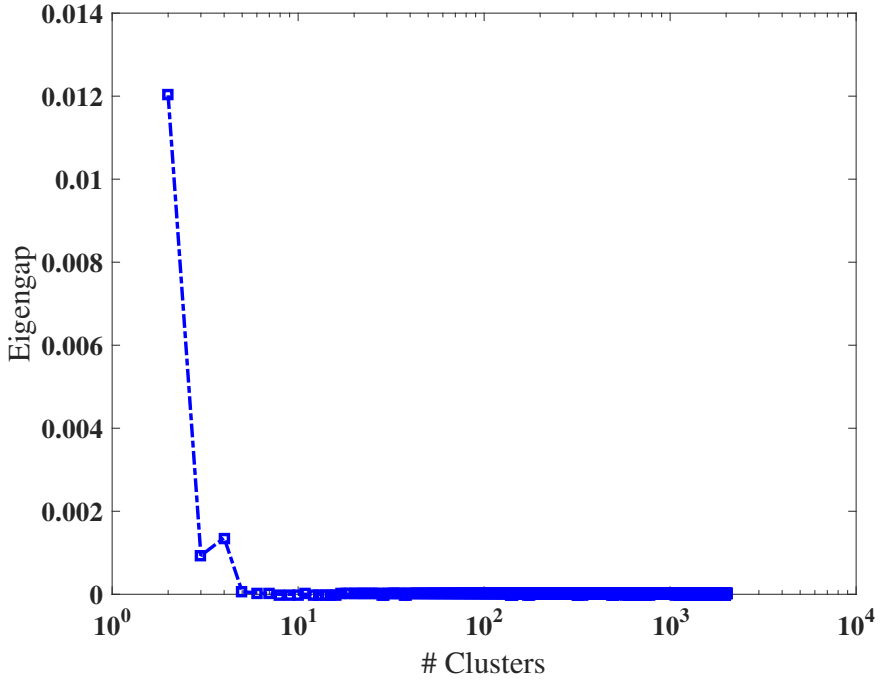


Figure 5.14: Eigengap distribution over #Clusters on $2,000$ representatives randomly selected from the *skin* data set.

sampling strategy is 0.8673. For this clustering result, SMSC also provides a subspace of $[0.3235, 0.3006, 0.3759]$. It indicates that, to distinguish between skin and non-skin, we need to consider all color features. Besides, SMSC provides some other stable clusterings with 2 clusters in different subspaces, e.g., $[0.3865, 0.0000, 0.6135]$. The clustering performance and the CPU time cost shown in Table 5.9 both verify the effectiveness of our speed-up technique in SMSC.

5.5 Summary

In this chapter, to address the practical demands on multi-clustering, we tackle the challenges of how to model the quality of clusterings and how to find multiple stable clusterings in a given data set.

We apply the idea of clustering stability based on Laplacian eigengap to multi-clustering. Mathematically, we prove that the larger the eigengap, the more stable the clustering. Then, We develop a novel subspace multi-clustering method MSC based on the notion of clustering stability. We model the problem of finding a stable clustering as an optimization problem maximizing the eigengap over the feature subspaces. The number of clusters k is determined by maximizing the eigengap on the original data. We find that when k is fixed, the optimization problem is unfortunately non-convex, and thus we propose a heuristic random-

ized method using iterative gradient ascent. In order to find multiple independent stable clusterings with the same number of clusters, we introduce to the optimization problem a constraint on the differences between weight vectors.

A unique advantage of our method comparing to the existing multi-clustering methods is that our method can provide interpretable feature subspaces to help users understand corresponding clusterings. Another advantage is that users do not need to specify the number of clusters and the number of alternative clusterings in the data set, which is usually difficult for users without given any guidance. Our method can heuristically estimate the number of meaningful clusterings in a data set, which is generally infeasible in the existing multi-clustering methods. An empirical study on synthetic and real data sets clearly demonstrates the effectiveness of our method.

Chapter 6

Conclusions

In this big data era, obtaining large amount of data has become more and more feasible, while the main challenge rises from discovering new knowledge from raw data itself, that is, discovering without any domain knowledge. Clustering methods can provide useful group information of data to help users extract new knowledge. However, the challenge to efficiently and effectively understand a clustering structure remains, due to the reason that there is a big semantic gap between low-level clustering structures and high-level understandable meanings.

Subspace clustering methods have strengths on either finding feature subspaces in which preferred (by end users) clustering structures reside or helping end users intuitively interpret clustering structures discovered in specific feature subspaces. That is, subspace clustering methods, more often than not, can provide understandable information organizations for end users. Specifically, subspaces are used to bridge the semantic gap. Due to the high demand of subspace clustering methods in real-world applications, such as content-based image retrieval, customer relationship management, and DNA microarray analysis, efficient, effective, and user-friendly methods have received more and more attentions from the research community. In this thesis, we develop three subspace clustering methods to tackle arising challenges from real-world applications.

In this chapter, we first summarize the thesis, and then discuss a few interesting future research directions.

6.1 Summary of The Thesis

In this thesis, we study subspace clustering problem in different application scenarios (e.g., semi-supervised and unsupervised). We propose three efficient and effective subspace clustering methods and make the following contributions.

- We develop a regularized distance metric learning method for a semi-supervised scenario, in which end users may provide limited amount of side information (e.g.,

instance-level constraints) to tell their preferences on the clustering structures. To alleviate the common overfitting problem in DML, we, for the first time, introduce dropout to DML and apply it to both the learned metric and the training data. We illustrate that application of dropout to DML is essentially equivalent to matrix norm based regularization.

  - Applying dropout to metric: we show that dropout can be equivalent to Frobenius norm and L_p in expectation by applying appropriate dropout probabilities to the learned metric. Besides, we develop a structured regularizer to introduce different dropout probabilities for diagonal elements and off-diagonal elements.
  - Applying dropout to data: we observe that dropout behaves like a trace norm based regularizer in DML when applied to training data. Therefore, it controls the rank of the learned metric and leads to a skewed distribution of eigenvalues. This is in contrast to the previous studies that viewed dropout as a L_2 regularizer.

  Compared with the standard regularization scheme in DML, dropout is advantageous in simulating the structured regularizers. We verify, both empirically and theoretically, that dropout is effective in regulating the learned metric to avoid the overfitting problem.

- We, for the first time, develop a subspace hierarchical clustering method to help users view, search, and organize a large amount of data. To construct a hierarchy with semantics for efficient search, we present a simple strategy to extract all meaningful semantics from objects as candidate dimensions, and then develop an efficient unsupervised feature/dimension selection method to select a sufficient dimension subset to uniquely identify each object, which naturally constructs a most balanced hierarchical clustering structure for objects. We study this problem in a specific application scenario, that is, a large amount of raw images in CBIR.

  - Feature extraction: we treat each image as a bag of instances, where each instance is basically a sub-region of the image and has semantic meaning as "sky", "trees", or "mountains". We then extract all instances from the image collection and consider them as a candidate dimension set with semantics.
  - Feature selection: we formulate the problem of subspace hierarchical clustering as an optimization problem which aims to discover a minimum subset of dimensions such that each image can be uniquely identified in the lowest level of the hierarchy. Due to the hardness of this problem, we propose a greedy algorithm that selects a most discriminative dimension in each iteration.

  We empirically verify that a relatively smaller subset of dimensions are selected to build a most balanced (i.e., the number of images in the left child is almost the same

as that in the right child) binary tree structure. That is, our subspace hierarchical clustering structure supports more efficient search of relevant images in the hierarchy.

- We, for the first time, introduce the notion of clustering stability based on Laplacian eigengap into multi-clustering. We mathematically prove that the larger the eigengap, the more stable the clustering. Furthermore, we propose a novel multi-clustering method, MSC, based on the clustering stability. An advantage of our method comparing to the state-of-art multi-clustering methods is that our method can provide users a feature subspace to understand each clustering solution. Another advantage is that MSC does not require users to specify the number of clusterings and their number of clusters, which is usually difficult for users without any guidance.

  - The number of clusters: we use the number of clusters which can maximize the eigengap in the original feature subspace, because if we can find all stable clusterings with a fixed number of clusters, other clusterings with different number of clusters are basically combining or splitting clusters.
  - The interpretable feature subspace: we use a feature-weighting vector to indicate a feature subspace, and it can show used features and their importance.
  - The number of clusterings: we search all different and stable weight vectors in the simplex, which heuristically determines the number of clusterings.

  We also discuss a practical way to make MSC applicable to large-scale data. We verify MSC on both synthetic and real, small and big data sets.

6.2 Future Study: Extending the Thesis Directly

It is interesting to extend our subspace clustering methods in this thesis to other well related application scenarios. Some of them are listed below.

- *Semi-supervised subspace clustering*: In Chapter 3, we aim to discover a feature subspace, in which a global distance metric is applicable to all data points. In some applications, such as fine-grained visual categorization, we need to distinguish subordinate categories within each entry-level category. For example, an animal collection may contain images of "cat", "dog", "bird" and so on. However, the task is to distinguish their subordinate categories. That is, we need to recognize different species of different entry-level categories at the same time, such as "siamese" and "persian" for cat, "poodle" and "beagle" for dog, and "northern cardinal" and "indigo bunting" for bird. Different entry-level categories usually have their own ways to define their species. Therefore, it is difficult for a single feature subspace or a global metric to capture both the properties of the entry-level categories and their subordinate categories. A straightforward way is to conduct a two-stage metric learning, where the first

stage distinguishes entry-level categories and the second stage distinguishes species. However, we need to learn multiple metrics, each for an entry-level category in the second stage, which is computationally expensive. Therefore, how to effectively and efficiently use DML to find one or multiple feature subspaces for this kind of problems is an challenging future direction.

- *Subspace hierarchical clustering*: In Chapter 4, we propose to choose a minimum subset of semantic dimensions to construct a hierarchical clustering structure for efficient search in a large amount of raw images. Here, the image collection is static. However, in real-world applications, image collections are usually dynamic. For example, the number of photos in our phones is increasing every day, an X-ray image collection can be enlarged every day from routine examinations, and Instagram users are posting 216,000 new photos every minute [1]. The easiest way to maintain the hierarchical clustering structure is inserting each new image into the already constructed hierarchy using selected dimensions. However, the number of images in the leaves will increase and the hierarchy is likely to degenerate into a huge collection of images again. Moreover, discriminative dimensions changes when adding images. It seems that it is wiser to repeat the whole process periodically. However, it is too expensive to handle high-volume and high-velocity data set in such a way. Therefore, there is a great demand on online subspace clustering methods for this kind of applications.

- *Subspace multi-clustering*: It is often hard for users to determine the number of clusterings without substantial domain knowledge. In Chapter 5, MSC can heuristically determine the number of clusterings hidden in a given data set. Specifically, MSC keeps searching different subspaces with stable clusterings in the simplex until no new subspace can be found. However, there is no theoretical guarantee that the number of clusterings discovered by MSC matches the ground truth. This is still an open challenge. Moreover, to tackle large-scale data, we present a potential strategy. We select a reasonable number of representatives from the whole data set. Then, multiple clusterings are generated on these representatives. Thereafter, the rest data are assigned to their nearest clusters in the corresponding subspace. However, the clustering quality is to some extent compromised. Therefore, more efficient and effective subspace multi-clustering approaches are necessary in the future.

6.3 Future Research Directions

Besides extending our work in this thesis, it is highly interesting to provide understandable information organizations for other general applications using subspace clustering methods. Several potential ones are as follows.

[1] `http://time.com/73581/data-generated-online-every-minute-domo/`

- *Time-series clustering*: Clustering has been widely used for time series data with applications in a board range of scenarios, e.g., finding regions with similar temperature or cities with similar population growth. Many previous researches focused on the similarity measures between two finite-length time series, either on raw data or extracted features. However, time series are often high-dimensional or even infinite in the time space. Therefore, each time series becomes more and more unique as time goes. In fact, clusters of time-series are often hidden within specific time subspaces. Moreover, different time may promote different groups. Values observed at other time spots can be harmful for discovering these clusters. Many biclustering methods have been proposed to handle similar problems on gene expression data [119]. However, it is still a challenge to discover this kind of clusters when considering time as an infinite space.

- *Subspace multi-clustering with multiple sources/views*: Data can be collected from different sources, e.g., audio, image and text, each providing a representation of the data. Therefore, multiple clusterings can be obtained by considering each representation separately. Many previous researches focus on the techniques to establish a single clustering by combining information obtained from different sources/views. However, different sources/views may represent totally different perspectives of the data, and thus provide independent structures over the data. Moreover, it is still possible that a single source/view may be too weak to provide a sufficiently stable clustering. How to obtain multiple independent partitions from multiple views/sources will be an interesting and meaningful future direction.

Bibliography

[1] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 61–72, Philadelphia, PA, 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 70–81, Dallas, TX, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, Seattle, WA, 1998.

[4] S. Alelyani, J. Tang, and H. Liu. Feature selection for clustering: A review. In C. C. Aggarwal and C. K. Reddy, editors, *Data Clustering: Algorithms and Applications*. Taylor & Francis, 2013.

[5] I. Assent, R. Krieger, E. Müller, and T. Seidl. INSCY: Indexing subspace clusters with in-process-removal of redundancy. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 719–724, Pisa, Italy, 2008.

[6] I. Assent, R. Krieger, E.l Müller, and T. Seidl. DUSC: Dimensionality unbiased subspace clustering. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 409–414, Omaha, NE, 2007.

[7] A. Azran and Z. Ghahramani. Spectral methods for automatic multiscale data clustering. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 190–197, New York, NY, 2006.

[8] E. Bae and J. Bailey. COALA: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 53–62, Hong Kong, China, 2006.

[9] E. Bae, J. Bailey, and G. Dong. A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings. *Data Mining and Knowledge Discovery*, 21(3):427–471, 2010.

[10] J. Bailey. Alternative clustering analysis: A review. In C. C. Aggarwal and C. K. Reddy, editors, *Data Clustering: Algorithms and Applications*. Taylor & Francis, 2013.

[11] P. Baldi and P. J. Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822, 2013.

[12] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[13] P. Berkhin. A survey of clustering data mining techniques. In *Grouping Multidimensional Data - Recent Advances in Clustering*, pages 25–71. 2006.

[14] P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

[15] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 19–26, Brighton, UK, 2004.

[16] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[17] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 27–34, Brighton, UK, 2004.

[18] K. Böhm, F. Keller, E. Müller, H. Vu Nguyen, and J. Vreeken. CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection. In *Proceedings of the 13th SIAM International Conference on Data Mining*, pages 198–206, Austin, TX, 2013.

[19] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[20] L. Cao, L. Luo, and T. S. Huang. Annotating photo collections by label propagation according to multiple similarity cues. In *Proceedings of the 16th ACM International Conference on Multimedia*, pages 121–130, Vancouver, Canada, 2008.

[21] R. Caruana, M. F. Elhawary, N. Nguyen, and C. Smith. Meta clustering. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 107–118, Hong Kong, China, 2006.

[22] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[23] H. Chang and D.-Y. Yeung. Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of the 21st international conference on Machine learning*, pages 153–160, Banff, Canada, 2004.

[24] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.

[25] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, San Francisco, CA, 2011.

[26] Y. Chen, J. Z. Wang, and R. Krovetz. CLUE: Cluster-based retrieval of images by unsupervised learning. *IEEE Transactions on Image Processing*, 14(8):1187–1201, 2005.

[27] C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 84–93, San Diego, CA, 1999.

[28] R. L. F. Cordeiro, A. J. M. Traina, C. Faloutsos, and C. Traina Jr. Finding clusters in subspaces of very large, multi-dimensional datasets. In *Proceedings of the 26th International Conference on Data Engineering*, pages 625–636, Long Beach, CA, 2010.

[29] Y. Cui, X. Z. Fern, and J. G. Dy. Non-redundant multi-view clustering via orthogonalization. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 133–142, Omaha, NE, 2007.

[30] X. Dang and J. Bailey. Generating multiple alternative clusterings via globally optimal subspaces. *Data Mining and Knowledge Discovery*, 28(3):569–592, 2014.

[31] X. Dang and J. Bailey. A framework to uncover multiple alternative clusterings. *Machine Learning*, 98(1-2):7–30, 2015.

[32] S. Dasgupta and V. Ng. Mining clustering dimensions. In *Proceedings of the 27th International Conference on Machine Learning*, pages 263–270, Haifa, Israel, 2010.

[33] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[34] I. Davidson and Z. Qi. Finding alternative clusterings using constraints. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 773–778, Pisa, Italy, 2008.

[35] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[36] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, Corvallis, OR, 2007.

[37] H. Do, A. Kalousis, J. Wang, and A. Woznica. A metric learning perspective of SVM: On the relation of LMNN and SVM. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, pages 308–317, La Palma, Canary Islands, 2012.

[38] C. Domeniconi and M. Al-Razgan. Weighted cluster ensembles: Methods and analysis. *ACM Transactions on Knowledge Discovery from Data*, 2(4), 2009.

[39] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[40] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(4):32–57, 1973.

[41] P. Duygulu, K. Barnard, J. F. de Freitas, and D. A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the 7th European Conference on Computer Vision*, pages 97–112, Copenhagen, Denmark, 2002.

[42] J. Eakins and M. Graham. *Content-based Image Retrieval*. Education-line, 1999.

[43] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, OR, 1996.

[44] X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the 20th International Conference on Machine Learning*, pages 186–193, Washington, DC, 2003.

[45] I. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Center for Applied Scientific Computing, Lawrence Livermore National, 2002.

[46] J. E. Gentle. Matrix transformations and factorizations. *Matrix Algebra: Theory, Computations, and Applications in Statistics*, pages 173–200, 2007.

[47] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset, 2007.

[48] S. Günnemann, I. Färber, and T. Seidl. Multi-view clustering using mixture models in subspace projections. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 132–140, Beijing, China, 2012.

[49] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[50] J. A. Hartigan and M. A. Wong. Algorithm ASI36: A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[51] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[52] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[53] E. Hazan and S. Kale. Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

[54] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[55] S. C. H. Hoi, M. R. Lyu, and R. Jin. A unified log-based relevance feedback scheme for image retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 18(4):509–524, 2006.

[56] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[57] J. Hu and J. Pei. Subspace multi-clustering: A review. *Knowledge and Information Systems*, page to appear, 2017.

[58] J. Hu, Q. Qian, J. Pei, R. Jin, and S. Zhu. Finding multiple stable clusterings. In *Proceedings of the 15th IEEE International Conference on Data Mining*, pages 171–180, Atlantic City, NJ, 2015.

[59] M. Hua and J. Pei. Clustering in applications with multiple data sources - A mutual subspace clustering approach. *Neurocomputing*, 92:133–144, 2012.

[60] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[61] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[62] P. Jain, R. Meka, and I. S. Dhillon. Simultaneous unsupervised learning of disparate clusterings. In *Proceedings of the 8th SIAM International Conference on Data Mining*, pages 858–869, Atlanta, GA, 2008.

[63] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. Scalable density-based distributed clustering. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 231–244, Pisa, Italy, 2004.

[64] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 119–126, Toronto, Canada, 2003.

[65] T. Joachims. *Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2002.

[66] I. Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2002.

[67] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 246–256, Lake Buena Vista, FL, 2004.

[68] K. Kailing, H.-P. Kriegel, P. Kröger, and S. Wanka. Ranking interesting subspaces for clustering high dimensional data. In *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 241–252, Cavtat-Dubrovnik, Croatia, 2003.

[69] K. Kailing, H.-P. Kriegel, A. Pryakhin, and M. Schubert. Clustering multi-represented objects with noise. In *Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 394–403, Sydney, Australia, 2004.

[70] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[71] H.-P. Kriegel, P. Kröger, M. Renz, and S. H. R. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 250–257, Houston, TX, 2005.

[72] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

[73] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[74] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1+\epsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proceedings of the 45th Symposium on Foundations of Computer Science*, pages 454–462, Rome, Italy, 2004.

[75] A. T. Kyrillidis, S. Becker, V. Cevher, and C. Koch. Sparse projections onto the simplex. In *Proceedings of the 30th International Conference on Machine Learning*, pages 235–243, Atlanta, GA, 2013.

[76] M. H. C. Law, M. A. T. Figueiredo, and A. K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, 2004.

[77] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429, San Jose, CA, 2007.

[78] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2(1):1–19, 2006.

[79] F.-F. Li, R. VanRullen, C. Koch, and P. Perona. Rapid natural scene categorization in the near absence of attention. *PNAS*, 99(14):9596–9601, 2002.

[80] L.-J. Li and F.-F. Li. What, where and who? Classifying events by scene and object recognition. In *Proceedings of the 11th IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[81] M. Li, X.-C. Lian, J. T. Kwok, and B.-L. Lu. Time and space efficient spectral clustering via column sampling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2297–2304, Colorado Springs, CO, 2011.

[82] Y.-F. Li, J.-H. Hu, Y. Jiang, and Z.-H. Zhou. Towards discovering what patterns trigger what labels. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1012–1018, Toronto, Canada, 2012.

[83] M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

[84] D. K. H. Lim, B. McFee, and G. Lanckriet. Robust structural metric learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 615–623, Atlanta, GA, 2013.

[85] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[86] B. Long, P. S. Yu, and Z. M. Zhang. A general model for multiple view unsupervised learning. In *Proceedings of SIAM International Conference on Data Mining*, pages 822–833, Atlanta, GA, 2008.

[87] K. Matsuoka. Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):436–440, 1992.

[88] W. May, S. Fidler, A. Fazly, S. Dickinson, and S. Stevenson. Unsupervised disambiguation of image captions. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 85–89, Montréal, Canada, 2012.

[89] L. M. Mayron. *Image Retrieval Using Visual Attention*. Florida Atlantic University, 2008.

[90] M. Meilă and S. Shortreed. Regularized spectral learning. *Journal of Machine Learning Research*, 2006:1–20, 2006.

[91] Y. Mori, H. Takahashi, and R. Oka. Image-to-word transformation based on dividing and vector quantizing images with words. In *Proceedings of the 1st International Workshop on Multimedia Intelligent Storage and Retrieval Management*, pages 405–409, Orlando, FL, 1999.

[92] E. Müller, I. Assent, R. Krieger, S. Günnemann, and T. Seidl. DensEst: Density estimation for data mining in high dimensional spaces. In *Proceedings of SIAM International Conference on Data Mining*, pages 175–186, Sparks, NV, 2009.

[93] E. Müller, S. Günnemann, I. Färber, and T. Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Proceedings of the 28th IEEE International Conference on Data Engineering*, pages 1207–1210, Washington, DC, 2012.

[94] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[95] H. S. Nagesh, S. Goil, and A. N. Choudhary. Adaptive grids for clustering massive data sets. In *Proceedings of the 1st SIAM International Conference on Data Mining*, pages 1–17, Chicago, IL, 2001.

[96] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[97] Y. Nesterov. *Gradient Methods for Minimizing Composite Objective Function*. Core Louvain-la-Neuve, 2007.

[98] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.

[99] D. Niu, J. G. Dy, and M. I. Jordan. Multiple non-redundant spectral clustering views. In *Proceedings of the 27th International Conference on Machine Learning*, pages 831–838, Haifa, Israel, 2010.

[100] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: A review. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.

[101] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 418–427, Madison, WI, 2002.

[102] Z. Qi and I. Davidson. A principled and flexible framework for finding alternative clusterings. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 717–726, Paris, France, 2009.

[103] A. Rényi. Representations for real numbers and their ergodic properties. *Acta Mathematica Hungarica*, 8(3-4):477–493, 1957.

[104] S. Rifai, X. Glorot, Y. Bengio, and P. Vincent. Adding noise to the input of a model trained with a regularized objective. *CoRR*, abs/1104.3250, 2011.

[105] D. Ritendra, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, 2008.

[106] L. Rokach and O. Maimon. *Data Mining with Decision Trees - Theory and Applications. 2^{nd} Edition*, volume 81 of *Series in Machine Perception and Artificial Intelligence*. WorldScientific, 2014.

[107] V. Roth and T. Lange. Feature selection in clustering problems. In *Advances in Neural Information Processing Systems*, pages 473–480, 2003.

[108] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[109] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[110] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[111] K. Sequeira and M. J. Zaki. SCHISM: A new approach to interesting subspace mining. *International Journal of Business Intelligence and Data Mining*, 1(2):137–160, 2005.

[112] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the 21st international conference on Machine learning*, page 94, Banff, Canada, 2004.

[113] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, San Diego, CA, 2001.

[114] B. Shaw, B. C. Huang, and T. Jebara. Learning a distance metric from a network. In *Advances in Neural Information Processing Systems*, pages 1899–1907, 2011.

[115] N. Srivastava. Improving neural networks with dropout. *University of Toronto*, 182, 2013.

[116] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[117] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory*. Academic Press, San Diego, CA, 1990.

[118] Z. Szabó, B. Póczos, and A. Lörincz. Separation theorem for independent subspace analysis and its consequences. *Pattern Recognition*, 45(4):1782–1791, 2012.

[119] A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. *Handbook of computational molecular biology*, 9(1-20):122–124, 2005.

[120] C. Tang, A. Zhang, and J. Pei. Mining phenotypes and informative genes from gene expression data. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 655–660, Washington, DC, 2003.

[121] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[122] L. van der Maaten, M. Chen, S. Tyree, and K. Q. Weinberger. Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning*, pages 410–418, Atlanta, GA, 2013.

[123] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[124] S. Wager, S. Wang, and P. Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359, 2013.

[125] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on on Innovative Applications of Artificial Intelligence*, page 1097, Austin, TX, 2000.

[126] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:947–963, 2001.

[127] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[128] C. K. I. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. 2001.

[129] B. Wiswedel, F. Höppner, and M. R. Berthold. Learning in parallel universes. *Data Mining and Knowledge Discovery*, 21(1):130–152, 2010.

[130] D. M. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, 2010.

[131] S. Xiang, X. Tong, and J. Ye. Efficient sparse group feature selection via nonconvex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 284–292, Atlanta, GA, 2013.

[132] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 505–512, 2002.

[133] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[134] R. Xu and D. C. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[135] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 907–916, Paris, France, 2009.

[136] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2, 2006.

[137] S. Yang and L. Zhang. Non-redundant multiple clustering by nonnegative matrix factorization. *Machine Learning*, pages 1–18, 2016.

[138] W. Ye, S. Maurus, N. Hubig, and C. Plant. Generalized independent subspace clustering. In *Proceedings of the 16th IEEE International Conference on Data Mining*, pages 569–578, Barcelona, Spain, 2016.

[139] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 689–692, Melbourne, FL, 2003.

[140] M.-L. Zhang and Z.-H. Zhou. ML-kNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[141] P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

[142] D. Zhou and C. J. C. Burges. Spectral clustering and transductive learning with multiple views. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1159–1166, Corvallis, OR, 2007.

[143] Z.-H. Zhou and M.-L. Zhang. Multi-instance multi-label learning with application to scene classification. In *Advances in Neural Information Processing Systems 19*, pages 1609–1616, 2006.

[144] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.

[145] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, Washington, DC, 2003.

[146] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, Washington, DC, 2003.

Appendix A

List of Publications

- J. Hu and J. Pei. "Subspace multi-clustering: A review". Knowledge and Information Systems (KAIS), 2017. DOI: 10.1007/s10115-017-1110-9.
- J. Hu, Q. Qian, J. Pei, R. Jin, and S. Zhu. "Finding multiple stable clusterings". Knowledge and Information Systems (KAIS), 2017, 51(3): 991-1021.
- J. Hu, Q. Qian, J. Pei, R. Jin, and S. Zhu. "Finding multiple stable clusterings" (**Best Paper Candidate**). In: Proceedings of the 15th IEEE International Conference on Data Mining (ICDM'15), Atlantic City, NJ, 2015, pp.171-180.
- Q. Qian, J. Hu, R. Jin, J. Pei, and S. Zhu. "Distance metric learning using dropout: A structured regularization approach". In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14), New York, NY, 2014, pp.323-332.
- J. Hu, J. Pei, and J. Tang. How can I index my thousands of photos effectively and automatically? An unsupervised feature selection approach. In: Proceedings of the 14th SIAM International Conference on Data Mining (SDM'14), Philadelphia, PA, 2014, pp.136-144.