Data Mining using Concepts of Independence, Unimodality and Homophily

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik

der Ludwig–Maximilians–Universität München

eingereicht von

Wei Ye

2018

1. Gutachter: Prof. Dr. Christian Böhm

2. Gutachter: Prof. Dr. Ambuj Singh

Tag der Einreichung: 5 July 2017

Tag der mündlichen Prüfung: 12 January 2018

Contents

List of Figures

List of Tables

List of Publications

The contributions of this thesis are based on the following four first-author papers:

1. **Wei Ye**, Linfei Zhou, Xin Sun, Claudia Plant, Christian Böhm, Attributed Graph Clustering with Unimodal Normalized Cut, *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)*, 2017.

2. **Wei Ye**, Linfei Zhou, Dominik Mautz, Claudia Plant, Christian Böhm, Learning from Labeled and Unlabeled Vertices in Networks, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2017.

3. **Wei Ye**, Samuel Maurus, Nina Hubig, Claudia Plant, Generalized Independent Subspace Clustering, *IEEE International Conference on Data Mining (ICDM)*, 2016.

4. **Wei Ye**, Sebastian Goebl, Claudia Plant, Christian Böhm, FUSE: Full Spectral Clustering, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

The following papers are not included in this thesis but are also considered as partial fulfillment of the requirements for the doctorate:

1. **Wei Ye**, Bianca Wackersreuther, Christian Böhm, Michael Ewers, Claudia Plant, IDEA: Integrative Detection of Early-stage Alzheimer's Disease, *Workshop on Data Mining for Medicine and Healthcare in conjunction with 16th SIAM International Conference on Data Mining (SDM)*, 2016.

2. Dominik Mautz, **Wei Ye**, Claudia Plant, Christian Böhm, Towards an Optimal Subspace for K-Means, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2017.

3. Linfei Zhou, **Wei Ye**, Bianca Wackersreuther, Claudia Plant, Christian Böhm, A Pseudometric for Gaussian Mixture Models, *International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA)*, 2017.

4. Linfei Zhou, **Wei Ye**, Zhen Wang, Claudia Plant, Christian Böhm, Indexing Multiple-Instance Objects, *International Conference on Big Data Analytics and Knowledge Discovery (DEXA)*, 2017.

5. Linfei Zhou, **Wei Ye**, Bianca Wackersreuther, Claudia Plant, Christian Böhm, Novel Similarity Measure for Gaussian Mixture Models, *International Conference on Big Data Analytics and Knowledge Discovery (DEXA)*, 2017.

6. Linfei Zhou, **Wei Ye**, Claudia Plant, Christian Böhm, Analysis of Complex Data Using Gaussian Mixture Models, *International Conference on Big Data Analytics and Knowledge Discovery (DaWaK)*, 2017.

Abstract

With the widespread use of information technologies, more and more complex data is generated and collected every day. Such complex data is various in structure, size, type and format, e.g. time series, texts, images, videos and graphs. Complex data is often high-dimensional and heterogeneous, which makes the separation of the wheat (knowledge) from the chaff (noise) more difficult. Clustering is a main mode of knowledge discovery from complex data, which groups objects in such a way that intra-group objects are more similar than inter-group objects. Traditional clustering methods such as k-means, Expectation-Maximization clustering (EM), DBSCAN and spectral clustering are either deceived by "the curse of dimensionality" or spoiled by heterogenous information. So, how to effectively explore complex data? In some cases, people may only have some partial information about the complex data. For example, in social networks, not every user provides his/her profile information such as the personal interests. Can we leverage the limited user information and friendship network wisely to infer the likely labels of the unlabeled users so that the advertisers can do accurate advertising? This is the problem of learning from labeled and unlabeled data, which is literarily attributed to semi-supervised classification.

To gain insights into these problems, this thesis focuses on developing clustering and semi-supervised classification methods that are driven by the concepts of independence, unimodality and homophily. The proposed methods leverage techniques from diverse areas, such as statistics, information theory, graph theory, signal processing, optimization and machine learning. Specifically, this thesis develops four methods, i.e. FUSE, ISAAC, UNCut, and wvGN. FUSE and ISAAC are clustering techniques to discover statistically

independent patterns from high-dimensional numerical data. UNCut is a clustering technique to discover unimodal clusters in attributed graphs in which not all the attributes are relevant to the graph structure. wvGN is a semi-supervised classification technique using the theory of homophily to infer the labels of the unlabeled vertices in graphs. We have verified our clustering and semi-supervised classification methods on various synthetic and real-world data sets. The results are superior to those of the state-of-the-art.

Zusammenfassung

Täglich werden durch den weit verbreiteten Einsatz von Informationstechnologien mehr und mehr komplexe Daten generiert und gesammelt. Diese komplexen Daten unterscheiden sich in der Struktur, Größe, Art und Format. Häufig anzutreffen sind beispielsweise Zeitreihen, Texte, Bilder, Videos und Graphen. Dabei sind diese Daten meist hochdimensional und heterogen, was die Trennung des Weizens (Wissen) von der Spreu (Rauschen) erschwert. Die Cluster Analyse ist dabei eine der wichtigsten Methoden um aus komplexen Daten wssen zu extrahieren. Dabei werden die Objekte eines Datensatzes in einer solchen Weise gruppiert, dass intra-gruppierte Objekte ähnlicher sind als Objekte anderer Gruppen. Der Einsatz von traditionellen Clustering-Methoden wie k-Means, Expectation-Maximization (EM), DBSCAN und Spektralclustering wird dabei entweder "durch der Fluch der Dimensionalität" erschwert oder ist angesichts der heterogenen Information nicht möglich. Wie erforscht man also solch komplexe Daten effektiv? Darüber hinaus ist es oft der Fall, dass für Objekte solcher Datensätze nur partiell Informationen vorliegen. So gibt in sozialen Netzwerken nicht jeder Benutzer seine Profil-Informationen wie die persönlichen Interessen frei. Können wir diese eingeschränkten Benutzerinformation trotzdem in Kombination mit dem Freundschaftsnetzwerk nutzen, um von von wenigen, einer Klasse zugeordneten Nutzern auf die anderen zu schließen. Beispielsweise um zielgerichtete Werbung zu schalten? Dieses Problem des Lernens aus klassifizierten und nicht klassifizierten Daten wird dem semi-supversised Learning zugeordnet.

Um Einblicke in diese Probleme zu gewinnen, konzentriert sich diese Arbeit auf die Entwicklung von Clustering- und semi-überwachten Klassifikationsmethoden, die von den

Konzepten der Unabhängigkeit, Unimodalität und Homophilie angetrieben werden. Die vorgeschlagenen Methoden nutzen Techniken aus verschiedenen Bereichen der Statistik, Informationstheorie, Graphentheorie, Signalverarbeitung, Optimierung und des maschinelles Lernen. Dabei stellt diese Arbeit vier Techniken vor: FUSE, ISAAC, UNCut, sowie wvGN. FUSE und ISAAC sind Clustering-Techniken, um statistisch unabhängige Muster aus hochdimensionalen numerischen Daten zu entdecken. UNCut ist eine Clustering-Technik, um unimodale Cluster in attributierten Graphen zu entdecken, in denen die Kanten und Attribute heterogene Informationen liefern. wvGN ist eine halbüberwachte Klassifikationstechnik, die Homophilie verwendet, um von gelabelten Kanten auf ungelabelte Kanten im Graphen zu schließen. Wir haben diese Clustering und semi-überwachten Klassifizierungsmethoden auf verschiedenen synthetischen und realen Datensätze überprüft. Die Ergebnisse sind denen von bisherigen State-of-the-Art-Methoden überlegen.

Chapter 1

Introduction

We are drowning in data but starved for knowledge. Due to the widespread use of information technologies, huge amounts of complex data are generated and collected every day. Such complex data are in the forms of documents, time-series, images, audios, videos, graphs, etc. Analysing such complex data is of great challenge and importance. The concept of Knowledge Discovery in Databases (KDD) [49] has been evolved as a possible solution for complex data mining. The KDD process mainly contains the steps as indicated in Figure 1.1 from [95]. The steps contain selection, preprocessing, transformation, data mining and evaluation. This thesis only focuses on the data mining step which is to extract patterns from the transformed data using unsupervised or semi-supervised learning methods.

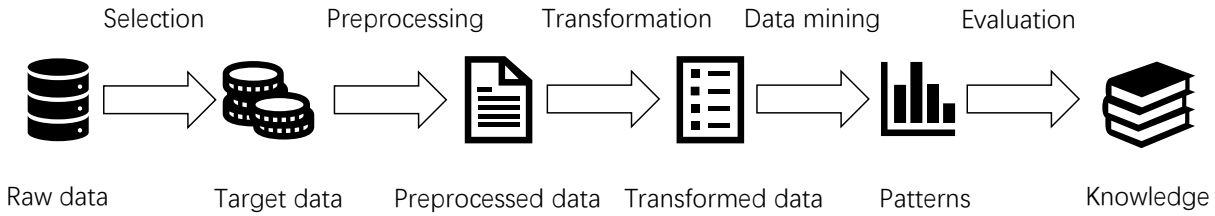


Figure 1.1: Knowledge Discovery in Databases (KDD) process.

For the step "data mining" in the KDD process, the ten most challenging problems have been identified in [79]. This thesis mainly focuses on the development of novel methods

for three challenges:

1. **Scaling up for high dimensional data and high speed data streams**.
2. **Mining complex knowledge from complex data**.
3. **Data mining in a network setting: community and social networks**.

One problem belonging to the first challenge is "the curse of dimensionality" [75] which refers to the difficulties that arise when analyzing data in high-dimensional spaces. One difficulty is that there tend to be little difference in the "similarity" or "dissimilarity" between different pairs of objects in high-dimensional spaces. Clustering methods that are based on Euclidean distance, Mahalanobis distance or Manhattan distance tend to fail in high-dimensional spaces. Subspace clustering methods have appeared as alternatives to full space clustering methods in high-dimensional spaces. In addition to "the curse of dimensionality", scalability is another problem in high-dimensional spaces. For example, the very famous spectral clustering needs to eigen-decompose the graph Laplacian matrix, which costs $\mathcal{O}(n^3)$ (n is the number of data objects) that prevents its application on large-scale data. Chapter 3 and Chapter 4 deal with the first challenge. The two unsupervised learning methods FUSE [100] and ISAAC [101] are based on the concept of independence.

One task belonging to the second challenge is the cluster detection in attributed graphs or data that are not i.i.d (independent and identically distributed), i.e. data objects are not independent of each other and are not of a single type. Such data includes social networks where friendship relationships are available along with the users' individual attributes (cf. Figure 1.2); system biology where interacting genes and their specific expression levels are given; and sensor networks where connections between sensors as well as individual measurements are recorded. Generally, not all attributes are relevant to the graph structure. To detect clusters in attributed graphs, we need to consider both types of information. How to effectively and efficiently use available information remains a big challenge. In Chapter 5, we propose a method UNCut to deal with this challenge.

One task belonging to the third challenge is the vertex classification in plain graphs.

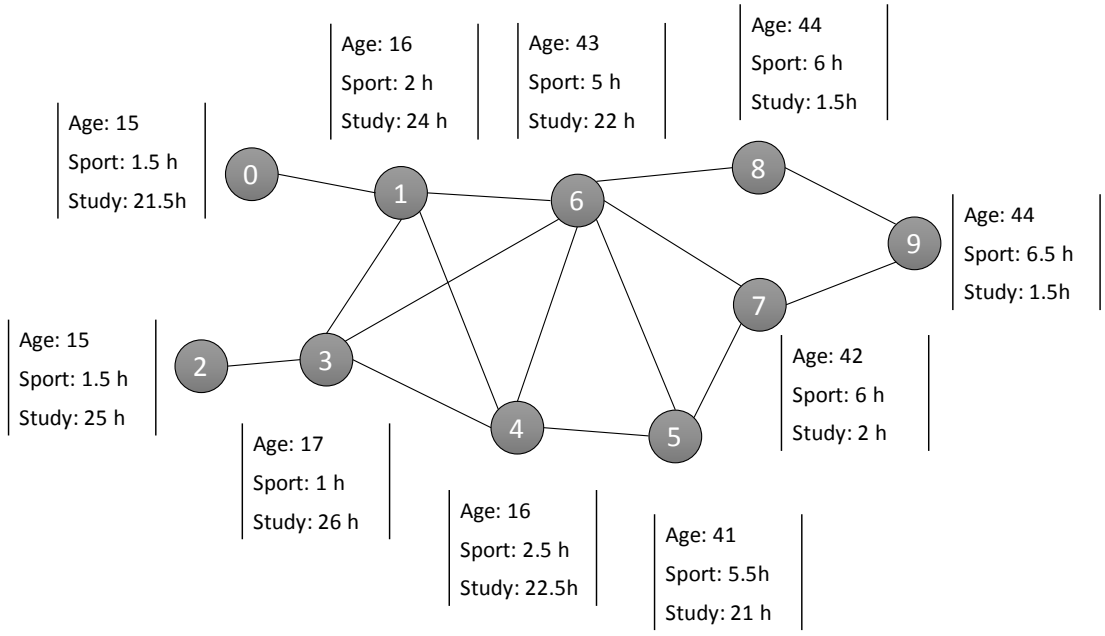


Figure 1.2: An example social network with graph and vector data.

Generally, we know the labels of some vertices in graphs. However, only very few vertices have labels compared to large amounts of unlabeled vertices. For example, in social graphs, not every user provides his/her profile information such as the personal interests which are relevant for targeted advertising. Can we leverage the limited user information and friendship graph wisely to infer the labels of unlabeled users? Chapter 6 in this thesis deals with the vertex classification in graphs.

To deal with the above listed three challenges, this thesis develops four novel methods using concepts of independence, unimodality and homophily.

1.1 Independence

The task of Blind Source Separation (BSS) is to separate a set of source signals from a set of mixed signals under some assumptions, e.g. the source signals are non-negative, or statistically independent, etc. In this thesis, the concept of statistically independence is exploited to discover non-redundant or statistically independent patterns from data. In probability theory, two random variables are statistically independent if the realization of

one does not affect the probability distribution of the other. Two main BSS techniques called Independent Component Analysis (ICA) [5] and Independent Subspace Analysis (ISA) [4] are adopted as foundations of our proposed methods FUSE and ISAAC. Why do we want to find non-redundant patterns?

High-volume data may have different perspectives inside, which can be clustered in different ways in different subspaces. Figure 1.3 shows the Amsterdam Library of Object Images (ALOI) data set (can be found here[1]) that can be interpreted in different ways, i.e. objects can be clustered by their shape or color. A global dimensionality reduction method like PCA [44] cannot deal with this kind of multifaceted data. Therefore, subspace clustering is developed to tackle the difficulties. Subspace clustering achieves finding all clusters in all subspaces. By using relevant attributes to form a subspace, subspace clustering methods can not only avoid "the curse of dimensionality", but also find different perspectives in data. However, many existing subspace clustering methods find many redundant clusters which do not report extra useful or novel information in the data and which result in high runtime, low quality results and overwhelming data analysts. Thus, detecting non-redundant patterns is of great importance in data mining.

Figure 1.3: Four objects of different shapes and colors from ALOI.

1.2 Unimodality

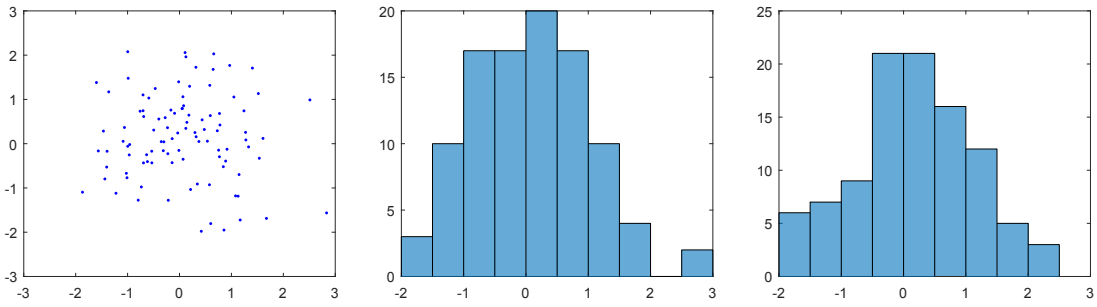
In statistics, a unimodal distribution refers to a probability distribution which only has a single mode (i.e. peak). A mode of a continuous probability distribution is a value at which the probability density function (PDF) attains its maximum value. If the maximum

[1] http://aloi.science.uva.nl/

value of a PDF can be attained at more than one value, such a distribution is called multimodal distribution. From the behavior of the cumulative distribution function (CDF), unimodal distribution can also be defined as: if the CDF is convex for $x < m$ and concave for $x > m$ (m is the mode), then the distribution is unimodal. Unimodal distributions include Gaussian distribution, Cauchy distribution, Student's t-distribution, chi-squared distribution, exponential distribution and etc. In the task of knowledge discovery from data, we consider each cluster taking a unimodal distribution. To measure the unimodality, we adopt a statistic test called Hartigans' dip test [51] in Chapter 5. Figure 1.4 and Figure 1.5 give us an intuitive demonstration to understand the concept of unimodality and multimodality. Figure 1.4(a) shows a Gaussian cluster with mean 0 and variance 1. Figure 1.4(b) depicts the histogram after projecting data onto the x-axis and Figure 1.4(c) depicts the histogram after projecting data onto the y-axis. If we apply the dip test on the data projection onto the x-axis, we get a dip value of 0.0258 and p-value of 0.957. Likewise, we apply the dip test on the data projection onto the y-axis, we get a dip value of 0.0226 and p-value of 0.994. The dip test tells us that the data shown in Figure 1.4(a) follows unimodal distributions because the p-value is greater than the significance level $\alpha = 0.05$. For the data shown in Figure 1.5(a), we project the data onto the x-axis and y-axis as well. Figure 1.5(b) and Figure 1.5(c) show their histograms. We apply the dip test on both axises. The dip value of the x-axis is 0.0867 and the p-value is 0; the dip value of the y-axis is 0.0937 and the p-value is 0. Since the p-value is less than the significance level $\alpha = 0.05$, the data follows multimodal distributions.

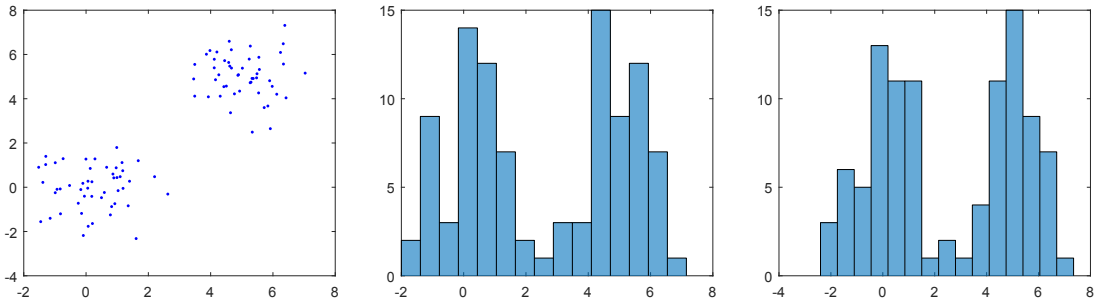
1.3 Homophily

Homophily ("birds of a feather flock together") in networks refers to that similar individuals are likely to be connected. For example, in a social network, people with similar interests, e.g. politics, religion, education and etc., are likely to be connected; in a citation network, papers with similar topics are likely to be connected. Homophily has also been discovered in other scenerios. For example, in personalized PageRank scenerio, peo-



(a) A Gaussian cluster. **(b)** The PDF of x-axis. **(c)** The PDF of y-axis.

Figure 1.4: The demonstration of the unimodality.



(a) Two Gaussian clusters. **(b)** The PDF of x-axis. **(c)** The PDF of y-axis.

Figure 1.5: The demonstration of the multimodality.

ple usually tend to like web pages that are heavily connected to their favorite ones; in content-based recommendation scenerio, people usually tend to like similar contents they have read before. Figure 1.6 shows the network derived from the books on US polictics around the time of the 2004 presidential election (the original network data can be found here[2]). The network has 105 vertices and 441 edges. Each vertex represents a book sold by Amazon and each edge represents frequent copurchasing of books by the same buyers. The books can be categorized into three types, i.e. "liberal" (green), "neutral" (red) and "conservative" (pink). We can see that similarity breeds connection [68]. If a buyer buys a liberal book, he/she tends to buy another liberal book; Likewise, if a buyer buys a conservative book, he/she tends to buy another conservative book. For the vertex classification in networks, one proper assumption of the network is that the network possesses a high

[2]http://www-personal.umich.edu/ mejn/netdata/

amount of homophily, i.e. similar vertices are more likely to be connected. In Chapter 6, we propose a semi-supervised learning framework called wvGN to infer the labels of the unlabeled vertices based on the theory of homophily.

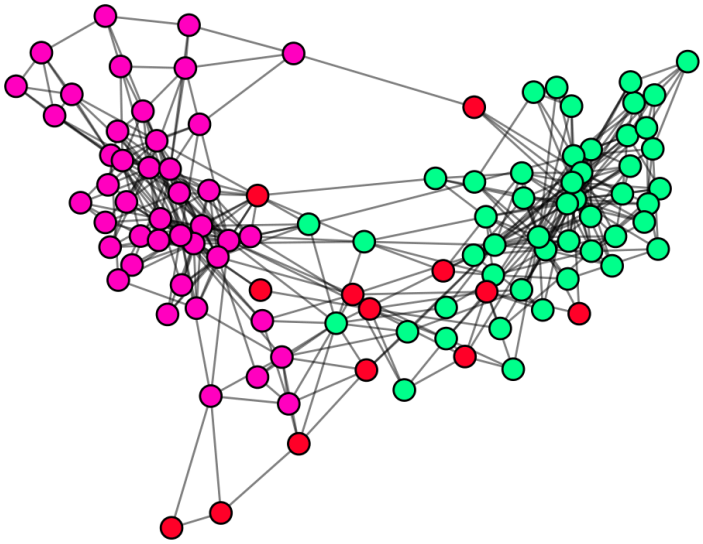


Figure 1.6: The demonstration of homophily in the Polbooks network.

1.4 Goals and Structures of this Thesis

To advance the state-of-the-art in solving three challenges in data mining, this publication-based thesis focuses on building novel methods that use concepts of independence, unimodality and homophily. The remainder of this thesis is organized as follows:

- Chapter 2 introduces notations that are used through this thesis and background techniques the proposed methods are based on.
- Chapter 3 presents a clustering technique called Full Spectral Clustering (FUSE) that exploits the cluster-separation information from all eigenvectors to deal with the multi-scale data which contains structures at different scales of size and density.
- Chapter 4 presents a subspace clustering method called ISAAC to detect multiple independent subspace clusters from high-dimensional data. The detected clusters are

non-redundant.

- Chapter 5 shows an effective and efficient clustering method for attributed graphs. The Hartigans' dip test and the normalized cut are combined in one framework to detect cohensive clusters which have densely connected edges and have as many unimodal attributes as possible.
- Chapter 6 proposes a semi-supervised learning method for the vertex classification in graphs. The proposed method is based on the theory of homophily.
- Chapter 7 gives concluding remarks, as well as directions for the future research.

Chapter 2

Background

In this chapter, we elaborate some background techniques this thesis is based on. Before giving their details, let me first give the main notations used through this thesis.

Notations. We use lower-case Roman letters (e.g. a, b) to denote scalars. Upper-case Roman letters (e.g. A, B) are used for continuous random variables. A random column-vector (list of random variables) of length d is denoted by an accented upper-case bold Roman letter (e.g. $\vec{\mathbf{A}} = (A_1, \ldots, A_d)^\intercal$). The probability density function (PDF) of a random variable A is denoted by $f_A(x)$. The joint PDF of random variables A and B is $f_{A,B}(x, y)$; for a random vector $\vec{\mathbf{A}}$ it is $f_{\vec{\mathbf{A}}}(\boldsymbol{x})$. We denote regular vectors (row) by boldface lower case letters (e.g. $\mathbf{x}$). Matrices are denoted by boldface upper case letters (e.g. $\mathbf{X}$). We denote entries in a matrix by non-bold lower case letters, such as x_{ij}. Row i of matrix $\mathbf{X}$ is denoted by the vector $\mathbf{x}_{i\cdot}$, column j by the vector $\mathbf{x}_{\cdot j}$. We use $[x_1, \cdots, x_n]$ to denote a row created by stacking n continuous random variables; similarly, we use $\mathbf{X} = [\mathbf{x}_1; \cdots; \mathbf{x}_m]$ to denote creating a matrix by stacking $\mathbf{x}_i$ along the rows. A set is denoted by calligraphic capital letters (e.g. $\mathcal{S}$). An undirected graph or network is denoted by $\mathsf{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a set of graph vertices and $\mathcal{E}$ is a set of graph edges. An undirected attributed graph or network is denoted by $\mathsf{G} = (\mathcal{V}, \mathcal{E}, \mathbf{F})$, where $\mathbf{F} \in \mathbb{R}^{n \times d}$ is a data matrix of attributes associated to vertices and d is the number of attributes. A graph cluster is a subset of vertices $\mathcal{S} \in \mathcal{V}$. The affinity matrix of the vertices is denoted by $\mathbf{A} \in \mathbb{R}^{n \times n}$

with $a_{ij} \geq 0$, $a_{ij} = a_{ji}$ (the graph is undirected). The degree matrix $\mathbf{D}$ is a diagonal matrix associated with $\mathbf{A}$ with $d_{ii} = \sum_j a_{ij}$. The normalized affinity matrix or random walk transition matrix $\mathbf{P}$ is defined as $\mathbf{D}^{-1}\mathbf{A}$. Thus, a normalized graph random-walk Laplacian matrix is $\mathbf{L} = \mathbf{I} - \mathbf{P} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ according to Meila and Shi [69], where $\mathbf{I}$ is an identity matrix. The indicator function is denoted by $\mathbb{1}(x)$. For a matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$, $diag(\mathbf{X}) \in \mathbb{R}^{n \times 1}$ is a vector created by extracting the diagonal of $\mathbf{X}$.

2.1 Independent Subspace Analysis

ISA [4] aims to find an orthogonal linear transformation for a given dataset which, upon application, yields several jointly-independent *source subspaces*, each containing one or more *source subspace components*. Components within the same subspace generally have high dependencies. The objective differs from Independent Component Analysis (ICA), which requires pairwise independence between *all* individual components. In the familiar context of the "Cocktail Party Problem", an ISA subspace is perhaps a string quartet, where dependency exists between the sounds produced by the ensemble members (the subspace components).

We assume in this paper that the data under consideration are realized from the random column-vector $\vec{\mathbf{X}} = (X_1, \ldots, X_m)^\intercal$. Our concrete data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is formed from n observations of this vector. ISA describes $\mathbf{X}$ as a linear mixture of sources, $\mathbf{X} = \mathbf{MS}$, where $\mathbf{M}$ is a mixing matrix and $\mathbf{S}$ corresponds to the sources. Without any loss of generality [110] we assume "whiteness", which means that 1) the data matrix $\mathbf{X}$ has been *whitened* by applying zero mean normalization and PCA (all variables are uncorrelated, with unit variance and zero expectation value), 2) the mixing matrix $\mathbf{M}$ is orthogonal and 3) the task is *complete* (implying that the mixing matrix is square and full rank, i.e. $\mathbf{M} \in \mathbb{R}^{m \times m}$). Inverting the system, we denote with $\mathbf{W} = \mathbf{M}^{-1}$ a "demixing" matrix such that $\mathbf{WX} = \mathbf{S} \in \mathbb{R}^{m \times n}$. The sources matrix $\mathbf{S}$ is understood to represent n observations of a random column-vector $\vec{\mathbf{S}} = (S_1, \ldots, S_m)^\intercal$, with each random variable S_i termed a *source subspace component*.

We express ISA's "pooling" of source subspace components into *source subspaces* through the use of q non-empty, pairwise-disjoint sets $\mathcal{S}_1, \ldots, \mathcal{S}_q$ (calligraphic font) of random variables. Together, these sets partition the random variables in $\vec{\mathbf{S}}$ (that is, $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_q = \{S_1, S_2, \ldots, S_m\}$). We denote with $\vec{\mathbf{S}}^1, \ldots, \vec{\mathbf{S}}^q$ each of the random vectors formed from the random variables of the corresponding subspaces, and hence we denote with $\mathbf{S}^1, \ldots, \mathbf{S}^q \in \mathbb{R}^{|\mathcal{S}_i| \times n}$ the sub-matrix of $\mathbf{S}$ which corresponds to the subspace $\mathcal{S}_i$.

ISA's task is to find the demixing matrix $\mathbf{W}$ such that the "independence" between source subspaces is maximized:

Definition 1 ***Independent Subspace Analysis [110].*** *Given a "data" matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the number q of subspaces to find and a vector $\boldsymbol{d} = (d_1, \ldots, d_q) \in \mathbb{N}^q$ holding the subspace dimensions ($\sum_{i=1}^{q} d_i = m$), find a demixing matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$ such that $\mathbf{S} = \mathbf{W}\mathbf{X}$ and the corresponding source subspaces $\mathcal{S}_1, \ldots, \mathcal{S}_q$ ($|\mathcal{S}_i| = d_i, \forall i = 1 \ldots q$) have minimal mutual information*

$$J_I(\mathbf{W}) := I(\mathcal{S}_1, \ldots, \mathcal{S}_q) \tag{2.1}$$

Various ISA algorithms exist which vary in terms of the applied cost function and the optimization technique [110]. Like the popular FastISA algorithm, the demixing matrix $\mathbf{W}$ is usually determined by iteratively updating its rows in a fixed-point manner. Unfortunately, FastISA and others dictate equal-sized subspaces ($|\mathcal{S}_1| = |\mathcal{S}_2| = \cdots = |\mathcal{S}_q|$). In our context this condition is too rigid – clusterings may well exist in subspaces of differing dimensionality. With [110] we choose in this work a variant which relaxes this requirement. In short, the approach makes use of the "ISA separation principle", which states that ISA can be solved by first performing ICA and then searching for groups of components such that the independence between those groups is maximized (the groups need not have an equal number of components). The only parameter this variant needs is the subspace cardinalities.

To better understand ISA, we show an example here in Figure 2.1. Figure 2.1(a) and (b) demonstrate two independent subspaces. Figure 2.1(c) and (d) are two axis-parellel subspaces of the mixed data shown in Figure 2.1(a) and (b). The elements of the mixing

matrix is uniformly distributed. Figure 2.1(e) and (f) show the two recovered independent subspaces by ISA. Compared with the data shown in Figure 2.1(a) and (b), we can see that the scale of the data shown in Figure 2.1(e) and (f) is changed because of the whitening. Compared with the data shown in Figure 2.1(b), we can also see that the data shown Figure 2.1(f) are rotated because ISA rotates the subspace to maximize the independence.

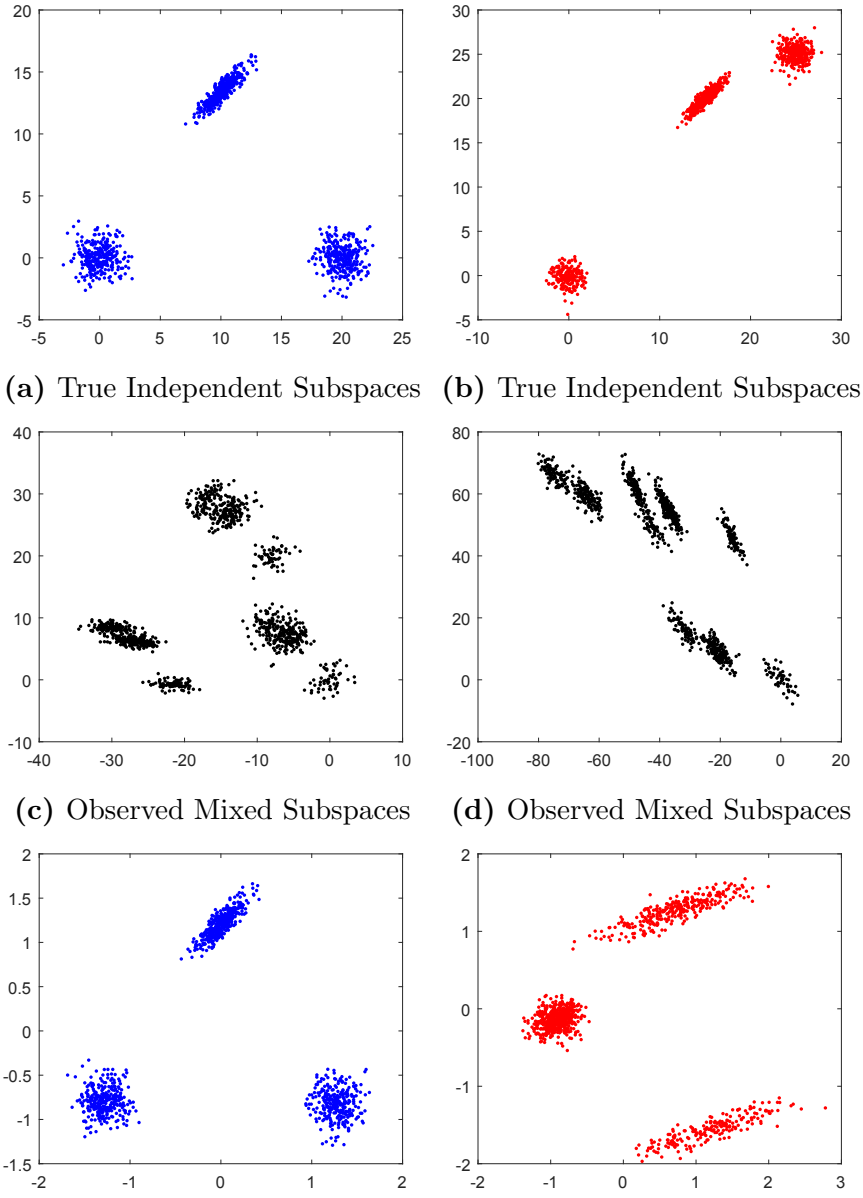
2.2 Independent Component Analysis

If the dimensionality of the sources is single, ISA becomes ICA [5]. Let $\mathbf{s}_1, \cdots, \mathbf{s}_m$ be m one-dimensional independent sources. Each has n i.i.d samples denoted by $\mathbf{s}_i = [s_{i1}, \cdots, s_{in}](1 \leq i \leq m)$. Let $\mathbf{S} = [\mathbf{s}_1; \cdots; \mathbf{s}_m] \in \mathbb{R}^{m \times n}$ and we assume $\mathbf{S}$ is hidden and only a matrix $\mathbf{X}$ of mixed sources can be observed. The task of ICA is to find a demixing matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$ such that $\mathbf{S} = \mathbf{WX}$ and every two components $\mathbf{s}_i$ and $\mathbf{s}_j$ $(1 \leq i, j \leq m, i \neq j)$ are statistically independent. Without loss of generality, we assume data has been whitened, which means (1) the expectation value is zero and the covariance matrix is the identity matrix ($\mathbf{I}$), (2) the demixing matrix is square, orthogonal ($\mathbf{W} \cdot \mathbf{W} = \mathbf{I}$) and full rank. To better understand ICA, we show here an example in Figure 2.2. Figure 2.2(a) shows three true signals: sinusoid, square, and sawtooth. Figure 2.2(b) shows three mixed signals which are generated by introducing uniform mixing weights into the true signals. Figure 2.2(a) shows three recovered signals (independent components). Compared with the true signals, the recovered signals have the same frequency, but the amplitudes may be different.

2.3 Normalized Cut

The definition of the widely used *normalized cut* [56] objective function is:

$$\mathrm{NCut}(\mathcal{S}) = \frac{\mathrm{cut}(\mathcal{S}, \overline{\mathcal{S}})}{\mathrm{vol}(\mathcal{S})} \quad . \tag{2.2}$$



(a) True Independent Subspaces **(b)** True Independent Subspaces

(c) Observed Mixed Subspaces **(d)** Observed Mixed Subspaces

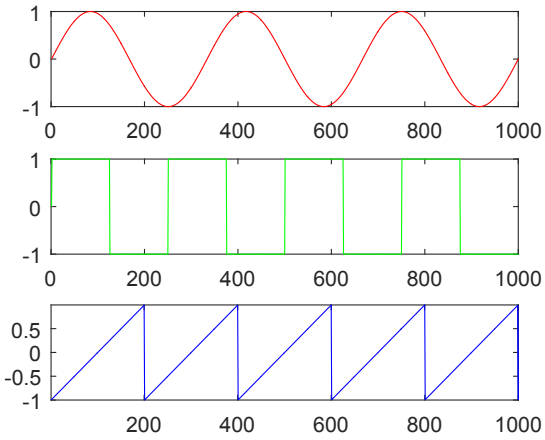
(e) Recovered Independent Subspaces **(f)** Recovered Independent Subspaces

Figure 2.1: ISA.

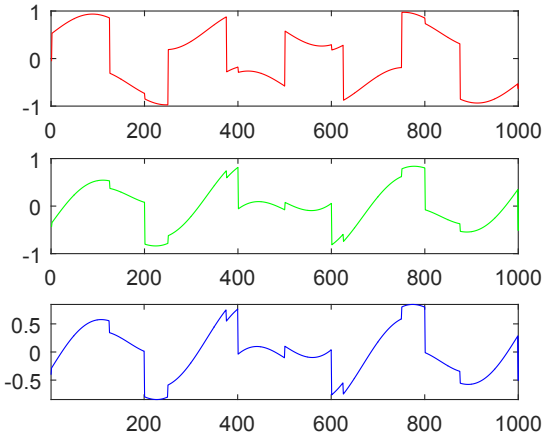
where $\text{cut}(\mathcal{S}, \overline{\mathcal{S}}) = \sum_{v_i \in \mathcal{S}, v_j \in \overline{\mathcal{S}}} a_{ij}$ and $\text{vol}(\mathcal{S}) = \sum_{v_i \in \mathcal{S}, v_j \in \mathcal{V}} a_{ij}$.

Equation 2.2 can be equivalently rewritten as (for a more detailed explanation, please refer to [97]):

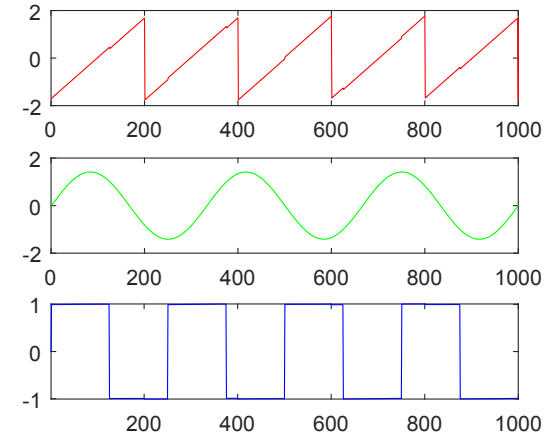
$$\text{NCut}(\mathcal{S}) = \mathbf{u}^\intercal \mathbf{L}\mathbf{u}, s.t.\ \mathbf{u}^\intercal \mathbf{D}\mathbf{u} = \text{vol}(\mathsf{G}),\ \mathbf{D}\mathbf{u} \perp \mathbf{1}\ . \tag{2.3}$$



(a) True Signals



(b) Observed Mixed Signals



(c) Independent Components

Figure 2.2: ICA.

where $\mathbf{u}$ is the cluster indicator vector and $\mathbf{u}^\intercal\mathbf{L}\mathbf{u}$ is the cost of the cut and $\mathbf{1}$ is a constant vector whose entries are all 1. Note that finding the optimal solution is known to be NP-hard [25] when the values of $\mathbf{u}$ are constrained to $\{1, -1\}$. But if we relax the objective function to allow it take values in $\mathbb{R}$, a near optimal partition of the graph G can be derived from the second smallest eigenvector of $\mathbf{L}$. More generally, k eigenvectors with the k smallest eigenvalues partition the graph into k subgraphs with near optimal normalized cut value.

2.4 Truncated Power Iteration

Although spectral clustering has gained its popularity and success in data mining and machine learning fields, its high time complexity ($\mathcal{O}(n^3)$ for computing the eigenvectors of the graph Laplacian matrix $\mathbf{L}$) limits its practical use in real-world data. To address the difficulty, Lin and Cohen [37] used truncated power iteration to find a pseudo-eigenvector on the normalized affinity matrix $\mathbf{W}$ with time complexity $\mathcal{O}(n)$, which is very efficient and attractive. Note that the k largest eigenvectors of $\mathbf{P}$ are also the k smallest eigenvectors of $\mathbf{L}$. Power Iteration (PI) is an efficient and popular method to compute the dominant eigenvector of a matrix. PI starts with a random vector $\mathbf{v}^0$ and iteratively updates as follows [37],

$$\mathbf{v}^t = \frac{\mathbf{P}\mathbf{v}^{t-1}}{\|\mathbf{P}\mathbf{v}^{t-1}\|_1} \tag{2.4}$$

Suppose $\mathbf{P}$ has eigenvectors $\mathbf{U} = [\mathbf{u}_1; \mathbf{u}_2; \cdots; \mathbf{u}_n]$ with eigenvalues $\mathbf{\Lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_n]$, where $\lambda_1 = 1$ and $\mathbf{u}_1$ is constant. We have $\mathbf{P}\mathbf{U} = \mathbf{\Lambda}\mathbf{U}$ and in general $\mathbf{P}^t\mathbf{U} = \mathbf{\Lambda}^t\mathbf{U}$. When ignoring renormalization, Equation 2.4 can be written as

$$\begin{aligned} \mathbf{v}^t &= \mathbf{P}\mathbf{v}^{t-1} = \mathbf{P}^2\mathbf{v}^{t-2} = \cdots = \mathbf{P}^t\mathbf{v}^0 \\ &= c_1\mathbf{P}^t\mathbf{u}_1 + c_2\mathbf{P}^t\mathbf{u}_2 + \cdots + c_n\mathbf{P}^t\mathbf{u}_n \\ &= c_1\lambda_1^t\mathbf{u}_1 + c_2\lambda_2^t\mathbf{u}_2 + \cdots + c_n\lambda_n^t\mathbf{u}_n \end{aligned} \tag{2.5}$$

According to Equation 2.5, we have

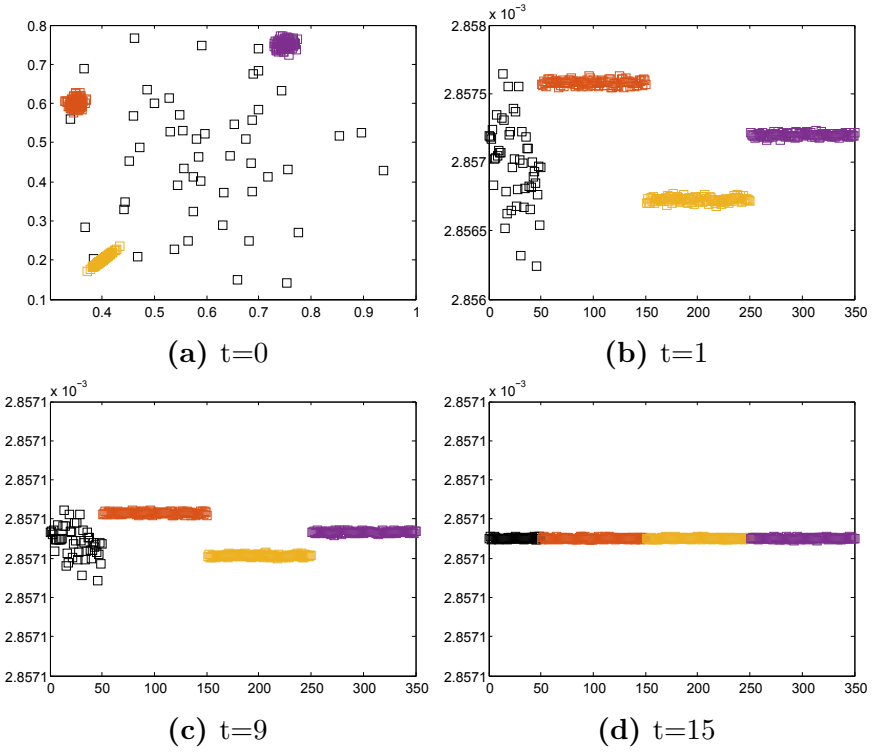
$$\frac{\mathbf{v}^t}{c_1\lambda_1^t} = \mathbf{u}_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^t \mathbf{u}_2 + \cdots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^t \mathbf{u}_n \tag{2.6}$$

So the convergence rate of PI towards the dominant eigenvector $\mathbf{u}_1$ depends on the significant terms $\left(\frac{\lambda_i}{\lambda_1}\right)^t$ $(2 \leqslant i \leqslant n)$. PI will finally converge to the dominant eigenvector $\mathbf{u}_1$ which is of little use in clustering. PIC [37] defines the velocity at t to be the vector $\boldsymbol{\delta}^t = \mathbf{v}^t - \mathbf{v}^{t-1}$ and defines the acceleration at t to be the vector $\boldsymbol{\epsilon}^t = \boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}$ and stops PI when $\|\boldsymbol{\epsilon}^t\|_{max}$ is below a threshold $\hat{\epsilon}$.

Here let me use Figure 2.3 to demonstrate the mechanism of the truncated power iteration. Figure 2.3(a) shows the data which contains four clusters. Figure 2.3(b) demonstrates the pseudo-eigenvector generated by the power iteration method at the step one. We can see that the four clusters are well separated in this pseudo-eigenvector. Figure 2.3(d) shows the pseudo-eigenvector at the step fifteen. We can see that the pseudo-eigenvector has converged and is of on use in clustering. Figure 2.3(c) shows the pseudo-eigenvector after the convergence of the truncated power iteration. This pseudo-eigenvector will be fed into k-means to find clusters.

2.5 The Dip Test

We apply a univariate statistic hypothesis test for unimodality called Hartigans' dip test [51] on the vertex attributes to measure the degree of the unimodality of a graph cluster in Chapter 5. The dip measures the departure of a distribution from unimodality. Before introducing the concept of the dip test, let me first introduce the concepts of the greatest convex minorant (g.c.m) and the least concave majorant (l.c.m.). The greatest convex minorant of $F(x)$ in $(-\infty, x_l]$ is sup $G(x)$ for $x \leq x_l$, where the sup is taken over all functions G that are convex in $(-\infty, x_l]$ and nowhere greater than $F(x)$. The least concave majorant of $F(x)$ in $[x_u, \infty)$ is inf $L(x)$ for $x \geq x_u$, where the inf is taken over all functions L that are concave in $[x_u, \infty)$ and nowhere less than $F(x)$.



(a) t=0 **(b)** t=1 **(c)** t=9 **(d)** t=15

Figure 2.3: Truncated power iteration.

Let $\mathcal{U}$ be the set of all unimodal distributions, the dip test of the distribution function $F(x)$ is computed as follows,

$$D(F) = \inf_{U \in \mathcal{U}} \sup_{x} |F(x) - U(x)| \tag{2.7}$$

The dip test is the infimum among the supremum computed between the cumulative distribution function (CDF) of F and the CDF of U from the set of unimodal distributions. The computation of the dip test is: Let $F(x)$ be an empirical distribution function for the sorted samples $x_1, ..., x_n$. There are $n \cdot (n-1)/2$ candidate modal intervals. Compute for each candidate $[x_i, x_j], i \leq j \leq n$ the g.c.m. of $F(x)$ in $(-\infty, x_i]$ and the l.c.m. of $F(x)$ in $[x_j, \infty)$ and let $d_{i,j}$ be the maximum distance of F to these computed curves (g.c.m. and l.c.m.). Finally, it selects the modal interval with the maximum distance which is the twice of the dip test. For more details, please refer to [7, 51].

As pointed out in [51], the class of uniform distributions is the most suitable for the

null hypothesis, because their dip test values are stochastically larger than those of other unimodal distributions. The p-value for the unimodality test is then computed by comparing $D(F)$ with $D(U^r)$ b times, each time with a different n observations from U, and the proportion $\sum_{1\leq r\leq b} \mathbb{1}(D(F) \leq D(U^r))/b$ is the p-value. If the p-value is greater than a significance level α, say 0.05, the null hypothesis H_0 that F is unimodal is accepted.

2.6 Kernel Density Estimation

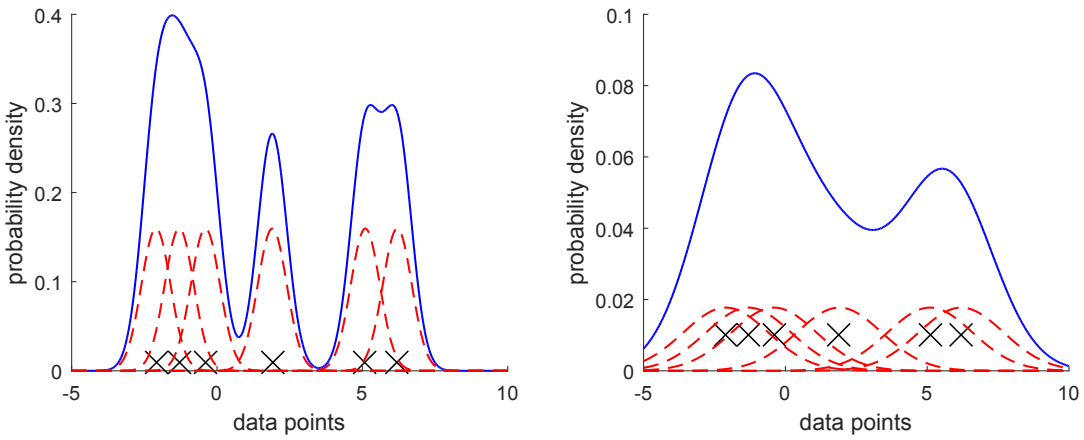
We use Kernel Density Estimation (KDE) with a Gaussian kernel for estimating probability density functions. For a vector $\boldsymbol{x} \in \mathbb{R}^d$ observed from a random vector $\vec{\mathbf{A}}$ of length d, the zero-mean Gaussian kernel with covariance matrix $\boldsymbol{\Sigma}$ is defined as

$$G(\boldsymbol{x}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\boldsymbol{x}^\intercal \boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right) \tag{2.8}$$

With such a kernel, KDE for the random vector $\vec{\mathbf{A}}$ finds an approximate value $\widehat{f}_{\vec{\mathbf{A}}}(\boldsymbol{x})$ to the true density value $f_{\vec{\mathbf{A}}}(\boldsymbol{x})$ at point $\boldsymbol{x}$ by summing contributions from Gaussians centered at each of the n observations in $\mathbf{A}$,

$$\widehat{f}_{\vec{\mathbf{A}}}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} G(\mathbf{x} - \mathbf{a}_{\cdot i}, \boldsymbol{\Sigma}) \tag{2.9}$$

One parameter related to KDE is the bandwidth. To demonstrate how the bandwidth affects the estimated probability density function of given data, we use Figure 2.4 as an example. We can see that when increasing the value of bandwidth, the estimated proability density function becomes more smooth. In this thesis, we choose elliptical Gaussian kernels, implying a diagonal $\boldsymbol{\Sigma}$ with positive bandwidth entries $\sigma_{j,j}$. We select bandwidths according to [16]: $\sigma_{j,j} = 0.9 \cdot n^{-1/(d+4)} \cdot \min(\alpha_j, \beta_j/1.34)$, where α_j and β_j are the variance and the inter-quartile range of the j^{th} dimension of all n observations respectively.



(a) σ=0.5 **(b)** σ=1.5

Figure 2.4: Kernel Density Estimation (Gaussian kernels with different bandwidths).

2.7 Support Vector Machines

Support Vector Machines (SVM) aim at finding a hyperplane in a mapping space (by a mapping function $\Phi(\mathbf{x})$) to maximize the margin between two classes. Mathematically speaking, the goal of SVM is to derive a discriminant function $f(\mathbf{x})$ such that the margin between support vectors from different classes are maximized. For all hyperplanes which separate classes perfectly, we prefer the one which has the maximized distance. In addition, we prefer the hyperplane which makes each point be on the correct side of the boundary, i.e. $f(\mathbf{x}_i) \cdot y_i > 0$, where $y_i \in \{\pm 1\}$. The objective function is:

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} \\ &\text{s.t. } y_i(\mathbf{w}^\intercal\mathbf{x}_i + b) \geq 1, i = 1, \cdots, n \end{aligned} \tag{2.10}$$

where $\frac{1}{2}$ is added for the mathematical convenience.

To take account the data points that do not satisfy $y_i(\mathbf{w}^\intercal\mathbf{x}_i + b) \geq 1$, we introduce slack variables $\xi_i \geq 0$ to make the following hold: 1) $\xi_i = 0$ if the data point is on or inside the correct margin boundary, 2) $\xi_i = |y_i - (\mathbf{w}^\intercal\mathbf{x}_i + b)|$ otherwise. Hence, the data point is in the correct side of the decision boundary if $0 \leqslant \xi_i < 1$ and the data point situates on the

wrong side of the decision boundary if $\xi_i > 1$.

We replace the hard constraints $y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1$ with the soft margin constraint $y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1 - \xi_i$ and the objective function becomes:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^\intercal \mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ &\text{s.t. } \xi_i \geq 0, y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, \cdots, n \end{aligned} \tag{2.11}$$

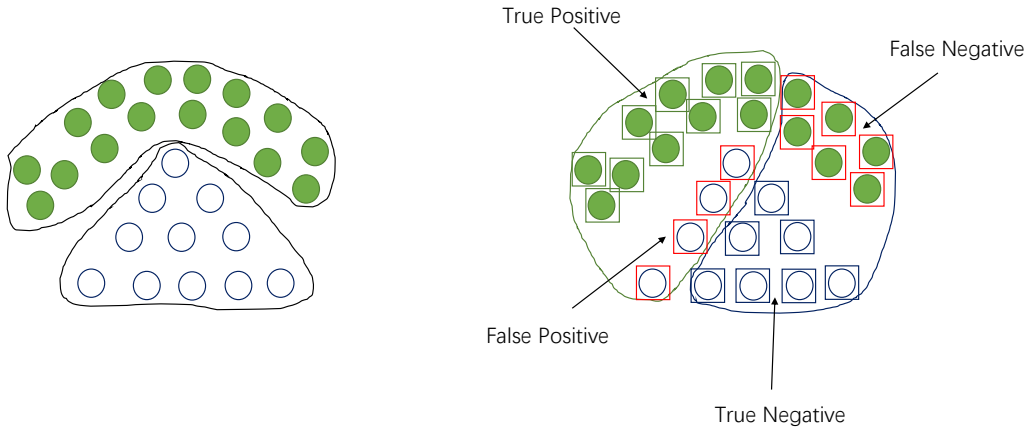
where C is a regularization parameter which is used for controlling the tolerance number of errors. Because $\xi_i > 1$ means data point i is misclassified, $\sum_{i=1}^{n} \xi_i$ can be interpreted as an upper bound on the number of misclassified data points. In this thesis, we focus on linear SVM.

2.8 Cluster Evaluation

"*The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.*" [6].

2.8.1 Precision, Recall and F1-score

First, let me use Figure 2.5 to denote the concepts of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) that compare the results of the clusterer/classifier under test with trusted external judgments. Figure 2.5(a) shows the external judgment (ground truth) of the data. Figure 2.5(b) is the clustering results of some clustering method. Accuracy, presicion and recall are then defined as: Precision $= \frac{TP}{TP+FP}$, Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$, Recall $= \frac{TP}{TP+FN}$. And the F1-score is defined as the harmonic mean of precision and recall: F1 $= \frac{2}{\frac{1}{\text{Precision}}+\frac{1}{\text{Recall}}} = \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$



(a) The ground truth. **(b)** The clustering result.

Figure 2.5: The demonstration for the concepts of true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

Macro-F1 score is the average F1-score over classes:

$$\text{Macro-F1} = \frac{1}{c}\sum_{i=1}^{c} \text{F1}^i$$

where c is the number of classes, and $\text{F1}^i = \frac{2 \cdot \text{Precision}^i \cdot \text{Recall}^i}{\text{Precision}^i + \text{Recall}^i}$ is the F1-score of the i-th class.

Micro-F1 [80] is computed as follows:

$$\text{Micro-F1} = \frac{\sum_{i=1}^{c} 2}{\sum_{i=1}^{c} \left(\frac{1}{\text{Precision}^i} + \frac{1}{\text{Recall}^i} \right)}$$

Macro-F1 [80] is more affected by the rare classes because all classes are weighted evenly while Micro-F1 is affected more by the common classes because it weights objects evenly.

2.8.2 Rand Index

Given a set of n objects $\mathcal{S} = \{o_1, \cdots, o_n\}$ and two partitions of $\mathcal{S}$ to compare, $\mathcal{X} = \{\mathcal{X}_1, \cdots, \mathcal{X}_r\}$ and $\mathcal{Y} = \{\mathcal{Y}_1, \cdots, \mathcal{Y}_s\}$. Now the TP, FP, TN, and FN are defined as follows:

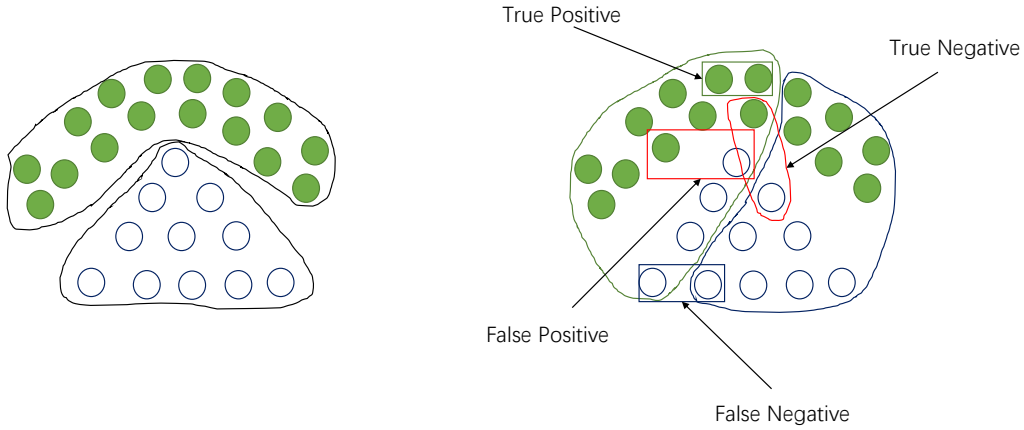
- TP: the number of pairs of elements in $\mathcal{S}$ that are in the same subset in $\mathcal{X}$ and also

in the same subset in $\mathcal{Y}$.

- FP: the number of pairs of elements in $\mathcal{S}$ that are not in the same subsets in $\mathcal{X}$ but in the same subset in $\mathcal{Y}$.

- TN: the number of pairs of elements in $\mathcal{S}$ that are not in the same subsets in $\mathcal{X}$ and also not in the same subsets in $\mathcal{Y}$.

- FN: the number of pairs of elements in $\mathcal{S}$ that are in the same subset in $\mathcal{X}$ but not in the same subsets in $\mathcal{Y}$.



(a) The ground truth. **(b)** The clustering result.

Figure 2.6: The demonstration for computing Rand Index.

The Rand Index R is defined as: $R = \frac{TP+TN}{TP+FP+TN+FN} = \frac{TP+TN}{n\cdot(n-1)/2}$. Before introducing the Adjusted Rand Index (ARI) [62], let me first summarize the overlap between $\mathcal{X}$ and $\mathcal{Y}$ in a contingency table where each element denotes the number of objects in common between $\mathcal{X}_i$ and $\mathcal{Y}_j$.

The ARI is defined as follows:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left(\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right)/\binom{n}{2}}{\frac{1}{2}\left(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right) - \left(\sum_i \binom{a_i}{2}\cdot\sum_j \binom{b_j}{2}\right)/\binom{n}{2}}$$

Table 2.1: The Contingency Table.

	$\mathcal{Y}_1$	$\mathcal{Y}_2$	$\dots$	$\mathcal{Y}_s$	Sums
$\mathcal{X}_1$	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
$\mathcal{X}_2$	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathcal{X}_r$	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
Sums	b_1	b_2	$\cdots$	b_s	

2.8.3 Purity

Before computing purity, each cluster is assigned to the class which dominates in the cluster. And the purity is computed as follows:

$$\text{Purity}(\mathcal{X}, \mathcal{Y}) = \frac{1}{n} \cdot \sum_{i=1}^{r} \max_{j \leqslant s} |\mathcal{X}_i \cap \mathcal{Y}_j|$$

The value of Purity ranges from 0 to 1. The higher the value is, the better the clustering is. One problem of purity is that if each object is a cluster, the value of Purity will be 1. Thus, Purity is usually used to evaluate the clustering with equal number of clusters to those of the ground truth.

2.8.4 Mutual Information

In information theory, the mutual information of two random variables maesures the amount of information they have in common. Unlike the correlation which can only capture the linear relationships between two random variables, mutual information can capture non-linear or high-order relationships. For two partitions or clusterings $\mathcal{X}$ and $\mathcal{Y}$, their mutual information is computed as follows:

$$I(\mathcal{X}; \mathcal{Y}) = \sum_{i=1}^{r} \sum_{j=1}^{s} p\left(\mathcal{X}_i \cap \mathcal{Y}_j\right) \cdot \log \left(\frac{p\left(\mathcal{X}_i \cap \mathcal{Y}_j\right)}{p(\mathcal{X}_i) \cdot p(\mathcal{Y}_j)} \right)$$

The normalized mutual information (NMI) [10] is defined as follows:

$$NMI(\mathcal{X};\mathcal{Y}) = \frac{I(\mathcal{X};\mathcal{Y})}{\sqrt{H(\mathcal{X}) \cdot H(\mathcal{Y})}}$$

where $H(\mathcal{X})$ and $H(\mathcal{Y})$ are the entropy of $\mathcal{X}$ and $\mathcal{Y}$, respectively. The definition of the entropy is:

$$H(\mathcal{X}) = -\sum_{i=1}^{r} p(\mathcal{X}_i) \cdot \log\left(p(\mathcal{X}_i)\right)$$

The adjusted mutual information (AMI) [73] is proposed to correct the measures for randomness. AMI subtracts the expectation value of the mutual information, so that the AMI is zero when two different distributions are random, and one when two distributions are identical. It is defined as follows:

$$AMI(\mathcal{X};\mathcal{Y}) = \frac{I\left(\mathcal{X};\mathcal{Y}\right) - E\left(I(\mathcal{X};\mathcal{Y})\right)}{\max\left(H(\mathcal{X}), H(\mathcal{Y}) - E\left(I(\mathcal{X};\mathcal{Y})\right)\right)}$$

where $E\left(I(\mathcal{X};\mathcal{Y})\right)$ is the expectation value of the mutual information between $\mathcal{X}$ and $\mathcal{Y}$.

Chapter 3

Full Spectral Clustering

Multi-scale data which contains structures at different scales of size and density is a big challenge for spectral clustering. Even given a suitable locally scaled affinity matrix, the first k eigenvectors of such a matrix still cannot separate clusters well. Thus, in this chapter, we exploit the fusion of the cluster-separation information from all eigenvectors to achieve a better clustering result. Our method **FU**ll **S**pectral Clust**E**ring (FUSE) is based on Power Iteration (PI) and Independent Component Analysis (ICA). PI is used to fuse all eigenvectors to one pseudo-eigenvector which inherits all the cluster-separation information. To conquer the cluster-collision problem, we utilize PI to generate p ($p > k$) pseudo-eigenvectors. Since these pseudo-eigenvectors are redundant and the cluster-separation information is contaminated with noise, ICA is adopted to rotate the pseudo-eigenvectors to make them pairwise statistically independent. To let ICA overcome local optima and speed up the search process, we develop a self-adaptive and self-learning greedy search method. Finally, we select k rotated pseudo-eigenvectors (independent components) which have more cluster-separation information measured by *kurtosis* for clustering. Various synthetic and real-world data verifies the effectiveness and efficiency of our FUSE method.

Parts of the materials presented in this chapter have been published in [100], where Wei Ye devised the main concept, did the most parts of the experimental evaluation and wrote the major parts of the paper; Christian Böhm and Claudia Plant supervised the

project and contributed to the concept development and paper writing; Sebastian Goebl contributed to the paper writing and some experimental evaluation.

"*Wei Ye, Sebastian Goebl, Claudia Plant, Christian Böhm. FUSE: Full Spectral Clustering. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 1985–1994, 2016.*"

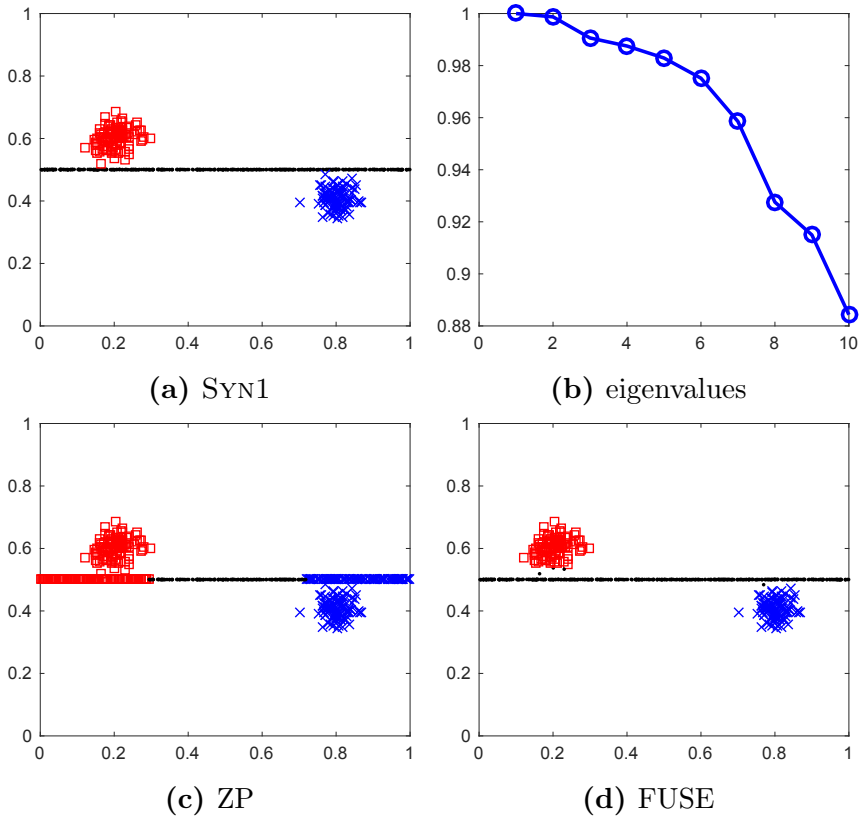
3.1 Motivation

Clustering is a basic technique in data analysis and mining. Two commonly used methods are k-means and Expectation Maximization clustering (EM) which pre-assume that data fits a Gaussian model. Such model-based clustering methods perform well if data fits the model. However, in most cases, we do not know the distribution of data. It is hard to decide which model to adopt. Spectral clustering, on the other hand, does not pre-assume any model. It only uses local information (point to point similarity) to achieve global clustering. Thus it is very elegant and popular in data ming and machine learning. Spectral clustering transforms the clustering of a set of data points with pairwise similarities into a graph partitioning problem, i.e., partitioning a graph such that the intra-group edge weights are high and the inter-group edge weights are low. There are three kinds of similarity graphs, i.e., the ε-neighborhood graph, the k-nearest neighbor graph and the fully connected graph [97]. Luxburg [97] emphasized that "*theoretical results on the question how the choice of the similarity graph influences the spectral clustering result do not exist*". However, the parameters (ε,k,σ) of these similarity graphs highly affect the clustering results, especially in cases where data contains structures at different scales of size and density. One usually used objective function in spectral clustering is *normalized cut* [56]. As pointed out in [13], the *normalized cut* criterion does not always work even given a proper affinity matrix.

Consider three clusters of different geometry shapes and densities in Figure 3.1(a). Both Gaussian clusters have 100 data points. The rectangular stripe cluster has 400 data points sampled from a uniform distribution. Conventional spectral clustering algorithms tend to

fail on this multi-scale data. Self-tuning spectral clustering (ZP) [66] proposes to use the locally scaled affinity matrix to solve the limitation. Further, ZP rotates the eigenvectors to create the maximally sparse representation to estimate the number of clusters automatically. However, such proposals still do not work on multi-scale data because of the unsuitability of the *normalized cut* criterion only using local information. Such an argument can be inferred from Figure 3.1(c). ZP fails to correctly separate the three clusters. Both cuts are along the stripe. Intuitively, it is not difficult to understand. The *normalized cut* criterion tries to make clusters "balanced" as measured by the number of vertices or edge weights. Since each of the two Gaussian clusters only has 100 data points and they are so close to the stripe cluster, cuts between the Gaussian clusters and the stripe cluster have a higher penalty than those along the stripe.

Differing from other spectral clustering algorithms, our method combines the cluster-separation information from all eigenvectors to achieve a better clustering result. As can be seen from Figure 3.1(d), only some controversial data points lying on the boundaries are clustered incorrectly. The fusion of the cluster-separation information from all eigenvectors is accomplished by exploiting truncated Power Iteration (PI). To yield good clustering, spectral clustering uses the first k eigenvectors of the graph Laplacian matrix. Similarly, we use PI to generate p ($p > k$) pseudo-eigenvectors. Each pseudo-eigenvector is a linear combination of all original eigenvectors, including the information not only from the "informative" eigenvectors but from the "noise" eigenvectors. Note that the pseudo-eigenvectors are redundant to each other. One main question is how to make the information from the "informative" eigenvectors stand out and suppress the information from the "noise" eigenvectors? In this paper, we use Independent Component Analysis (ICA) to reduce the redundancy, i.e., to make the pseudo-eigenvectors statistically independent (non-redundant) to each other. After whitening (more details in Section 3.2.1), ICA rotates the pseudo-eigenvectors to find the direction in which the entropy is minimized. Subsequently, a *kurtosis*-based selection strategy is exploited. Such a minimum-entropy rotation plus a *kurtosis*-based selection improve the cluster separation. The contributions are as follows:



(a) SYN1 **(b)** eigenvalues

(c) ZP **(d)** FUSE

Figure 3.1: Clustering results of ZP and FUSE on our SYN1 data ((b) gives the top 10 eigenvalues of the normalized affinity matrix).

- **We achieve the eigenvector-fusion by using Power Iteration (PI)**. The generated pseudo-eigenvectors include information from all eigenvectors.

- **We improve the cluster separation by applying ICA combined with a *kurtosis*-based selection strategy**. Since the generated pseudo-eigenvectors are redundant to each other, which is not beneficial to good clustering, we apply ICA to make them statistically independent. Then, a *kurtosis*-based selection strategy is exploited to improve the cluster separation. To the best of our knowledge, we are the first to apply ICA on spectral clustering.

- **We develop a greedy search method to render searching for statistically independent components more efficient and effective**. The greedy search strategy discriminates the search order to let ICA not easily get trapped into local

optimal. In addition, during the search process, the greedy search makes use of self-adaptive and self-learning strategies to balance the efficiency and effectiveness.

3.2 Full Spectral Clustering

3.2.1 Fusing eigenvectors

For real-world data, a single pseudo-eigenvector is not enough when the number of clusters is large. The reason is we need more eigenvectors to discriminate clusters when the cluster count increases. Thus, the cluster-collision problem may happen on one-dimensional pseudo-eigenvector. However, using PI p ($p > k$) times with random generated starting vectors to generate p pseudo-eigenvectors is not sufficient either, which can only alleviate the situation a little because of the redundant information provided by these pseudo-eigenvectors. It is just the first step of the eigenvector fusion. We also need to reduce the redundancy in these pseudo-eigenvectors. Our goal is twofold: 1) generate p pseudo-eigenvectors, 2) reduce redundancy to make the cluster-separation information stand out and suppress the noise information. The goal can also be rephrased as fusing the cluster-separation information from all original eigenvectors to improve clustering. Why do we need to fuse the information from all eigenvectors? The analysis in [13] shows that "*when confronted with clusters of different scales, corresponding to a multi-scale landscape potential, standard spectral clustering which uses the first k eigenvectors to find k clusters will fail*". Even given a locally scaled affinity matrix [66], ZP still cannot overcome the limitation if clusters have comparable densities.

For example, Figure 3.2(a) demonstrates the eigenvector space (consists of the eigenvectors associated with the top three minimum eigenvalues) found by ZP (other spectral clustering algorithms yield similar ones) on the running example (SYN1) in Section 3.1. It demonstrates that the three clusters are connected together in the eigenvector space, which is the reason for k-means' difficulty in separating them. The cluster-separation information is not provided by the first three eigenvectors. However, we can see from Figure 3.2(b)

that the blue and red clusters have fewer overlapped data points with the black cluster. If we fuse the information from the eigenvectors in Figure 3.2(b) to those in Figure 3.2(a), we can achieve a better clustering result. In this paper, we use truncated Power Iteration (PI) to fuse eigenvectors. Figure 3.2 (c) shows the four pseudo-eigenvectors returned by running PI four times with randomly generated starting vectors. The resulting pseudo-eigenvectors are very similar and redundant, e.g., the pseudo-eigenvectors $\mathbf{v}_1^t$ and $\mathbf{v}_4^t$. Thus, the cluster-separation information is not standing out. Figure 3.2(d) gives the pseudo-eigenvector space returned by our algorithm. In such space, the blue and red clusters have much fewer close data points to the black cluster compared to those in Figure 3.2(a), which makes k-means easily distinguish them from each other.

Consider that each pseudo-eigenvector generated by PI is a linear combination of all eigenvectors of the normalized affinity matrix $\mathbf{P}$ and every pair of distinct pseudo-eigenvectors is redundant. One way to reduce redundancy is to make p pseudo-eigenvectors statistically independent, which can be accomplished by ICA. Mathematically speaking, the problem formulation is as follows:

Definition 2 ***Statistically Independent Pseudo-eigenvectors.*** *Given a pseudo-eigenvector matrix $\mathbf{V} \in \mathbb{R}^{p\times n}$ generated by running PI p times, find a demixing matrix $\mathbf{W} \in \mathbb{R}^{p\times p}$ such that $\mathbf{E} = \mathbf{WV}$ and the sum of mutual information between pairwise components of $\mathbf{E}$ is minimized, where $\mathbf{E} \in \mathbb{R}^{p\times n}$ is a resulting independent pseudo-eigenvector matrix.*

$$J_I(\mathbf{W}) := \min \sum_{1\le i,j\le p, i\neq j} I(\mathbf{e}_{i\cdot}; \mathbf{e}_{j\cdot}) \tag{3.1}$$

The demixing matrix $\mathbf{W}$ can be derived by determining the directions of minimal entropy. To find such directions, ICA requires to whiten data, i.e., the expectation value of data is zero and the covariance matrix of data is the identity matrix, by applying PCA. The demixing matrix $\mathbf{W}$ is orthonormal in white space. After whitening data, ICA finds those directions of minimal entropy by rotating the whitened data. In this paper, instead of using fastICA [3], we use Jacobi ICA [30, 88] to find statistically independent pseudo-eigenvectors for the reasons that 1) we can choose different kinds of contrast functions,

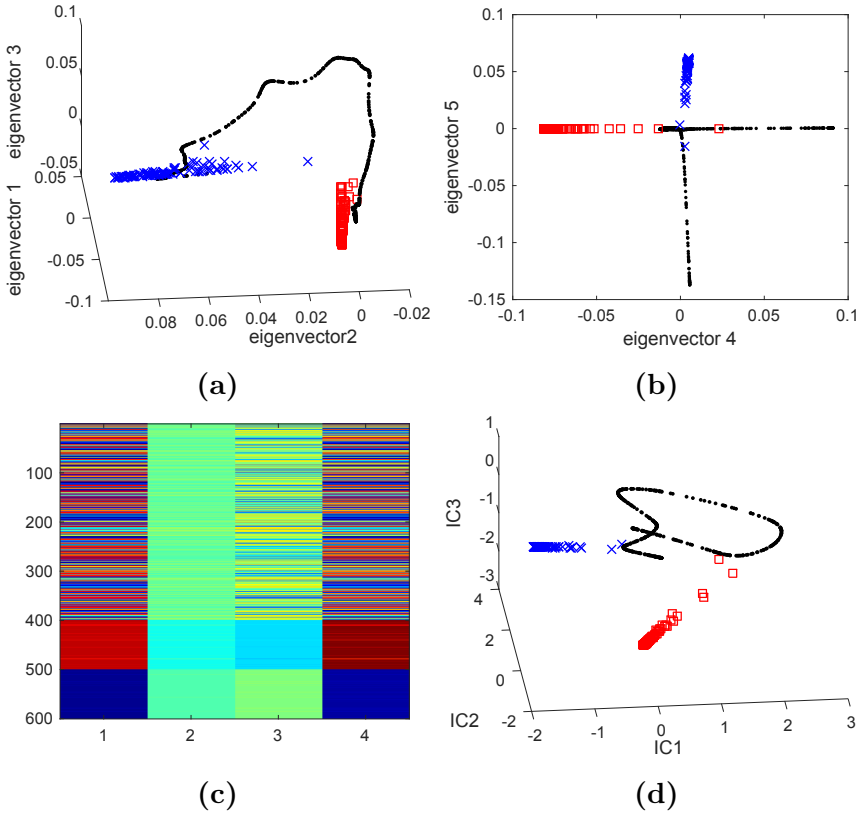


Figure 3.2: Demonstration of the (pseudo-)eigenvector space generated by ZP and FUSE on SYN1 data. (a) the eigenvector space (consists of the first three eigenvectors) returned by ZP, (b) the eigenvector space consists of the fourth and fifth eigenvectors returned by ZP, (c) four pseudo-eigenvectors generated by running PI four times with random initial vectors, (d) the pseudo-eigenvector space returned by FUSE, in which the clusters are well separated.

2) we can make it escape from local optima more easily. Note that Equation 3.1 can be solved by iteratively optimizing every pairwise mutual information. We rewrite Equation 3.1 as the following objective function:

$$\begin{aligned} &\min I(\mathbf{e}_{i\cdot};\mathbf{e}_{j\cdot}) \\ &\text{s.t.} \quad \mathbf{E} = \mathbf{WV},\ 1 \le i, j \le p,\ i \ne j \end{aligned} \tag{3.2}$$

Now it comes to how to select k independent components. Since ICA is interested in searching for non-Gaussian directions, in which negentropy is minimized. Non-Gaussian may be super-Gaussian as well as sub-Gaussian. We are only interested in sub-Gaussian

directions, in which clusters are as much separated as possible. Such directions can be best modeled by uniform distributions [17]. In this paper, we use *kurtosis* to measure the distance of the probability distribution of an independent component to Gaussian distribution. The kurtosis is the fourth *standardized moment*, defined as,

$$\text{Kurt}(X) = \frac{\mu_4}{\sigma^4} = \frac{E\left((X-\mu)^4\right)}{\left(E\left((X-\mu)^2\right)\right)^2} \tag{3.3}$$

where μ_4 is the fourth moment of the mean and σ is the standard deviation.

The kurtosis of any univariate Gaussian distribution is 3. The kurtosis of any sub-Gaussian distribution is below 3 and the kurtosis of any sup-Gaussian distribution is above 3. We prefer the independent components associated with the top k minimum kurtosis values.

3.2.2 Givens Rotation

The objective function in Equation 3.2 is difficult to solve. Inspired by Learned-Miller et. al [30] and Kirshner et. al [88] in which they used Givens rotation to estimate a demixing matrix for independent component analysis by sequentially rotating every two mixture components. The reason behind Givens rotation is: 2d-pairwise distances are not changing after rotation, thus joint distribution remains the same, whereas marginal distributions change after rotation. Thus, any metric based on joint distribution and marginal distributions varies when the rotation angle θ varies. We can find the maximal or minimal values of metrics with respect to θ. For d-dimensional data, a Givens rotation of angle θ for

dimensions i and j is defined as:

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cos\theta & \cdots & -\sin\theta & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

where $\sin\theta$ is on the j-th row and i-th column and $-\sin\theta$ is on the i-th row and j-th column of G.

The demixing matrix $\mathbf{W}$ can be estimated as

$$\mathbf{W} = \prod G(i,j,\theta^*), 1 \leq i,j \leq p, i \neq j \tag{3.4}$$

where $G(i,j,\theta^*)$ is a Givens rotation of the best angle θ^* for optimizing the mutual information of the dimensions i and j of a data matrix.

3.3 Algorithm

3.3.1 Greedy Search

Learned-Miller et. al [30] and Kirshner et. al [88] optimized Equation 3.2 by exhaustively search over $K = 150$ values of θ in the range $\left[0, \frac{\pi}{2}\right]$ which is time-consuming. Besides, they did not discriminate the order of optimization for different pairwise dimensions, which results in getting easily trapped in local optima. Considering the two limitations, we propose a new optimization method which is very efficient and effective.

To speed up the search process, we do not exhaustively search over $K = 150$ values of θ in the range $\left[0, \frac{\pi}{2}\right]$. In contrast, we use the history search as anchors to guide the

search process. From this perspective, the greedy search is a self-learning method based on its learned knowledge. The strategy of adjusting the search resolution (see the following) makes our greedy search self-adaptive. Note that we only need to consider θ in the interval $\left[0, \frac{\pi}{2}\right]$ because the effectiveness of any 90 degree rotation is equivalent as explained in [30]. In this paper, we adopt kernel generalized variance (KGV) proposed by Bach and Jordan [34] to estimate pairwise mutual information considering its linear complexity and especially its smoothness w.r.t. log function. For a detailed description, please see [34,109]. Now we give an example to demonstrate our greedy search method. Because in practical use we can choose different contrast functions, here we just give a generalized function f to demonstrate the main idea. The curve in Figure 3.3 (a) is a function of θ. The goal is to find the best θ^* achieving the maximal objective function f in the interval $\left[0, \frac{\pi}{2}\right]$ (the minimal f can be achieved by finding the maximal $-f$). Our method is described as follows:

Case 1: As depicted in Figure 3.3 (a), we set the ascending and descending step size to $\frac{\pi}{2K}$. We choose three different initial θ (in our example is θ_1, θ_2 and θ_3) with the same interval (e.g. $\frac{\pi}{2K}$) and compute their objective function ($f(\theta_1)$, $f(\theta_2)$ and $f(\theta_3)$). If $f(\theta_3) \leq f(\theta_2) \leq f(\theta_1)$, we assume that the function is continuously decreasing and we multiply the descending step size by $\tau = 2$ (the search resolution) and update θ_3 for the following iteration. If τ is two large, the method may skip some important search niche, while if τ is too small, the efficiency will be decreased. We update $\theta_1 = \theta_2$, $\theta_2 = \theta_3$, $f(\theta_1) = f(\theta_2)$ and $f(\theta_2) = f(\theta_3)$ as history reference values for the following search.

Case 2: We compute $f(\theta_3)$ as depicted in Figure 3.3 (b). If $f(\theta_2) \leq f(\theta_3)$ and $f(\theta_2) \leq f(\theta_1)$, we assume the function is continuously increasing. We set the descending step size to its initial value and multiply the ascending step size by τ and update θ_3. Also, we update $\theta_1 = \theta_2$, $\theta_2 = \theta_3$, $f(\theta_1) = f(\theta_2)$ and $f(\theta_2) = f(\theta_3)$ as history reference values for the following search. If $f(\theta_3) \leq f(\theta_2) \leq f(\theta_1)$, go to case 1.

Case 3: We compute $f(\theta_3)$ as depicted in Figure 3.3 (c). If $f(\theta_1) \leq f(\theta_2) \leq f(\theta_3)$, we assume the function is continuously increasing. We set the descending step size to its initial value and multiply the ascending step size by τ and update θ_3. Also, we update

$\theta_1 = \theta_2$, $\theta_2 = \theta_3$, $f(\theta_1) = f(\theta_2)$ and $f(\theta_2) = f(\theta_3)$ as history reference values for the following search. if not, go to case 4.

Case 4: In this case (Figure 3.3 (d)), $f(\theta_1) \leq f(\theta_2)$ and $f(\theta_3) \leq f(\theta_2)$, we assume there may be some peaks in the interval. We exhaustively search in the interval $[\theta_1, \theta_3]$ with a step size $\frac{\pi}{2K}$. Finally, we update $\theta_1 = \theta_2$, $\theta_2 = \theta_3$, $f(\theta_1) = f(\theta_2)$, $f(\theta_2) = f(\theta_3)$ and set the ascending step size to its initial value. Since now $f(\theta_2) \leq f(\theta_1)$, we assume the function is continuously decreasing. We multiply the descending step size by τ and update θ_3. If $f(\theta_3) \leq f(\theta_2)$, go to case 1; if $f(\theta_3) \geq f(\theta_2)$, go to case 2.

We repeat the above four cases until $\theta_3 \geq \frac{\pi}{2}$. Note that, in each case, we update the best objective function value f_b and θ^*.

3.3.2 FUSE

As said before, in [30] and [88], the authors do not differ between the order of optimizing pairwise dimensions which results in the algorithm's getting easily trapped in local optima. In this paper, we use a greedy selection method, i.e., computing the objective function values for each pairwise dimensions and then optimizing pairs according to their objective function values from the worst to the best, to make our method not easily get trapped in local optima. Our pseudo-code is given in Algorithm 1.

Steps 1 – 2 initialize the demixing matrix $\mathbf{W}$ to an identity matrix, compute the affinity matrix $\mathbf{A}$ and normalize it to $\mathbf{P}$. Steps 3 – 8 generate $p = k + 1$ pseudo-eigenvectors using power iteration (PI). In step 4, the starting vector is randomly generated following a Gaussian distribution with mean 0 and variance 1. In Steps 5 – 8, each starting vector is iteratively updated by power iteration until the acceleration threshold $\hat{\epsilon}$ or the maximum iteration number is reached. Step 10 whitens the pseudo-eigenvector matrix $\mathbf{V}$ to let it have zero expectation value and an identity covariance matrix. In step 11, $\boldsymbol{c}$ includes the indices of each pairwise components in $\mathbf{E}$ and their mutual information values ($a_i \in \{1, 2, \cdots, p\}$). To let the algorithm escape from local optima, step 14 sorts $\boldsymbol{c}$ in descending order w.r.t. mutual information values. Steps 15 – 19 use the greedy search to find the best θ^* for each

pairwise components of **E** to make them statistically independent and update components' pairwise mutual information value stored in c_{j3}, and also update **E** and **W**. We set the mutual information threshold to 0.1 for a balance between the efficiency and effectiveness. If we set it lower, the efficiency will be decreased but effectiveness will be increased and vice versa. Step 20 returns the selected independent components which will be fed to k-means to find clusters.

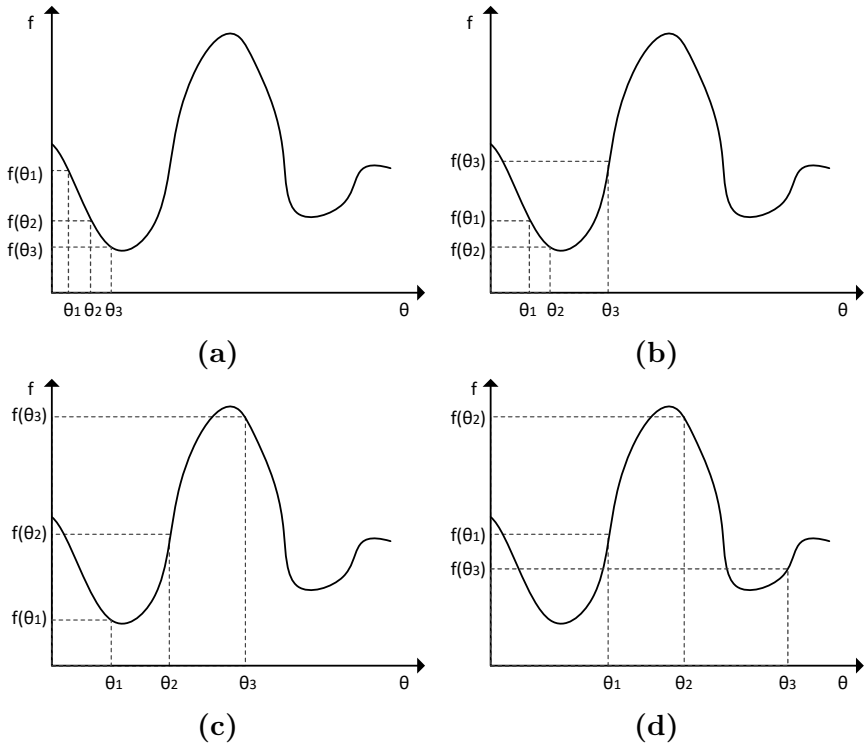


Figure 3.3: Greedy search strategy

3.3.3 Complexity Analysis

We omit the runtime for computing the affinity matrix which is a common step in all spectral clustering methods. The analysis of runtime complexity of FUSE is as follows: In steps 3 – 8, generating one pseudo-eigenvector costs $\mathcal{O}(e)$ time [39], where e is the number of edges, and thus the runtime complexity to generate $p = k + 1$ pseudo-eigenvectors by power iteration is $\mathcal{O}((k+1)e)$. In step 10, as a preprocessing step, whiten-

ing data costs $\mathcal{O}((k+1)^2 n)$ time. In this paper, we adopt Kernel Generalized Variance (KGV) using incomplete Cholesky decomposition proposed in [34] to estimate mutual information. The complexity of KGV is $\mathcal{O}(m^2 M^2 n)$ [34], where m is data dimension and M is the maximal rank considered by the low-rank decomposition algorithms for the kernels. In step 11, the computation time for all pairwise mutual information values is $\frac{k(k+1)}{2} \cdot \mathcal{O}(2^2 M_1^2 n) = \mathcal{O}(k^2 M_1^2 n)$. To make FUSE escape from local optimal, we sort $\boldsymbol{c}$ using quick sort algorithm. The runtime complexity of the ordering process is $3(k+1) \cdot \mathcal{O}(l \log l) = 3(k+1) \cdot \mathcal{O}(\frac{k(k+1)}{2} \log \frac{k(k+1)}{2}) = \mathcal{O}(k^4 \log k)$. The time cost of finding independent pseudo-eigenvectors is $3(k+1) \cdot \frac{k(k+1)}{2} \cdot K \cdot \mathcal{O}(2^2 M_2^2 n) = \mathcal{O}(k^3 M_2^2 n)$. Finally, we use k-means to cluster on the selected independent pseudo-eigenvectors. The total runtime complexity of FUSE is $\mathcal{O}\left(ke + k^2 M_1^2 n + k^3 M_2^2 n + k^4 \log k\right)$ plus the time complexity of k-means, i.e., $\mathcal{O}(nk) \times$(# k-means iterations) [20].

3.4 Experimental Evaluation

Competitors: To evaluate the performance of FUSE, we adopt three spectral clustering methods NCut [56], NJW [9] and ZP [66] and power-iteration-based clustering methods PIC [37], PIC-k [36], DPIC [71] and DPIE [45] as competitors. FUSE and all the comparison methods are written in Matlab. All experiments are run on the same machine with an Intel Core Quad i7-3770 with 3.4 GHz and 32 GB RAM. The code of FUSE and all synthetic and real-world data used in this paper are available at the website[1].

Parameters: The parameters for spectral clustering, power-iteration-based clustering methods are set according to their original papers. For FUSE, we set $\hat{\epsilon}_i = i \cdot \lceil \log(k) \rceil \cdot \frac{1e-5}{n}$ as adopted by DPIE [45]. For text data, we use cosine similarity ($\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}$) to compute the affinity matrix. For network data, the element a_{ij} of the affinity matrix $\mathbf{A}$ is simply 1 if blog i has a link to j or vice versa, otherwise $a_{ij} = 0$. For all other data, the locally scaled affinity matrix is computed as the way proposed in [66] with $K_{NN} = 7$. The original ZP method automatically chooses the number of clusters. For a fair comparison, we give

[1] https://github.com/yeweiysh/FUSE

Algorithm 1: FUSE

Input: Data $\mathbf{X} \in \mathbb{R}^{m \times n}$

Output: cluster indicator vectors

$T \leftarrow 1000$, *levels* $\leftarrow 3$, *sweeps* $\leftarrow p$; /* $p = k + 1$ */

Initialize the demixing matrix $\mathbf{W} \in \mathbb{R}^{p \times p}$ to the identity matrix and compute the random walk transition matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$;

for $j \leftarrow 1$ **to** p **do**

  $t \leftarrow 0, \mathbf{v}_j^0 \leftarrow$ **randn** $(1, n)$; /* $\mathbf{v}_j \in \mathbb{R}^{1 \times n}$ */

  /* power iteration */

  repeat

    $\mathbf{v}_j^{t+1} \leftarrow \frac{\mathbf{P}\mathbf{v}_j^t}{\|\mathbf{P}\mathbf{v}_j^t\|_1}$;

    $\boldsymbol{\delta}^{t+1} \leftarrow |\mathbf{v}_j^{t+1} - \mathbf{v}_j^t|$;

    $t \leftarrow t + 1$;

  until $\|\boldsymbol{\delta}_j^{t+1} - \boldsymbol{\delta}_j^t\|_{\max} \leq \hat{\epsilon}$ **or** $t \geq T$;

$\mathbf{V} = [\mathbf{v}_1; \ldots; \mathbf{v}_p]$;

$\mathbf{V} \leftarrow$ **whiten** $(\mathbf{V})$,$\mathbf{E} \leftarrow \mathbf{W}\mathbf{V}$;

/* $\boldsymbol{c}$ has $l = \binom{p}{2}$ tuples */

$\boldsymbol{c} \leftarrow ((a_1, a_2, I_1 = I(\mathbf{v}_{a_1 \cdot}; \mathbf{v}_{a_2 \cdot})), \ldots, (a_{l-1}, a_l, I_l = I(\mathbf{v}_{a_{l-1} \cdot}; \mathbf{v}_{a_l \cdot})))$;

for *level* $\leftarrow 1$ **to** *levels* **do**

  for *sweep* $\leftarrow 1$ **to** *sweeps* **do**

    $\boldsymbol{c} \leftarrow$ **order_descending_by_I_value** $(\boldsymbol{c})$;

    /* minimize pairwise mutual information */

    for $j \leftarrow 1$ **to** l **do**

      if $c_{j3} > 0.1$ **then**

        $[\theta^*, c_{j3}] \leftarrow$ **greedy_search** $(c_{j1}, c_{j2}, \mathbf{E})$;

        $\mathbf{E} \leftarrow G(c_{j1}, c_{j2}, \theta^*)\mathbf{E}$;

        $\mathbf{W} \leftarrow G(c_{j1}, c_{j2}, \theta^*)\mathbf{W}$;

compute kurtosis of each pseudo-eigenvector in $\mathbf{E}$ and return the pseudo-eigenvectors associated with the top k minimum values;

ZP the correct number of clusters.

Since all the comparison algorithms use k-means in the last step, in each experiment k-means is run 100 times with random starting points and the most frequent cluster assignment is used [37]. We run each experiment 50 times and report the mean and standard deviation of Adjusted Mutual Information (AMI) [73]. For AMI, higher value means better clustering.

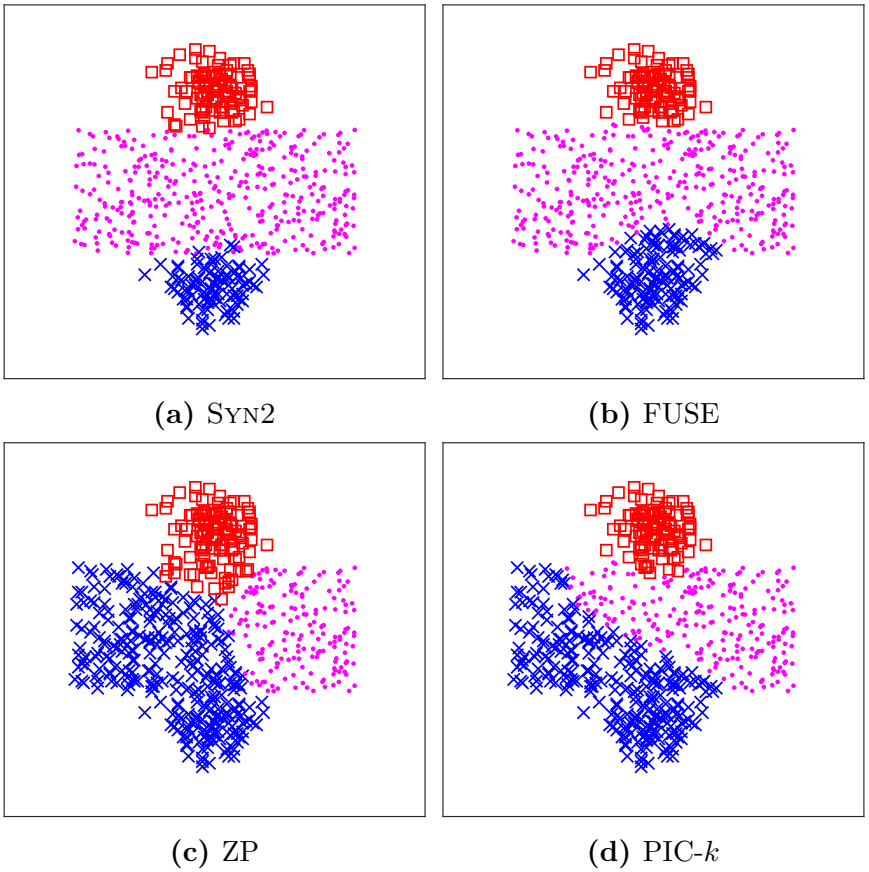
3.4.1 Synthetic Data

Quality

Synthetic dataset SYN1 is the running example used in Section 3.1. SYN1 has three clusters. Each of the two Gaussian clusters has 100 data points and the stripe cluster has 400 data points. We have shown the results before.

Synthetic dataset SYN2 has three clusters as well depicted in Figure 3.4(a). Both Gaussian clusters have 100 data points and the rectangular cluster has 400 data points. Both Gaussian clusters have some very close data points to the rectangular cluster making them hard to be separated correctly. The mean AMI of FUSE is 0.750 and the highest of the comparison algorithms' is 0.574 achieved by PIC-k. ZP only has a value 0.483. Figure 3.4(b)– (d) give us an intuitive demonstration. FUSE just wrongly clustered a few data points in the magenta rectangular cluster to the blue Gaussian cluster, while ZP and PIC-k wrongly clustered about a half of the data points in the magenta rectangular cluster to the blue Gaussian cluster. Our algorithm is superior to the competing algorithms.

SYN3 in Figure 3.5(a) also has three clusters. Two Gaussian clusters each has 90 and 92 data points, respectively. The blue ring cluster has 130 data points. SYN3 is very interesting because the density of the blue ring cluster is lower than those of the Gaussian clusters. And the ring cluster is very close to the Gaussian clusters, which could make the $K_{NN} = 7$ neighbors of some points in the blue ring cluster be belonging to the Gaussian clusters. SYN3 is also difficult to cluster correctly. However, FUSE achieved the best compared to all the comparison algorithms. PIC-k even clustered several data points



(a) SYN2 **(b)** FUSE

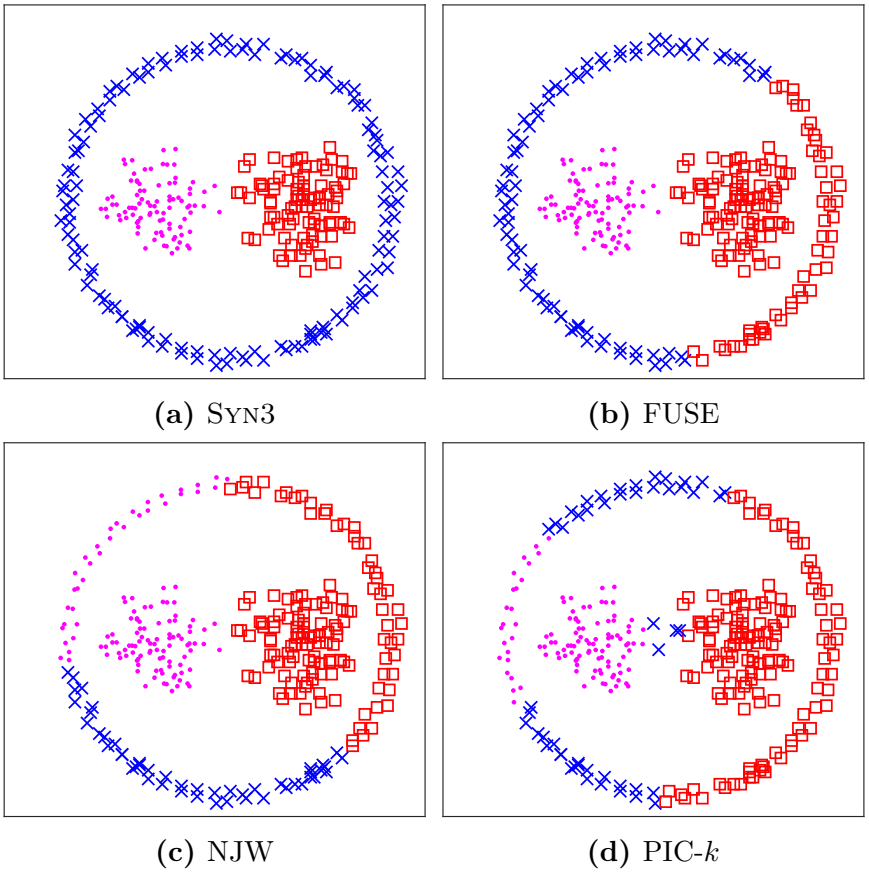
(c) ZP **(d)** PIC-k

Figure 3.4: Clustering results on SYNC2 (shown are the most frequent clusterings)

belonging to two Gaussian clusters to the ring cluster, which did not make sense.

SYN4 in Figure 3.6(a) contains five clusters, each of the two Gaussian clusters has 100 data points, each of the two square clusters has 82 and 100 data points respectively and the ring cluster has 56 data points. The ring cluster is very close to the square clusters and even has some overlap with the two Gaussian clusters. Still, FUSE achieved the best result, only not distinguishing between the overlapped data points from the Gaussian clusters and the ring cluster. ZP wrongly detected a half of the data points in the ring cluster to the green Gaussian cluster. PIC-k wrongly clustered several data points in the ring cluster to the black square cluster although the densities of these two clusters are significantly different.

For all these multi-scale synthetic datasets, our algorithm FUSE outperforms all the



(a) SYN3 **(b)** FUSE

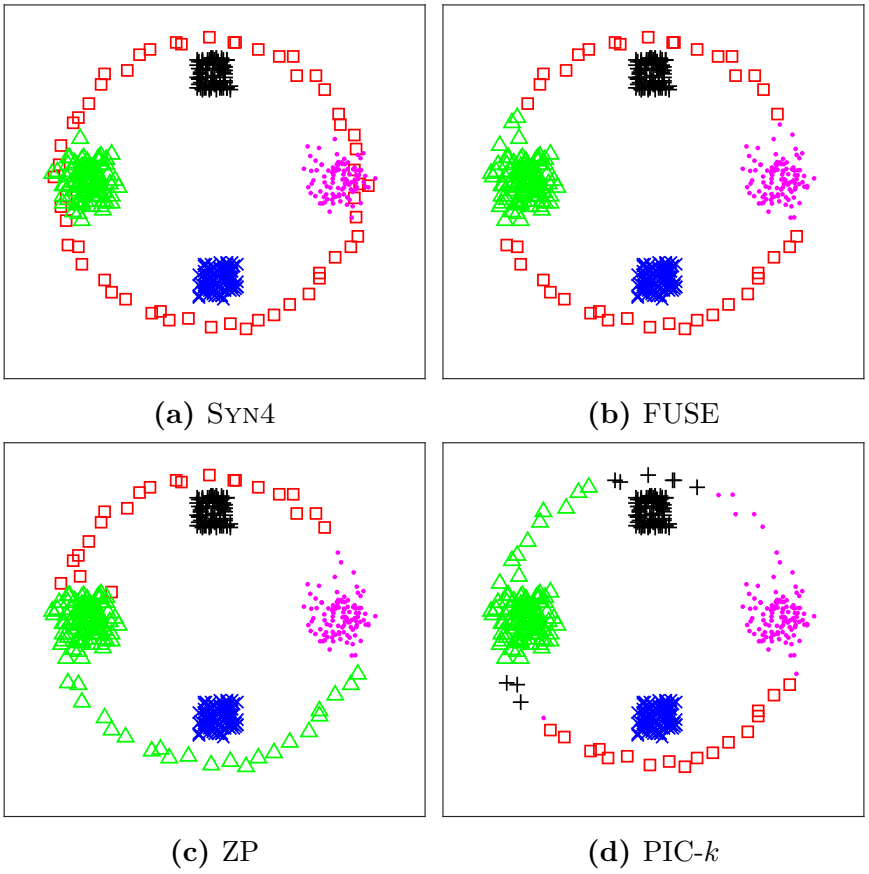
(c) NJW **(d)** PIC-k

Figure 3.5: Clustering results on SYNC3 (shown are the most frequent clusterings)

competing algorithms. FUSE is even superior to the spectral clustering algorithms ZP, NCut and NJW, which proves that the *normalized cut* criterion is not alway suitable for clustering. FUSE-E is our algorithm using the exhaustive search strategy over θ proposed in [30,88] which does not discriminate the order of optimization of the pseudo-eigenvectors generated by PI. We can see that sometimes it gets trapped in local optimal (the result on SYN3). Our algorithm FUSE adopting the greedy search strategy achieves quite similar or even better results than using the exhaustive search strategy.

Scalability

In this experiment, we want to test the runtime against the number of data points using data SYNC5. Synthetic dataset SYNC5 is generated as follows: Firstly, we generate two



(a) SYN4 **(b)** FUSE

(c) ZP **(d)** PIC-k

Figure 3.6: Clustering results on SYNC4 (shown are the most frequent clusterings)

2D clusters sampled from uniform distributions with the number of data points 4,000 and 1,000 respectively. The two clusters are our basis clusters. Then at each step we increase the data points in each cluster by the size of its basis. Finally, we have data points varying from 5,000 to 30,000 by a step size 5,000. We feed each algorithm with the same affinity matrix. Thus, the runtime does not include the computation time for the affinity matrix. The results are demonstrated in Figure 3.7. Since the runtime of NJW and NCut are similar, here we only show the runtime of NCut for a clearer demonstration. Figure 3.7 shows that the runtime of FUSE becomes much lower than that of ZP, NCut and DPIC when increasing the number of data points. Compared to PIC, we can see their slope variances are quite similar. The runtime difference between FUSE and PIC is owing to that FUSE needs to determine the directions in which the entropy of pseudo-eigenvectors

Table 3.1: AMI on Synthetic Data (mean ± standard deviation)

AMI	SYNC1	SYNC2	SYNC3	SYNC4
FUSE	0.715±0.131	**0.750±0.127**	**0.702±0.048**	**0.891±0.016**
FUSE-E	**0.735±0.142**	**0.750±0.120**	0.688±0.044	0.886±0.018
ZP	0.374±0	0.483±0	0.528±0	0.882±0
NCUT	0.370±0	0.479±0	0.522±0	0.874±0.002
NJW	0.379±0	0.451±0	0.533±0	0.879±0.002
PIC	0.355±0.088	0.309±0	0.494±0.059	0.840±0.034
PIC-k	0.324±0.048	0.574±0.105	0.508±0.055	0.8544±0.022
DPIC	0.324±0.094	0.465±0.113	0.499±0.107	0.482±0.064
DPIE	0.350±0.056	0.128±0.097	0.310±0	0.630±0

is minimized. Compared with PIC-k, the runtime of FUSE becomes close to that of PIC-k when the number of data points increases to 30,000. Note that FUSE-E is our algorithm using the exhaustive search strategy. FUSE is faster than FUSE-E as can be seen from the figure. Our algorithm is of more practical use than ZP, NCut, NJW and DPIC.

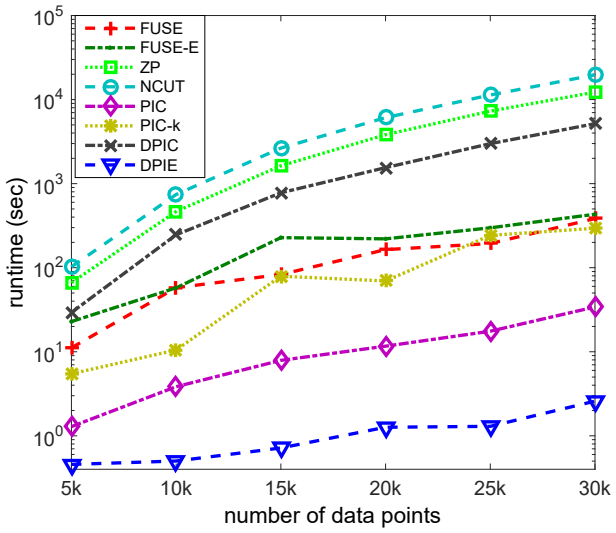


Figure 3.7: Runtime comparison

3.4.2 Real-world Data

Clustering

Now we demonstrate the effectiveness of our FUSE on seven real-world datasets. PENDIGITS is available from UCI machine learning repository[2]. The original datasets MNIST, 20NEWSGROUPS, REUTERS21578, TDT2 and RCV1 are available at this website[3]. 20NGD from [37] is a subset of 20NEWSGROUPS, and MNIST0127 from [71] is a subset of MNIST. TDT2_3CLASSES, REUTERS_4CLASSES and RCV1_4CLASSES are samples from original REUTERS21578, TDT2 and RCV1 corpus using random indices from the website3[0]. AGBLOG is a connected network dataset of 1222 liberal and conservative political blogs mined from blog homepages [37]. For text datasets, we use the preprocessed document-term matrix to compute the TF-IDF matrix. Then each feature vector is normalized to have unity norm. Finally, we use cosine similarity to compute the affinity matrix. The statistics of all datasets are given in Table 3.2.

Table 3.2: Statistics of Datasets

Dataset	#instances	#features	#clusters
PENDIGITS	7494	16	10
MNIST0127	4189	784	4
AGBLOG	1222	498	2
20NGD	800	26214	4
TDT2_3CLASSES	314	36761	3
REUTERS_4CLASSES	649	18933	4
RCV1_4CLASSES	1000	29985	4

From Table 3.3, we can see that on all datasets, FUSE achieves the best results, even outperforms self-tuning spectral clustering algorithm (ZP) and the conventional spectral clustering algorithms (NCut and NJW). Compared to PIC and PIC-k, FUSE improves AMI on each dataset. The most likely reason is the pseudo-eigenvectors found by FUSE are statistically independent (non-redundant), which make every cluster stand out in each

[2]http://archive.ics.uci.edu/ml/
[3]http://www.cad.zju.edu.cn/home/dengcai/Data/data.html

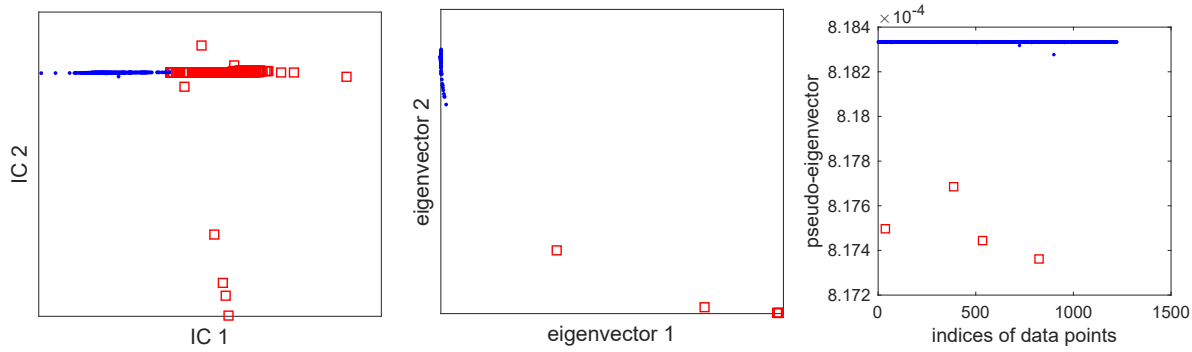
pseudo-eigenvector. Compared to DPIC and DPIE which also aim at reducing redundancy in pseudo-eigenvectors generated by PI, FUSE improves AMI much on most datasets. One reason is that finding directions in which the entropy is minimized is much more beneficial to clustering.

Table 3.3: AMI on Real-world Data (mean ± standard deviation)

AMI	PENDIGITS	MNIST0127	AGBLOG	20NGD
FUSE	**0.828±0.009**	**0.594±0.043**	**0.729±0.001**	**0.348±0.017**
ZP	0.813±0	0.444±0	0.017±0	0.293±0
NCUT	0.800±0	0.418±0	0.002±0	0.325±0.022
NJW	0.800±0	0.496±0.040	0.017±0	0.265±0
PIC	0.680±0	0.446±0.034	0.226±0.317	0.318±0.004
PIC-k	0.773±0.024	0.456±0.006	0.227±0.288	0.284±0.051
DPIC	0.616±0.044	0.331±0.006	0.330±0	0.046±0.028
DPIE	0.624±0.034	0.011±0.003	0.050±0	0.271±0.046
AMI	TDT2_3CLASSES	REUTERS_4CLASSES	RCV1_4CLASSES	
FUSE	**0.951±0.005**	**0.593±0.031**	**0.493±0.015**	
ZP	0.673±0	0.585±0.02	0.452±0	
NCUT	0.670±0	0.539±0.004	0.405±0	
NJW	0.670±0	0.567±0.005	0.402±0	
PIC	0.308±0.287	0.310±0	0.335±0.034	
PIC-k	0.400±0.300	0.366±0.069	0.351±0.021	
DPIC	0.606±0.019	0.234±0.028	0.150±0.044	
DPIE	0.135±0.197	0.434±0.198	0.404±0.043	

Two most interesting results are on AGBLOG and TDT2_3CLASSES datasets. All comparison methods except PIC and PIC-k fail on AGBLOG dataset. AGBLOG dataset has two balanced clusters with the number of instances 586 and 636, respectively. We show the most frequent data embeddings of FUSE and ZP (the results of NJW and NCut are very similar) in Figure 3.8(a), (b) and (c). We can see that most data points (blue ones in Figure 3.8(b) and (c)) are assigned to one cluster, which makes the results of ZP and PIC-k not appealing. However, our algorithm finds the embedding space in which two clusters are separated evenly.

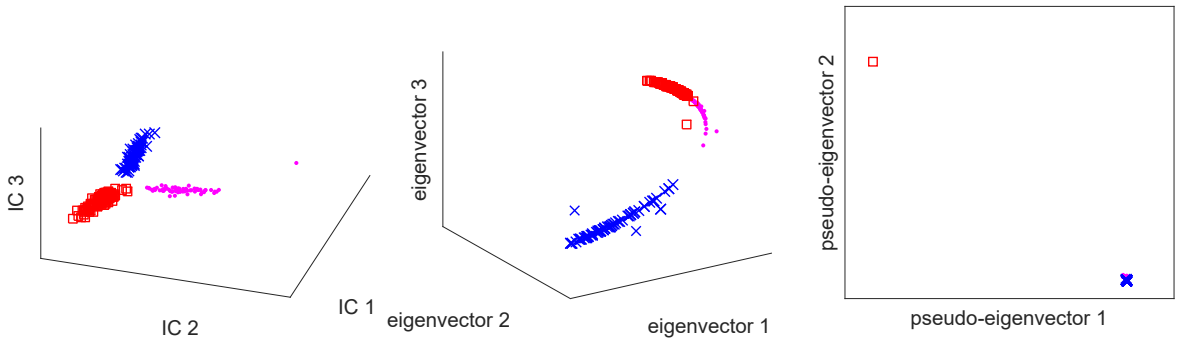
Figure 3.9 shows the clustering results on the TDT2_3CLASSES dataset. Figure 3.9(b) demonstrates the eigenvector space found by ZP, in which the red square cluster and the



(a) FUSE **(b)** ZP **(c)** PIC-k

Figure 3.8: The embedding space found by FUSE, ZP and PIC-k on AGBLOG data.

dot magenta cluster are connected together. ZP only achieves 0.673 in terms of AMI. PIC-k found two pseudo-eigenvectors. Also in its found embedding space, two clusters (blue and magenta) are not well separated. However, in the embedding space detected by our algorithm, three clusters are well separated, which makes the value of AMI much higher than those of the competing methods as can be seen in Table 3.3. Thus, the embedding space found by our algorithm is much more attractive and effective.



(a) FUSE **(b)** ZP **(c)** PIC-k

Figure 3.9: The embedding space found by FUSE, ZP and PIC-k on TDT2_3CLASSES data.

If we look into Table 3.4, we can find that we have six datasets on which the runtime of our method is much lower than that of NCUT and NJW. And we also have four datasets on which our method is faster than ZP. Our method is very efficient and promising for practical use. However, on PENDIGITS dataset, FUSE is slower than the conventional

spectral clustering algorithms because the maximal rank M considered by KGV is close to n which costs much time to compute the pairwise mutual information.

Table 3.4: Runtime (sec) on Real-world Data (mean ± standard deviation)

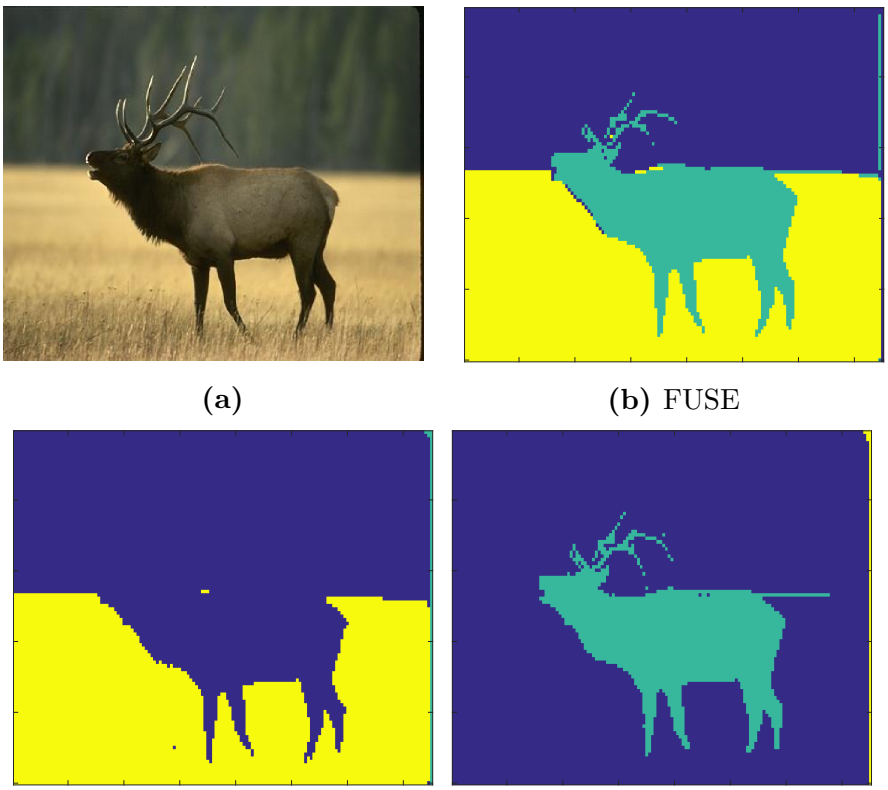
Runtime	PENDIGITS	MNIST0127	AGBLOG	20NGD
FUSE	90.644±13.336	3.831±0.890	0.325±0.074	0.474±0.132
ZP	50.250±3.105	41.188±0	0.781±0	0.271±0
NCUT	50.375±2.066	62.697±0	2.318±0	0.957±0
NJW	51±3.406	59.940±0	0.374±0	0.807±0
PIC	5.197±0.017	0.249±0.031	0.063±0.010	0.013±0.003
PIC-k	13.745±5.020	0.263±0.036	0.035±0.006	0.002±0.002
DPIC	299.556±30.740	20.270±1.933	0.521±0.047	0.468±0.078
DPIE	0.099±0.520	1.816±3.822	0.003±0.002	0.097±0.009
Runtime	TDT2_3CLASSES	REUTERS_4CLASSES	RCV1_4CLASSES	
FUSE	0.826±0.376	0.394±0.201	0.391±0.148	
ZP	0.032±0	0.438±0	52.714±4.310	
NCUT	51.429±3.824	51±3.512	52.143±3.891	
NJW	50.286±3.302	51±3.873	52.143±4.100	
PIC	0.035±0.042	0.010±0.006	0.003±0	
PIC-k	0.005±0.015	0.002±0	0.003±0	
DPIC	0.099±0.078	0.231±0.042	0.502±0.074	
DPIE	0.183±0.019	0.104±0.015	3.287±0.041	

Image Segmentation

In this section, we apply our algorithm on image segmentation. Figure 3.10 and Figure 3.11 show two examples from the Berkeley Segmentation Dataset and Benchmark [4]. Each pixel is represented as a five dimensional vector of its pixel coordinates x and y, and the color intensities [20]. We set the number of clusters in Figure 3.10 three and four for Figure 3.11. Since the results returned by ZP, NCut and NJW are very similar, we only show the results of ZP here. For other methods, we show such methods whose results are more interpretable. Figure 3.10(b) shows that FUSE separates the deer, the grass and the forest very well. ZP correctly segments the grass, but does not distinguish the deer from the forest, while PIC-k recognizes the deer but not the grass or the forest. Compared

[4] https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images.html

to Figure 3.10(a), Figure 3.11(a) demonstrates a more challenging task because the two elephants are very similar in terms of the color and position. However, our method FUSE successfully distinguishes these two similar elephants and also recognizes the sky. ZP cannot separate the two elephants, but the sky is well segmented. DPIE can also distinguish the two elephants but the segmentation is worse than FUSE's. In addition, DPIE does not segment the sky well.



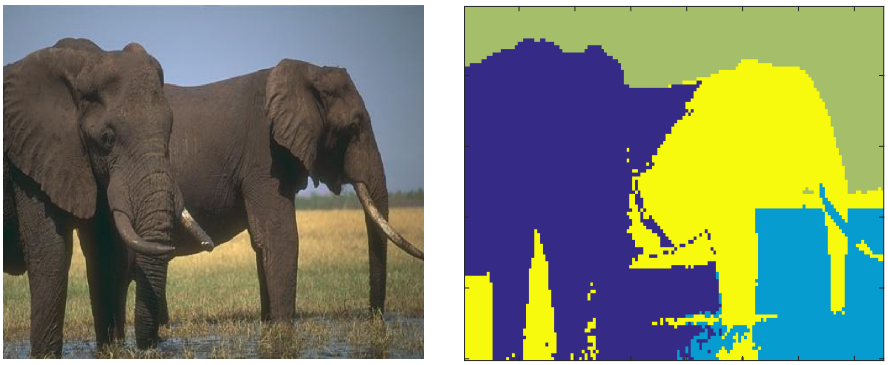
(a) **(b)** FUSE

(c) ZP **(d)** PIC-k

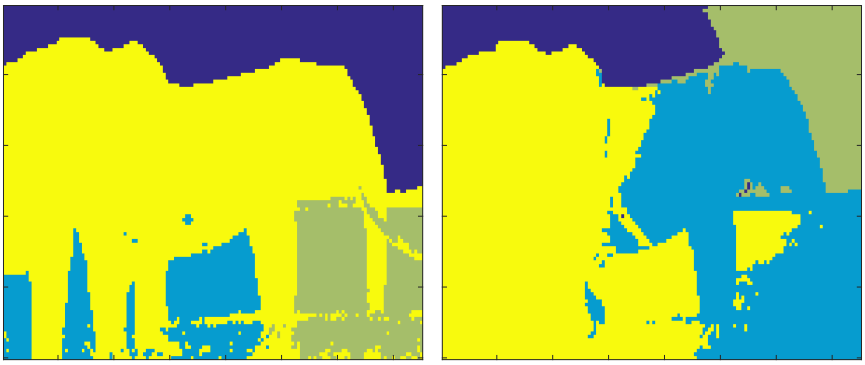
Figure 3.10: Image segmentation (shown are the most frequent clusterings of each method)

3.5 Related Work

Spectral clustering. Spectral clustering is very popular in data mining owing to its ability to detect arbitrary-shape clusters in data spectrum space. Spectral clustering can be



(a) **(b)** FUSE



(c) ZP **(d)** DPIE

Figure 3.11: Image segmentation (shown are the most frequent clusterings of each method)

divided into three categories by the type of Laplacian matrix, i.e., unnormalized spectral clustering, normalized spectral clustering proposed by Shi and Malik (NCut) [56] and another normalized spectral clustering proposed by Ng, Jordan and Weiss (NJW) [9]. After deciding the type of Laplacian matrix, it computes the first k eigenvectors of the Laplacian matrix and then uses k-means to cluster in the space formed by these eigenvectors. Spectral clustering is very elegant. However, the computation cost is very high for large-scale data. Finding eigenvectors takes $\mathcal{O}(n^3)$ in general. Recently, researchers have proposed many fast approximating techniques, such as IRAM and sampling techniques [22,26]. Spectral clustering assumes that the "informative" eigenvectors are those associated with the smallest k eigenvalues, which seems not to be successful on some real-world data with much noise or multi-scale density. And this promotes many researchers work on how to

select much informative eigenvectors, and how to estimate the local scale of data with varying densities, shapes and levels of noise [20, 66, 94, 108]. ZP [66] is a representative of all these algorithms. ZP takes local scaling into consideration and constructs a locally scaled affinity matrix which proves beneficial to clustering especially for multi-scale data or data with irregular background clutter. ZP also exploits the structure of eigenvectors to improve clustering. As NCut, NJW and other spectral-based clustering methods, ZP only uses the first k eigenvectors to cluster, which is not appropriate in some cases. However, our algorithm FUSE exploits all "informative" eigenvectors and fuses all their information to accomplish better clustering.

Power-iteration-based clustering. Power Iteration Clustering (PIC) [37] uses truncated power iteration on a normalized affinity matrix of the data points to find a very low-dimensional data embedding which is a linear combination of the major eigenvectors for clustering. It is very elegant and efficient. However, the assumptions it bases on are very strict and it returns only one pseudo-eigenvector which prevents its performance on data with large number of clusters, where cluster-collision problem is easy to happen. PIC-k [36] has been proposed to alleviate the situation but actually it still cannot solve the cluster-collision problem due to much similarity exists in the returned pseudo-eigenvectors. Another clustering algorithm based on power iteration is Deflation Power Iteration Clustering (DPIC) [71] which uses Schur complement deflation to generate multiple orthogonal pseudo-eigenvectors. However, the pseudo-eigenvectors still contain noise together with cluster-separation information. Diverse Power Iteration Clustering (DPIE) [45] normalizes the residue (regression) error which is obtained by subtracting the effects of the already-found DPIEs from the embeddings returned by PIC. However, DPIE cannot guarantee to find diverse embeddings in every iteration and it bases on the assumption that clear eigen-gap exists between every two successive eigenvalues which is also very strict. Our method FUSE does not make any assumptions and finds statistically independent pseudo-eigenvectors, each of which is a different linear combination of the original eigenvectors. Besides, each statistically independent pseudo-eigenvector eliminates noise and only keeps cluster-separation information which makes FUSE much more advanced and effective.

3.6 Summary

We have proposed FUSE to handle multi-scale data on which the *normalized cut* criterion tends to fail even given a suitable locally scaled affinity matrix. FUSE exploits PI and ICA to fuse all "informative" eigenvectors to yield better clustering. Since the pseudo-eigenvectors fused by PI are redundant and the cluster-separation information does not stand out, ICA is adopted to reduce the redundancy. Then, a *kurtosis*-based selection strategy is used to improve cluster separation. To speed up the search process, we have developed a greedy search method which learns from its history search records and also adaptively adjusts its search resolution. Extensive experiments and evaluations on various synthetic and real-world data show FUSE's promising in dealing with multi-scale data.

Chapter 4

Finding Multiple Independent Subspace Clusters

In Chapter 3, we proposed full spectral clustering (FUSE) method to find clusters in full space. However, high-dimensional data can encapsulate different object groupings in subspaces of arbitrary dimension and orientation. Finding such subspaces and the groupings within them is the goal of generalized subspace clustering. In this chapter we present a generalized subspace clustering technique capable of finding *multiple* non-redundant clusterings in arbitrarily-oriented subspaces. We use Independent Subspace Analysis (ISA) to find the subspace collection that minimizes the statistical dependency (redundancy) between clusterings. We then cluster in the arbitrarily-oriented subspaces identified by ISA. Our algorithm ISAAC (Independent Subspace Analysis and Clustering) uses the Minimum Description Length principle to automatically choose parameters that are otherwise difficult to set. We comprehensively demonstrate the effectiveness of our approach on synthetic and real-world data.

Parts of the materials presented in this chapter have been published in [101], where Wei Ye was mostly responsible for the development of the main idea, conducted the most parts of the experimental evaluation, and wrote the major parts of the paper; Claudia Plant supervised the project and helped with the development of the idea and the paper writing;

Samuel Maurus and Nina Hubig wrote some parts of the paper and did some experiments.

"*Wei Ye, Samuel Maurus, Nina Hubig, Claudia Plant. Generalized Independent Subspace Clustering. IEEE 16th International Conference on Data Mining (ICDM), pp. 569–578, 2016.*"

4.1 Motivation

Data can hold a wealth of potential insights, some of them perhaps in the form of object clusterings. However, as the dimensionality of a dataset increases, full-space clustering methods (e.g. classical k-means or DBSCAN) encounter a suite of difficulties colloquially attributed to *the curse of dimensionality*. These difficulties are detailed with examples in [46]. One difficulty is the increasing "meaninglessness" when trying to discriminate between "similar" and "dissimilar" objects in high-dimensional spaces. Another difficulty arises when some attributes are relevant to some object groupings but not to others (or the attributes are simply always irrelevant). These difficulties become evident on datasets of even moderate dimensionality. To make matters worse, real-world datasets are almost invariably contaminated with noise and outliers.

The research field of *subspace clustering* focuses on addressing these difficulties. Generally, the goal of algorithms from this field is to find clusters in subspaces of the original feature space. Clearly there is an infinite number of such subspaces, so this is no trivial problem. To increase tractability, much existing research presents techniques which find *axis-parallel* subspaces only. In this work we make no such restriction and focus on 1) the general problem of finding arbitrarily-oriented subspaces and 2) achieving multiple independent clusterings in these subspaces.

To set the stage, we introduce in Figure 4.1 our "running example" of an illustrative (synthetic) 4D dataset[1] which is known to contain *different* clusterings in two subspaces. The projections of the raw-form data onto two of its 2D *axis-parallel* subspaces[2] are shown

[1]The data and its generative model are available in our supplement.

[2]For reference we also provide the full scatterplot matrix for the four axis-parallel dimensions in our supplement (none of the axis-parallel views shows convincing clusterings in both subspaces).

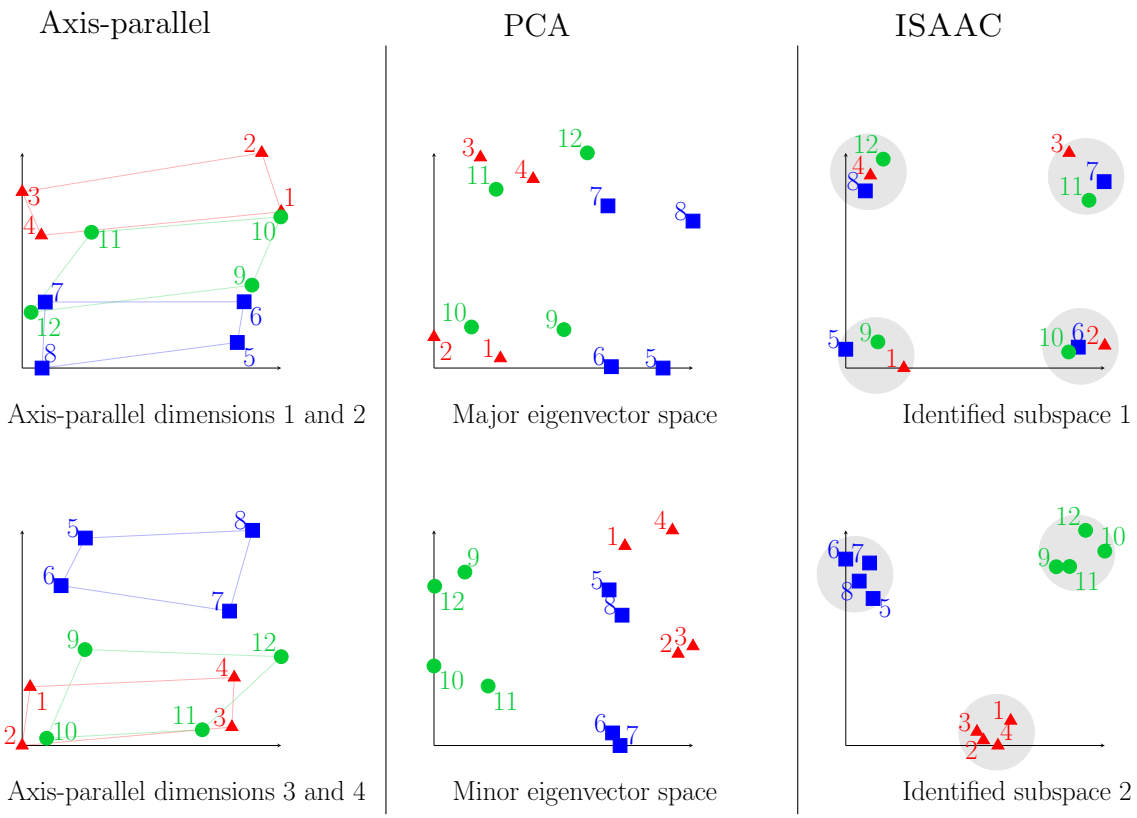


Figure 4.1: An illustrative 4D dataset with twelve objects. The axis-parallel subspaces (**left**) show little evidence of object-grouping, and we can clearly see from the object relationships across subspaces (e.g. comparing the blue rectangles) that the subspaces have a high mutual information (*redundancy*, *dependency*). PCA (**middle**) struggles to identify a more interesting grouping. ISAAC (**right**) finds two clusterings in two subspaces, one with four clusters and the other with three. The grouping of objects in each clustering is different, so both clusterings are informative and non-redundant.

on the left. Classical full-space clustering methods like k-means view the data from this perspective, where our naked eye fails to identify any obvious grouping behavior. Furthermore, if one considers the positions of the points in relation to one another (enhanced in the image by lines that connect the objects of common color), one appreciates why the mutual information between these subspaces is relatively high. That is, when considering subspaces from this first perspective, the bottom subspace offers little information gain over the top.

To mitigate such problems, it is common to pre-process data using Principal Component

Analysis (PCA) before the application of a full-space clustering technique. In this example, the major and minor eigenvector spaces of the PCA result (Figure 4.1, middle) indeed show some improvement, but struggle to uncover any highly-convincing groupings. This is unsurprising: PCA fails in such settings [46] because it is a *global* dimensionality reduction technique that finds *one* optimal representation for the *complete* set of points. In our example we have different clusterings in different subspaces, so the global approach falters.

The algorithm we propose in this paper is named ISAAC. On the right of Figure 4.1 we see that it is able to correctly identify multiple clusterings, that is, a clustering in each of two different arbitrarily-oriented subspaces. We can see that the number of clusters in each clustering may be different, and also note that the number of dimensions per subspace may be different (in this example both subspaces have two dimensions). Importantly, ISAAC focuses on *minimizing redundancy between clusterings by maximizing independence between subspaces*. Practically this means that each clustering found by ISAAC in each independent subspace is highly informative and non-redundant. The lower subspace, for example, shows tight grouping of objects with *common* color and shape. In contrast, the upper subspace shows tight grouping of objects with *heterogeneous* color and shape. The upper subspace hence encapsulates different grouping behavior, which is non-redundant information potentially leading to another valuable domain insight.

To arrive at such a solution, our method ISAAC combines Independent Subspace Analysis (ISA) [4] and clustering in one automatic framework. Through ISA we linearly transform the original space into several pairwise-independent (non-redundant) subspaces. We find the appropriate subspace cardinalities (required by ISA) using a greedy heuristic search that exploits the Minimum Description Length (MDL) principle. For practically finding the correlation clusters in each subspace, we use EM clustering with a hard-assignment of objects to clusters after each expectation-maximization step (MDL needs definite assignment of objects to clusters), although we note that our technique is agnostic to the exact algorithm used in this step. Again MDL is adopted to automatically choose the number of clusters in each independent subspace, making ISAAC parameter-free in theory and practice. The contributions can be summarized as follows:

- **We present a highly effective technique for finding multiple subspaces and the clusterings within them**. Our algorithm ISAAC uses a greedy search algorithm to parameterize ISA and acquire promising independent subspaces for clustering.
- **We minimize the redundancy between clusterings to maximize the insights**. The subspaces found by ISA are inherently non-redundant, making each clustering informative.
- **We remain robust against noise** because ISAAC can separate noise attributes from those crucial for clustering.
- **We support automation by exploiting information theory**. ISAAC is *parameter-free* because all building blocks of the framework exploit the MDL principle.

4.2 Generalized Independent Subspace Clustering

4.2.1 Statistical Independence in Clustering

Why does it make sense to search for statistically-independent subspaces in the context of clustering? We offer a brief review of the intuition. From probability theory we know that two random variables A and B are *statistically independent* when their joint probability density function (PDF) is factorizable to a product of its marginal PDFs: $f_{A,B}(A,B) = f_A(A) \cdot f_B(B)$. In this situation we understand that knowing the a_i value from a realization $(a_i, b_i)^\intercal$ of the random vector $(A,B)^\intercal$ gives us no additional information about the value of b_i, and vice versa.

We can apply this basic formulation to pairs of subspaces, each of which is simply a set of random variables.

Definition 3 *(Independent Subspaces) Two subspaces* $\mathcal{A} = \{A_1, \ldots, A_u\}$ *and* $\mathcal{B} = \{B_1, \ldots, B_v\}$ *are called mutually independent when the joint PDF of all involved random*

variables (i.e. from both subspaces) is factorizable to a product of the joint PDFs of the subspace-specific random variables:

$$\begin{aligned} & f_{A_1,\ldots,A_u,B_1,\ldots,B_v}(\alpha_1,\ldots,\alpha_u,\beta_1,\ldots,\beta_v) \\ & = f_{A_1,\ldots,A_u}(\alpha_1,\ldots,\alpha_u)\cdot f_{B_1,\ldots,B_v}(\beta_1,\ldots,\beta_v), \\ & \quad \forall \alpha_1,\ldots,\alpha_u,\beta_1,\ldots,\beta_v \in \mathbb{R} \end{aligned} \tag{4.1}$$

Consider again ISAAC's solution to our running example (Figure 4.1), which illustrates independence in the context of clustering. Knowing that an object is in the "red cluster" in the bottom subspace tells us little about the cluster in which that object is found in the top subspace. The two subspaces are hence highly independent (non-redundant). If the same points were instead clustered *similarly* in both subspaces, the two subspaces would be highly dependent (redundant).

At this stage it is also appropriate to review the notion of statistical independence *within* a subspace. Inside a subspace, we have one random variable associated with each dimension. Broadly speaking, if objects exhibit strong clustering behavior within a subspace, then the statistical independence between the random variables of that subspace is low (equivalently, the *dependency* is high). For example, looking at the lower subspace found by ISAAC in our "running example" (Figure 4.1), we gain a high amount of information about the "x-axis" position of a point if we know that it lies on the bottom of the "y-axis" (i.e. in the cluster of points $1-2-3-4$). In the case of the other extreme, if the distribution of objects within a subspace were uniformly random (no clustering behavior), then statistical dependence would be minimal.

In summary, we seek high independence *between* subspaces in order to reduce redundancy and find multiple potential insights of data, and low independence *within* a subspace for identifying interesting object groupings. ISA optimizes based on the first condition, and also permits high-dependency within identified subspaces (the second condition). However, it is of course possible for ISA to yield a set of subspaces where the dependency *within* each is not "interesting" enough (i.e. no interpretable clustering behavior). In Section 4.2.4 we

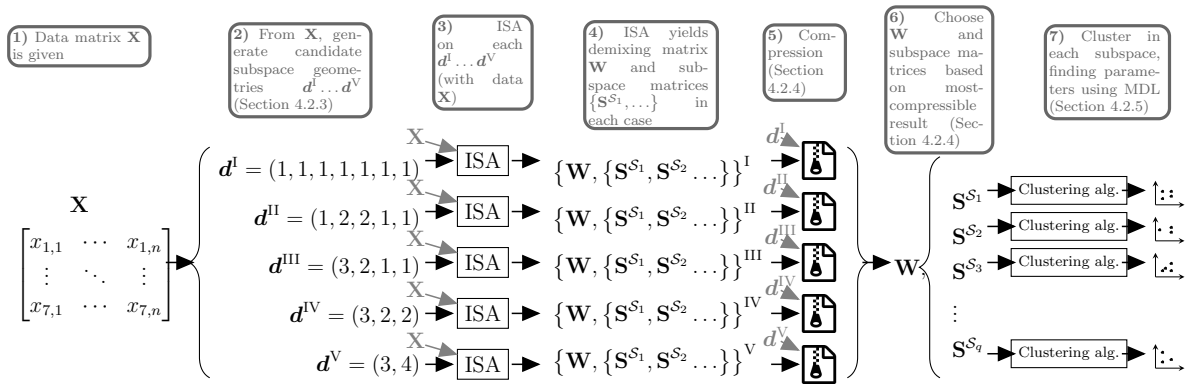


Figure 4.2: Conceptual overview of our "workflow" for clustering in independent subspaces.

describe how our algorithm ISAAC mitigates such unfavorable solutions by equating "interpretability" with "compressability", thus weeding out candidate solutions which have negligible clustering behavior within each subspace.

4.2.2 Broad Overview of our Approach

Before risking "losing sight of the forest for the trees", we first consider in Figure 4.2 an overview of our approach. The data matrix $\mathbf{X}$ is the input to our algorithm. Based only on $\mathbf{X}$, we calculate candidate parameters for input to ISA (Section 4.2.3). For each candidate parameter we then solve **Problem ISA**. The solution in each case is a demixing matrix $\mathbf{W}$ and a set of subspace matrices encapsulating the projection of the original data onto the subspaces identified by ISA.

We choose the most appropriate solution from this set (Section 4.2.4). Finally, for the chosen solution, we perform clustering in each subspace, appropriately choosing clustering parameters for each subspace (Section 4.2.5).

In order to select the most appropriate solution, we adopt the Minimum Description Length (MDL) principle [55]. Given a set of candidate models, the core idea of the MDL principle is to choose the model which allows a receiver to exactly reconstruct the original data using the most succinct transmission. Exploiting the idea that *any regularity in data can be used to compress that data*, MDL balances the coding length of the model and the

coding length of the deviations of the data from that model. More concretely, the coding cost for transmitting data D together with a "hypothesizing" model M is[3]

$$L(D, M) = L(M) + L(D|M). \tag{4.2}$$

The reader can find a more complete treatment of MDL in [55]. Note that we use an entropy-coding strategy in this work (to optimize the overall compression of the data). The following subsections provide the technical details for each stage.

4.2.3 Choosing Candidate Subspace Cardinalities

Our task in Stage 2 of Figure 4.2 is to find candidates for the subspace cardinalities vector $\boldsymbol{d}$ required by **Problem ISA**. As ISA is invariant to permutations of a given $\boldsymbol{d}$, the number of possibilities for the vector $\boldsymbol{d}$ is equal to the partition[4] $p(m)$ of m. For increasing m the value of $p(m)$ quickly becomes intractably large, so we need a heuristic for generating sensible candidates. The heuristic's guiding principle is to search for the axis-parallel subspaces of $\mathbf{X}$ that maximize the intra-subspace dependency and minimize inter-subspace dependency. As depicted with $\boldsymbol{d}^{\mathrm{I}}, \ldots, \boldsymbol{d}^{\mathrm{V}}$ for the seven-dimensional example in Figure 4.2, our heuristic involves generating these candidates in a bottom-up fashion.

Our first candidate is the vector $\boldsymbol{d}^{\mathrm{I}} = (1, 1, \ldots, 1) \in \{1\}^m$, which corresponds to a single subspace for every dimension of $\mathbf{X}$. Given $\boldsymbol{d}^{\mathrm{I}}$, we "merge" cardinalities based on the pairwise dependencies between the corresponding subspaces in $\mathbf{X}$. For example, our running-example data (Figure 4.1) was drawn from a random vector $\vec{\mathbf{X}}$ of length $m = 4$, so $\boldsymbol{d}^{\mathrm{I}} = (1, 1, 1, 1)$. Based on this first vector, we consider the data subspaces $\{X_1\}, \{X_2\}, \{X_3\}, \{X_4\}$ and compute a measure reflecting their pairwise dependencies. We then merge those subspaces that we deem "dependent", taking the strength of these dependencies into account in order to resolve merge conflicts.

To support this strategy, we require a mechanism for deciding whether two subspaces

[3] We abuse the notation here: D and M are *not* random variables.

[4] The number of ways to express m as the sum of positive integers (order irrelevant).

are "dependent enough" to warrant a merge. Here we again turn to information theory: if the cost to compress the objects in separate subspaces exceeds that for a joint subspace, we consider those subspaces as merge candidates:

Definition 4 *(Mergeable Subspaces) We define two subspaces $\mathcal{A}$ and $\mathcal{B}$ for the same n objects as mergeable when*

$$C_M(\mathcal{A}, \mathcal{B}) = C_H(\mathcal{A} \cup \mathcal{B}) - C_H(\mathcal{A}) - C_H(\mathcal{B}) < 0, \tag{4.3}$$

where

$$C_H(\mathcal{X}) = \frac{|\mathcal{X}|}{2} \cdot \log_2(n) + \sum_{i=1}^{n} \log_2 \frac{1}{\widehat{f}_{\vec{\mathbf{X}}}(\mathbf{x}_{\cdot i})}$$

is the entropy cost for encoding the n objects in subspace $\mathcal{X}$ using the KDE estimate $\widehat{f}_{\vec{\mathbf{X}}}$ for its probability-density function (the first summand represents the cost to encode the KDE model, i.e. its covariance matrix).

The C_M values for our running example are shown in Table 4.1. Based on (4.3), all of these pairs are a candidate for merging. We greedily choose $\{X_1, X_2\}$ and $\{X_3, X_4\}$ (these being the strongest). Our second candidate vector is hence $\boldsymbol{d}^{\text{II}} = (2, 2)$. This process repeats until there are no more merge candidates, or until we have arrived at one "global" subspace ($|\boldsymbol{d}| = 1$). With these termination conditions it is trivial to show that the maximal number of candidates generated is m, and that convergence is guaranteed. In the case of our running example, the second merge iteration sees $C_M(\{X_1, X_2\}, \{X_3, X_4\}) = 2.81 > 0$, so no further merges are performed.

Table 4.1: Example merge costs C_M (running example).

Candidate (i, j)	$(1,2)$	$(1,3)$	$(1,4)$	$(2,3)$	$(2,4)$	$(3,4)$
$C_M(X_i, X_j)$	-10.31	-0.36	-0.03	-1.13	-0.46	-8.21

4.2.4 Compressing ISA Solutions

Our task in Stage 5 of Figure 4.2 is to compress the ISA solutions corresponding to each of our cardinality vectors $\boldsymbol{d}^{\mathrm{I}}, \ldots, \boldsymbol{d}^{\mathrm{V}}$. In accordance with (4.2) we find the total ISA-solution coding cost $L_I(D, M)$ by summing two costs: the cost $L_I(M)$ to compress the ISA model, and the cost $L_I(D|M)$ to compress the ISA data with respect to the ISA model:

$$L_I(D, M) = L_I(M) + L_I(D|M) \tag{4.4}$$

$L_I(M)$: Compressing the ISA Model

The elements of the ISA model requiring compression are 1) the subspace cardinalities vector $\boldsymbol{d}$ and 2) the demixing matrix $\mathbf{W}$. In tune with the MDL principle we wish to penalize clustering models with a higher complexity, which in our case corresponds to those models with a larger number of subspaces. However, we do not wish to bias solutions for which structure can be found in the demixing matrix $\mathbf{W}$, as this matrix is simply a rotation matrix and we view all such matrices to be equally complex. Our coding cost for the ISA model is hence the sum of a fixed cost for $\mathbf{W}$ and a variable cost for $\boldsymbol{d}$ (depending on its length q):

$$L_I(M) = \frac{m^2}{2} \cdot \log_2(n) + (q+1) \cdot \log_2(m) \tag{4.5}$$

Here we have used the technique from [55] (stating that a cost of $\frac{p}{2} \cdot \log_2(n)$ is sufficient in terms of accuracy for p real parameters modeling n data objects) for parameter-encoding the mixing matrix. For the vector $\boldsymbol{d}$ we need to encode q integers and the value of q itself, all of which are upper-bounded by m (hence requiring $\log_2(m)$ bits each).

$L_I(D|M)$: Compressing the ISA Output Data

We seek to compress the matrix $\mathbf{S}$, that is, our "demixed" data. To this end we require a joint probability density function for each subspace's set of random variables. Again we use Kernel Density Estimation for this task (Section 2.6). Each KDE model then needs to be likewise encoded. A zero-mean KDE model using an elliptical Gaussian kernel consists

only of the diagonal covariance matrix $\mathbf{\Sigma}$. Over all subspaces we hence need to encode a total of m covariance matrix entries, so the cost C_k of encoding all KDE models is

$$C_k = \frac{m}{2} \cdot \log_2(n). \tag{4.6}$$

Using KDE we obtain estimates $\widehat{f}_{\vec{\mathbf{S}}^1}, \ldots, \widehat{f}_{\vec{\mathbf{S}}^q}$ for the probability density functions of the random vectors $\vec{\mathbf{S}}^1, \ldots, \vec{\mathbf{S}}^q$. These random vectors contain the same random variables as the subspaces $\mathcal{S}_1, \ldots, \mathcal{S}_q$; the n realizations of these random vectors (our transformed original observations) are encapsulated by the sub-matrices $\mathbf{S}^1, \ldots, \mathbf{S}^q$. Using entropy coding, object j within subspace i (denoted with the vector $\mathbf{s}^i_{\cdot j}$) is assigned a coding cost of $\log_2(1/f_{\vec{\mathbf{S}}^i}(\mathbf{s}^i_{\cdot j}))$. Substituting our KDE estimates and aggregating for all subspaces and objects gives us the coding cost for the demixed data,

$$C_{\mathbf{S}} = \sum_{i=1}^{q} \sum_{j=1}^{n} \log_2 \frac{1}{\widehat{f}_{\vec{\mathbf{S}}^i}(\mathbf{s}^i_{\cdot j})}. \tag{4.7}$$

The coding cost of our data transformed with respect to the ISA model is then

$$L_I(D|M) = C_k + C_{\mathbf{S}}. \tag{4.8}$$

4.2.5 Compressing the Subspace Clusterings

In Stage 7 of Figure 4.2 it is in theory possible to use any full-space clustering technique. In this paper, we use EM clustering with a hard-cluster assignment after each expectation-maximization step (henceforth denoted EM_h). We compute the total clustering coding cost $L_C(D, M)$ for *one* subspace (we cluster independently in each subspace, choosing the most suitable k value for each) by summing two costs: the cost $L_C(M)$ to compress the clustering model, and the cost $L_C(D|M)$ to compress the clustered data with respect to the clustering model:

$$L_C(D, M) = L_C(M) + L_C(D|M) \tag{4.9}$$

$L_C(M)$: Compressing the Clustering Model

EM_h requires two parameters per cluster: the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. In our context, clustering in subspace $\mathcal{S}_i$ sees each mean vector $\boldsymbol{\mu}$ described by one parameter for each of its d_i dimensions, and each covariance matrix $\boldsymbol{\Sigma}$ described with one parameter for each of $d_i \cdot (d_i + 1)/2$ entries. (We exploit symmetry here.) Using the parameter-coding model from Section 4.2.4, we find the cost to compress the clustering model in subspace $\mathcal{S}_i$ as

$$L_C(M) = \sum_{j=1}^{k} \frac{(d_i^2 + 3d_i)}{4} \cdot \log_2(n_j), \tag{4.10}$$

where k denotes the number of clusters and n_j denotes the number of objects in the j-th cluster.

$L_C(D|M)$: Compressing the Clustered Data

Assuming EM_h is given a subspace data matrix $\mathbf{S}^i \in \mathbb{R}^{d_i \times n}$ corresponding to subspace $\mathcal{S}_i$, it partitions that data into k clusters. Let $\boldsymbol{c} \in \{1, \ldots, k\}^n$ hold the cluster-assignments for the n objects (that is, object $p \in \{1, \ldots, n\}$ is assigned to cluster c_p). Let $\boldsymbol{n} \in \mathbb{N}^k$ hold the object counts for the k clusters (that is, cluster $j \in \{1, \ldots, k\}$ has n_j objects). We encode each object $\boldsymbol{o}$ in cluster j dimension-wise using Gaussian probability density functions $\widehat{f}_1^j(o), \ldots, \widehat{f}_{d_i}^j(o)$. For each of these distribution functions we estimate the mean and variance from the cluster-object values in that dimension.

The cost for compressing an object $\boldsymbol{o}$ in cluster j of subspace i then considers the cost of its cluster assignment and the cost of its deviation from the model in each dimension:

$$C_H(\boldsymbol{o}; n_j, \widehat{f}_1^j(o), \ldots, \widehat{f}_{d_i}^j(o)) = \log_2\left(\frac{n}{n_j}\right) + \sum_{p=1}^{d_i} \log_2\left(\frac{1}{\widehat{f}_p^j(o_p)}\right)$$

We aggregate over all clusters and their objects to arrive at the cost for one subspace $\mathcal{S}_i$:

$$L_C(D|M) = \sum_{j=1}^{k} \left(\sum_{\boldsymbol{o} \in \{\boldsymbol{s}^i_{\cdot r} | c_r = j\}} C_H(\boldsymbol{o}; n_j, \widehat{f}_1^j(o), \ldots, \widehat{f}_{d_i}^j(o)) \right) \tag{4.11}$$

4.3 Algorithm

Here we detail ISAAC in two parts, namely 1) automating ISA to find the best $\mathbf{S}$, $\mathbf{W}$ and $\boldsymbol{d}$ (Stages 1–6 in Figure 4.2), and 2) automating the clustering in each subspace (Stage 7 in Figure 4.2). We provide an implementation in our supplement[5].

Algorithm 2: Parameter-free ISA

Input: Data $\mathbf{X} \in \mathbb{R}^{m \times n}$.
Output: Matrices $\mathbf{W}_b$ and $\mathbf{S}_b$ ($\mathbf{S}_b = \mathbf{W}_b\mathbf{X}$), vector $\boldsymbol{d}_b$.

$\boldsymbol{d} \leftarrow (1, \ldots, 1) \in \{1\}^m$; /* First candidate */
$\boldsymbol{a} \leftarrow (\{X_1\}, \ldots, \{X_m\})$;
$(\mathbf{S}, \mathbf{W}) \leftarrow \mathbf{ISA}(\mathbf{X}, \boldsymbol{d})$; /* First ISA result */
$c_b \leftarrow L_I(\mathbf{S}, \{\boldsymbol{d}, \mathbf{W}\})$; /* Eq. (4.4) */
$\mathbf{S}_b \leftarrow \mathbf{S}$, $\mathbf{W}_b \leftarrow \mathbf{W}$, $\boldsymbol{d}_b \leftarrow \boldsymbol{d}$; /* Track best */
while $q > 1$ **do** /* $q =$ length($\boldsymbol{d}$) */
    /* $\boldsymbol{c}$ has l_c tuples: each a merge candidate with C_M value (Eq. (4.3)) */
    $\boldsymbol{c} \leftarrow ((a_1, a_2, t_1 = C_M(a_1, a_2)),$
        $\ldots, (a_{q-1}, a_q, t_{l_c} = C_M(a_{q-1}, a_q)))$;
    /* Abort if nothing to merge */
    if $\min(t_1, \ldots, t_{l_c}) > 0$ **then** break;
    $\boldsymbol{c} \leftarrow$ **order_ascenting_by_t_value** ($\boldsymbol{c}$); /* Sort */
    $\boldsymbol{d} \leftarrow ()$, $\boldsymbol{a} \leftarrow ()$; /* New $\boldsymbol{d}, \boldsymbol{a}$ candidates */
    for $k \leftarrow 1$ **to** l_c **take** c_i **as** $(\mathcal{X}_i, \mathcal{X}_j, t_k)$ **and do**
        if $t_k < 0$ **and** $(\mathcal{X}_i \cup \mathcal{X}_j) \cap \left(\bigcup_{\mathcal{X} \in \boldsymbol{a}} \mathcal{X}\right) = \emptyset$ **then**
            /* Merge $\mathcal{X}_i$ and $\mathcal{X}_j$ */
            $\boldsymbol{a}$.**push** $(\mathcal{X}_i \cup \mathcal{X}_j)$, $\boldsymbol{d}$.**push** $(|\mathcal{X}_i \cup \mathcal{X}_j|)$;
        else
            $\boldsymbol{a}$.**pushAll** $(\mathcal{X}_i, \mathcal{X}_j)$, $\boldsymbol{d}$.**pushAll** $(|\mathcal{X}_i|, |\mathcal{X}_j|)$;
    $(\mathbf{S}, \mathbf{W}) \leftarrow \mathbf{ISA}(\mathbf{X}, \boldsymbol{d})$; /* ISA result */
    if $L_I(\mathbf{S}, \{\boldsymbol{d}, \mathbf{W}\}) < c_b$ **then**
        $c_b \leftarrow L_I(\mathbf{S}, \{\boldsymbol{d}, \mathbf{W}\})$;
        $\mathbf{S}_b \leftarrow \mathbf{S}$, $\mathbf{W}_b \leftarrow \mathbf{W}$, $\boldsymbol{d}_b \leftarrow \boldsymbol{d}$;

[5] https://github.com/yeweiysh/ISAAC

4.3.1 Parameter-free ISA

The commented pseudo-code for our parameter-free ISA approach is given in Algorithm 2. Lines 7– 16 represent the procedure for generating candidate subspace cardinality vectors $\boldsymbol{d}$ (Section 4.2.3). At line 7 we build a vector $\boldsymbol{c}$ of tuples that encapsulates the information we showed with an example in Table 4.1: each $\boldsymbol{c}$ entry contains a potential merge candidate and its corresponding "dependency indicator" value C_M (Equation 4.3). We sort $\boldsymbol{c}$ based on these values (line 10) to ensure that the most dependent subspaces are merged first, and populate a new $\boldsymbol{d}$ candidate (and its corresponding subspaces $\boldsymbol{a}$) by merging those subspaces which have not already been merged and which also satisfy our merge condition (line 13). We execute ISA for the new candidate $\boldsymbol{d}$ vector (line 17) and greedily set the result as our local optimum on improvement (lines 18– 20).

4.3.2 Independent Clustering

Given $(\mathbf{W}_b, \mathbf{S}_b, \boldsymbol{d}_b)$ from Algorithm 2, we cluster in each subspace $\mathcal{S}_1, \ldots, \mathcal{S}_q$ independently. To automatically find the number of clusters in subspace $\mathcal{S}_i$, we perform EM_h (EM clustering with a hard-cluster assignment after each expectation-maximization step) with $k = 2$ and increment k until Equation (4.9) ceases to decrease (descending search) or until $k > n$. The solution is that which corresponds to the minimum coding cost (Equation 4.9). This procedure is repeated for each subspace. The pseudo-code is given in Algorithm 3.

4.3.3 Convergence and Complexity

We first consider the complexity of a single call to ISA (lines 3 and 17). As discussed in Section 2.1, we choose with [110] an ISA implementation which supports heterogeneous subspace dimensionalities. It relies on the *ISA separation principle*, which proposes that the ISA task can be solved by ICA preprocessing and subsequent clustering of the ICA components into statistically-independent groups. The ICA implementation (FastICA) has guaranteed convergence and a worst-case runtime in $\mathcal{O}(nm)$ (assuming its iteration

Algorithm 3: Independent Clustering

Input: $\mathbf{S}_b$ and $\boldsymbol{d}_b$
Output: the cluster indicator matrix $\mathbf{I}$

```
1  I ← ∅, tmp ← ∅, l(1) = 1e10;
2  for i ← 1to length(d_b) do
3      k ← 2, c ← ∅;
4      while true do
5          tmp ← EM_h(S_b^i, k);                    /* Temporary Cluster indicator */
6          l(k) ← evaluating the coding cost by Eq. (4.11);
7          if l(k) < l(k − 1) and k <= n then
8              c ← tmp;
9              k ← k + 1;
10         else
11             break;
12     I(i, :) ← c;        /* To create a matrix by stacking c along the rows */
```

count to be bounded). Given the ICA result, ISA proceeds to group the components into subspaces. This grouping is equivalent to multiplying the ICA mixing matrix $\mathbf{W}$ by a permutation matrix, for which there quickly become an intractable number of possibilities for large m [110]. The implementation hence uses a greedy approach for finding an optimal permutation matrix: it iterates over all pairs of components between subspaces (the count of which is in $\mathcal{O}(m^2)$ for our initial $\boldsymbol{d}$ vector), swapping them when beneficial. It does this for a maximum fixed number of iterations, thus has guaranteed convergence with a worst-case run-time complexity in $\mathcal{O}(nm^2)$. Whitening data, a preprocessing step in ISA, likewise has complexity in $\mathcal{O}(nm^2)$, so the overall run-time complexity of a call to ISA is in $\mathcal{O}(nm^2)$.

Next, we consider the evaluation of coding cost L_I for an ISA solution (lines 4 and 18). For Kernel Density Estimation we use the tractable solution discussed in [1] with time complexity in $\mathcal{O}(nm)$, avoiding the naïve approach's quadratic cost in n. After we have the KDE estimate, equation (4.7) is evaluated in $\mathcal{O}(nm)$ time. The run-time growth rate for evaluating L_I is hence in $\mathcal{O}(nm)$.

On line 7 we compute dependency indicators for each pair of subspaces. The patho-

logical case here is for a candidate $\boldsymbol{d} = (\sqrt{m}, \ldots, \sqrt{m}) \in \mathbb{Z}_+^{\sqrt{m}}$, which implies $\mathcal{O}(m)$ pairs for which we need to calculate the measure C_M. For each combination we again depend on KDE, requiring $\mathcal{O}(n\sqrt{m})$ for each subspace. The run-time of line 7 hence grows with $\mathcal{O}(nm\sqrt{m})$ in the worst case.

Finally, from Section 4.2.3 we know that the main loop (line 6) has guaranteed worst-case convergence in m iterations (the pathological case for the number of $\boldsymbol{d}$-vector candidates). Algorithm 2 hence has a worst-case run-time complexity in $\mathcal{O}(m\,(nm\sqrt{m} + nm^2 + nm)) = \mathcal{O}(nm^3)$.

For a given subspace and k value, clustering with EM_h has a run-time in $\mathcal{O}(nmk)$ (again assuming a bounded number of E-M iterations). Our search for the optimal k for a given subspace introduces an additional loop with worst-case n iterations (again a pathological case; practically the number of iterations is around a few dozen). In the worst-case we also have m subspaces in which to perform clustering, so the worst-case computational complexity of the clustering stage is in $\mathcal{O}(n^2m^2k)$. Assuming the worst-case for both stages we find the **worst-case ISAAC run-time complexity** as $\mathcal{O}(nm^3 + n^2m^2k)$.

4.4 Experimental Evaluation

We present comparisons with both axis-parallel subspace clustering and arbitrarily-oriented multiple-clustering methods. With INSCY [47], STATPC [43] and RESCU [31] we have axis-parallel methods which, like ISAAC, focus on reducing redundancy between clusterings. With ORTH1, ORTH2 (*orthogonal clustering*, and *clustering in orthogonal subspaces*, both presented in [106]) and MSC [24] we likewise have three non-redundant methods for finding arbitrarily-oriented subspaces. In addition, we also compare to the baseline of ISAAC, i.e. ISA plus EM_h, denoted by ISAEM_h. The parameters for subspace clustering methods are set according to their original papers. For MSC, we provide the true number of subspaces and clusters. In real-word data, we set the number of subspaces for MSC equal to those found by ISAAC. For ORTH1 and ORTH2, we provide the true number of clusters. For ISAEM_h, we provide the true subspace cardinalities and the true

number of clusters in each subspace. All experiments were done on the same machine (Intel Quad i7-3770, 3.4 GHz, 32 GB RAM). We report the "pair counting F1-measure" [28], the harmonic mean of precision and recall, for quality in all cases. All synthetic data can be found in the supplement.

4.4.1 Synthetic Data

Cluster Quality

To generate synthetic dataset, we assume we have q two-dimensional "ground truth" subspaces with a clustering in each. Half of these subspaces contain four clusters; the remaining subspaces contain six (varying size of clusterings). Correlations between the observations in each cluster are obtained by 1) starting with n observations generated from uncorrelated standard normal variables, 2) choosing a correlation matrix $\mathbf{C}$ and scalar $r \in [0, 1]$ such that $c_{i,i} = 1$ and $c_{i,j} = r, i \neq j$, and 3) applying $\mathbf{C}$'s Cholesky transformation to the observations. Clusters are then positioned in the two-dimensional space by their respective centers – each a random sample from the set $[20, 80]^2$. To simulate non-redundancy between clusterings, we randomly permute the object IDs in each subspace before merging them to a full-space dataset. We generate another two synthetic datasets with three- and five-dimensional "ground truth" subspaces using the same method.

Figure 4.3 shows the effect of increasing the number of subspaces q (here $n = 1200$). A point on the plot represents the mean value over ten independently-generated datasets; the error bars (shown only for every second q value to reduce clutter) represent one standard deviation in each direction. Note also that we omit the poorer of ORTH1 and ORTH2 in each plot to reduce clutter. Given the true subspace cardinalities and the true number of clusters in each subspace, ISAEM$_h$ consistently outperforms. ISAAC's outperforms on the synthetic datasets with 2D and 3D "ground truth" subspaces, despite not being "helped" with the provision of any input parameters (the other multiple-clustering techniques require the correct number of subspaces and/or clusters). On the synthetic dataset with 5D "ground truth" subspaces, MSC and ORTH1 have a similar performance compared to

ISAAC. The first row of Figure 4.4 shows the four independent subspaces contained in the synthetic dataset with 2D "ground truth" subspaces. The second row of Figure 4.4 demonstrates the four independent subspaces (corresponding to the subspaces depicted in the first row) and clusterings found by ISAAC. Compared to the original subspaces, we can see from the figure that the found subspaces are rotated because ISA tries to find arbitrarily-oriented subspaces which are a linear combination of the original ones. When checking the found subspace (the second row, second column of Figure 4.4, which corresponds to the subspace shown in the first row, second column of Figure 4.4), we find the top two very-close clusters are not separated by ISAAC. In other independent subspaces, clusters are separated cleanly.

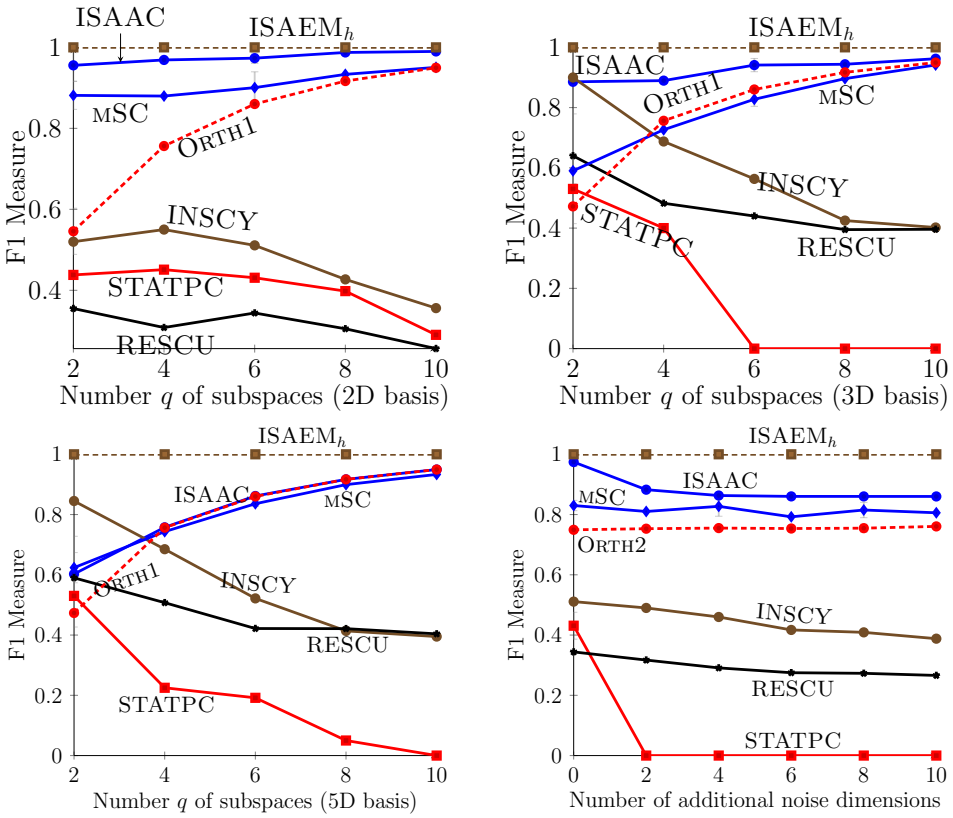


Figure 4.3: Variation in quality. From the left to the right, (1), (2) and (3): increasing subspaces q and (4) additional noise dimensions.

Robustness against noise attributes

Fixing $q = 6$ and otherwise following the same generation process, we measure the effect of adding noise dimensions to our data (Figure 4.3 (2D basis)). Keeping in mind that ISAAC is an automated technique that requires no parameters, it remains promisingly robust to an increasing number of noise attributes and continues to achieve an F1 value superior to the competition. Of note is that STATPC is particularly susceptible to noise.

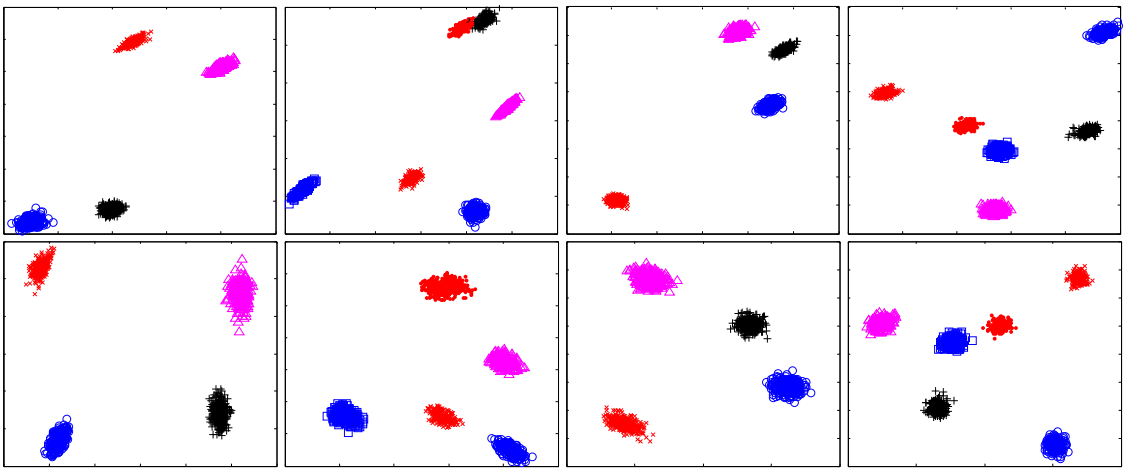


Figure 4.4: Cluster Quality. the first row: synthetic data with four two-dimensional independent subspaces; the second row: the four independent subspaces and clusterings found by ISAAC.

Scalability

Figure 4.5 shows how execution time grows with increasing dataset dimensions n and m. For the case of varying n we fix $m = 4$ (two two-dimensional subspaces, each with three clusters generated as above). For the case of varying m we fix $n = 1000$ and assign ten dimensions to each subspace (giving ten subspaces for $m = 100$, each containing two clusters). It is important to note that ISAAC is the only fully-automatic method being evaluated: it invests time to search for appropriate model-values in various stages of the framework (this additional effort is included in the experimental results). Despite this – and ignoring constant factors – ISAAC "holds its own" in terms of the run-time growth rate. Its observed quadratic growth rate in n is comparable to MSC, INSCY and STATPC

(ORTH2 behaves linearly, and RESCU's runtime grows with n^3). ISAAC's runtime grows proportionally to m^3, and is hence faster than INSCY and comparable to STATPC and (eventually) MSC. RESCU behaves quadratically.

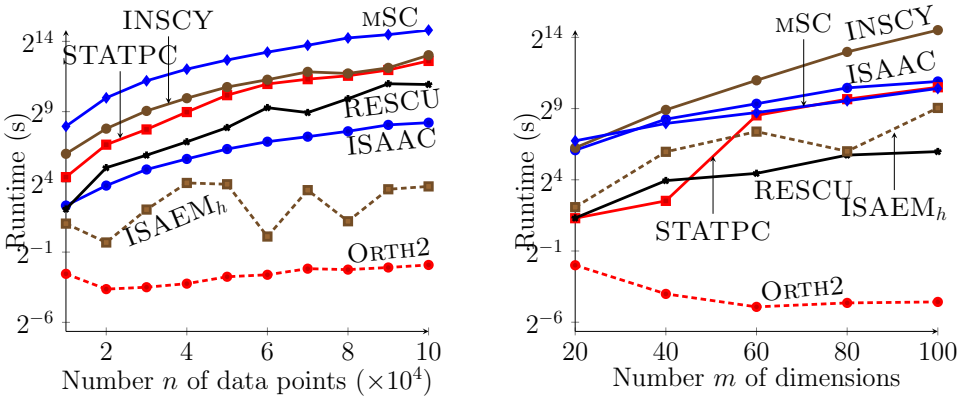


Figure 4.5: Variation in runtime. From the left to the right, increasing number of (1) data objects n and (2) dimensions m.

4.4.2 Experiments on Real-world Data

Quantitative Analysis

We compare the F1 measure on nine real-world datasets. The Breast (Wisconsin Diagnostic), Ecoli, Spam, Shuttle, Musk and Connectionist Bench (Sonar, Mines vs. Rocks) datasets are from the benchmark UCI repository. The metabolic dataset is from a PKU newborn screening [12]. Dancing Stick Figures (DSF) [86] is a multi-view dataset with 900 samples of 20×20 images across nine stick figures (Figure 4.6). Amsterdam Library of Object Images (ALOI) [48] collection consists of images of 1000 common objects taken from various angles and under various illumination conditions. We chose four different objects (Figure 4.7) with all their images taken from different viewing directions. We extracted color and texture features with 611 dimensions for each image using the method proposed in [82] (code can be found here[6]). Then for DSF and ALOI, we further apply

[6]http://www.cat.uab.cat/Research/ColorTextureDescriptors/

PCA as a preprocessing step (also used in [72, 86]), retaining at least 90% of the variance (five principal components). All data is available in our supplement.

Figure 4.6: Nine raw samples from the Dancing Stick Figures.

Figure 4.7: Four objects of different shapes (ball and cylinder) and colors (green and red) from ALOI.

ISAAC is deployed in our proposed automated fashion (parameter-free). ORTH1 and ORTH2 require the number of clusters, so we use the number of class labels in each respective dataset. For DSF we inform ORTH1 and ORTH2 that there are three clusters in each subspace (based on the qualitative intuition in the next section). In addition to the number of clusters, MSC requires the number of subspaces – here we provide it with the same value found by ISAAC in all cases.

Table 4.2 reports the F1 measure for each dataset and algorithm. We see that ISAAC obtains a stronger F1 measure in all cases, even outperforming techniques like MSC, ORTH1 and ORTH2 which have the advantage of being given the correct number of clusters as a parameter.

Qualitative Analysis

We now qualitatively interpret and compare the results for Dancing Stick Figures and Amsterdam Library of Object Images multi-view datasets. For the other datasets, since we do not have "ground truth" subspaces, we omit their interpretation.

Dancing Stick Figures Dataset In the DSF data, ISAAC finds three independent subspaces. The first and second subspace contain three clusters and the third contains

Table 4.2: F1 measure on real-world data (with dimensions (n;m)). $*$ failed because of non-trivial bugs in the OpenSubspace implementation [32].

Dataset	Metabolic (709;10)	Ecoli (336;7)	Breast (569;30)	Spam (4601;57)	Shuttle (43500;9)
ISAAC	**0.81**	**0.71**	**0.71**	**0.69**	**0.78**
MSC	0.41	0.50	0.68	0.68	0.65
ORTH1	0.54	0.43	0.69	0.68	0.63
ORTH2	0.54	0.42	0.69	0.68	0.65
STATPC	0.44	0.32	0.61	0.68	0.19
INSCY	0.29	0.11	0.65	0.01	–*
RESCU	0.26	0.07	0.39	–*	–*
Dataset	C. Bench (208;60)	Musk (476;166)	DSF (900;5)	ALOI (288;5)	
ISAAC	**0.66**	**0.67**	**0.86**	**0.87**	
MSC	0.61	0.65	0.74	0.79	
ORTH1	0.60	0.65	0.71	0.67	
ORTH2	0.52	0.65	0.74	0.62	
STATPC	0	0	0.60	0.42	
INSCY	0	–*	0.62	0.27	
RESCU	0	–*	0.58	0.33	

four. Figure 4.8 depicts the means of the detected clusters in the first and second subspaces (we don't show clusters in the third subspace because they are not very interpretable). We clearly see a compelling separation into upper- and lower-body motions (two non-redundant views on the data). In comparison, we see in Figure 4.8 the two subspaces found by ORTH2 (the best of the competition from Table 4.2). Here ORTH2 fails to detect any intuitive and convincing perspectives.

Amsterdam Library of Object Images Dataset For the ALOI data, ISAAC finds three independent subspaces. The means of the detected clusters in the subspaces are depicted in Figure 4.9. Again the subspaces show three interesting perspectives on the data: one groups by *shape* (cylinder and ball), another by *color* (red and green) and the other by *size* (small and big). It is very interesting that ISAAC detects the subspace in which objects of similar sizes are clustered. In comparison, the two subspaces detected by MSC (the best of the competition from Table 4.2) show that it fails to identify the

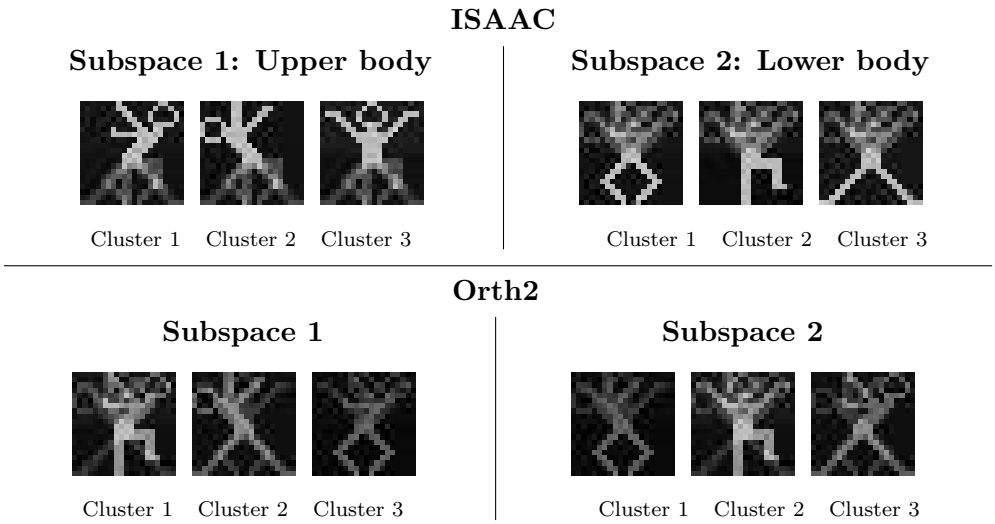


Figure 4.8: The means of the clusters detected by ISAAC (top) and ORTH2 (bottom) in two subspaces (Dancing Stick Figures data). ISAAC identifies clear upper- and lower-body perspectives.

separation between color and shape.

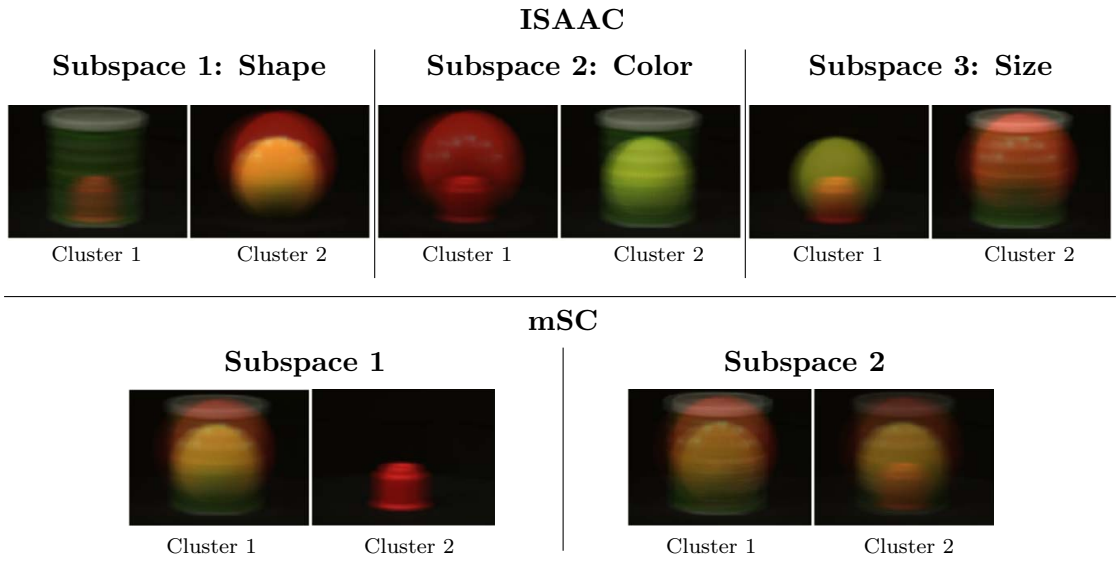


Figure 4.9: The means of the clusters detected in the ALOI data by ISAAC (three subspaces, top) and MSC (two subspaces, bottom). ISAAC successfully identifies subspaces which partition color and shape. In addition, it finds subspace in which objects of similar sizes are grouped together. (*best viewed in color*)

4.5 Related Work and Discussion

Subspace Clustering finds clusters in projections of original data space. The most related subspace clustering algorithms to our approach are the redundancy-reducing subspace clustering algorithms, which include INSCY [47], STATPC [43] and RESCU [31]. INSCY achieves efficient subspace clustering by using depth-first processing with in-process-removal of redundancy. STATPC approximately extracts a suitable reduced, non-redundant set of statistically significant regions to detect clusters. RESCU involves a global optimization that detects the most interesting non-redundant subspace clusters by inspecting overlapping clusters and reducing the results to a manageable size. Our algorithm ISAAC combines MDL with ISA in an automatic framework to find statistically independent subspaces in which clusterings are non-redundant. Since ISAAC is only interested in "independence", it may neglect searching some arbitrarily-oriented subspaces which are not independent but in which clusters may really reside.

Multiple Clustering is also a related field for our method. Multiple clustering seeks to partition a given set of objects in different ways, which represents different perspectives of the data. COALA [29] generates multiple clusterings by using instance level constrains. NACI [103] is driven by using mutual information to optimize the dual objective functions of both quality and dissimilarity. minCEntropy [72] presents a computationally efficient nonparametric entropy estimator to quantify both clustering quality and distinctiveness. However, the above methods can only generate two alternative clusterings. MVGen [87] generates multiple clusterings of data by using multiple mixture models. MVGen uses the iterated conditional modes (ICM) principle and adopts Bayesian model selection to make a balance between the complexity of the model and its goodness of fit. SMVC [86] integrates semi-supervised clustering with multiple clustering and uses variational Bayesian methods for efficient learning. However, the purposes of MVGen and SMVC are to detect multiple, overlapping clustering views which are not non-redundant. Multiple Stable Clustering [52] detects multiple stable clusterings using the idea of clustering stability based on Laplacian Eigengap. But the found multiple stable clusterings cannot guarantee diversity, i.e., some

clusterings are redundant and potentially difficult to interpret. mSC [24] integrates the relaxed spectral clustering objective with the Hilbert-Schmidt independence criterion (HSIC) to find multiple non-redundant views, and then uses spectral clustering to find clusters in each view. Orthogonal projection clustering (OPC) [106] uses two strategies, (1) orthogonal clustering (Orth1), and (2) clustering in orthogonal subspaces (Orth2), to partition data to achieve multiple clusterings. The last three non-redundant multiple clustering algorithms achieve the same goal as ISAAC. Differing from Orth1 which directly seeks non-redundant clusterings, ISAAC indirectly seeks multiple non-redundant clusterings by using ISA to generate independent subspaces. Thus, clusterings in those subspaces are independent (non-redundant). The strategy is very similar to those of Orth2 and mSC which also firstly seek independent or orthogonal subspaces followed by clustering in those subspaces.

4.6 Summary

We have presented ISAAC, a parameter-free technique for generalized subspace clustering. It can find multiple clusterings in arbitrarily-oriented subspaces of heterogeneous dimensionality such that pairwise clusterings are highly statistically-independent (non-redundant) and contain potentially-differing numbers of clusters. To this end ISAAC combines Independent Subspace Analysis (ISA) and clustering in one automatic framework. Automation is made possible through the MDL principle, where model parameters are selected by balancing accuracy and complexity. An efficient, MDL-driven greedy search heuristic helps ISAAC to find the best space partition. Experiments on synthetic and real-world data show promising comparisons to state-of-the-art methods with respect to efficiency and effectiveness.

Chapter 5

Finding Cohesive Clusters in Attributed Graphs

In addition to the numerical data which is mined in Chpater 3 and Chapter 4, graph data such as social graphs, citation graphs, protein-protein interaction graphs, etc., are prevalent in the real world. And graph vertices are often associated with attributes. For example, in addition to their connection relations, people in friendship networks have personal attributes, such as interests, ages, and living places. Such graphs are called attributed graphs. The detection of clusters in attributed graphs is of great practical application, e.g., targeted ads. Attributes and edges often provide complementary information. The effective use of both types of information promises meaningful results. In this chapter, we propose a method called UNCut (for Unimodal Normalized Cut) to detect cohesive clusters in attributed graphs. A cohesive cluster is a subgraph that has densely connected edges and has as many homogeneous (unimodal) attributes as possible. We adopt the *normalized cut* to assess the density of edges in a graph cluster. To evaluate the unimodality of attributes, we propose a measure called *unimodality compactness* which exploits Hartigans' dip test. Our method UNCut integrates the *normalized cut* and *unimodality compactness* in one framework such that the detected clusters have low *normalized cut* and *unimodality compactness* values. Extensive experiments on various synthetic and real-world data ver-

ify the effectiveness and efficiency of our method UNCut compared with state-of-the-art approaches.

Parts of the materials presented in this chapter have been published in [99], where Wei Ye developed the main idea, implemented the algorithm, did the most parts of the experimental evaluation, and wrote the major parts of the paper; Linfei Zhou and Xin Sun did some experiments and wrote some parts of the paper; Christian Böhm and Claudia Plant supervised the project and contributed to the development of the idea and paper writing.

"*Wei Ye, Linfei Zhou, Xin Sun, Claudia Plant, Christian Böhm. Attributed Graph Clustering with Unimodal Normalized Cut. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD), 2017.*"

5.1 Motivation

Real-world graphs (networks) tend to have attributes associated with vertices. For example, in social networks such as Facebook, Google+ and Twitter, users have their personal information, e.g., interests, ages, living places, etc., in addition to their friendship relationships. Proteins in a protein-protein internation network may have gene expressions in addition to their interaction relations. Such graphs are often referred to as *attributed graphs* in which vertices represent entities, edges represent their relations and attributes describe their own characteristics. Often the attributes and edges provide complementary information [40]. Neither can we infer vertex relationships from their attributes nor vice versa. Nevertheless, both types of information can be valuable for the detection of clusters in attributed graphs. Traditional methods for attributed graph clustering consider all attributes to compute the similarity. However, some attributes may be irrelevant to the edge structure and thus clusters only exist in the subset (subspace) of attributes. Currently, several methods have been proposed to detect subspace clusters in attributed graphs, such as CoPaM [40] and SSCG [85]. CoPaM uses various pruning strategies to find maximal cohesive patterns in the subspace of attributes. One major problem with CoPaM is that

it outputs a large number of clusters which have few vertices or attributes and which overwhelm data analysts. As for SSCG, it needs to eigen-decompose the graph Laplacian matrix and to update the subspace dependent weight matrix in every iteration, which is not scalable for large-scale graphs. How to effectively find clusters in attributed graphs remains a big challenge.

In this work, we develop an effective and efficient method to find cohesive clusters in attributed graphs. A cohesive cluster is a subgraph that has densely connected edges and has as many homogeneous (unimodal) attributes as possible. Why do we prefer to find cohesive clusters? One proper answer is that the more cohesive a graph cluster is, the more information it can reveal. For example, in social networks, if social networking advertisers know more characteristics of people, they can do targeted ads more precisely. Figure 5.1 demonstrates an example social network with three attributes (age, sport time per week, and studying time per week) associated to each vertex. The task is to divide the network into two distinct parts which have as many homogeneous (unimodal) attributes as possible. In this example social network, we have two candidate partitions, i.e., by the orange dashed line and by the blue dashed line. The orange dashed line divides the network into two cohesive clusters $\mathcal{C}_1 = \{0, 1, 2, 3, 4, 5, 6\}$ that is cohesive on the attribute studying time and $\mathcal{C}_2 = \{7, 8, 9\}$ that is cohesive on all the attributes. The blue dashed line divides the network into another two cohesive clusters $\mathcal{C}_3 = \{0, 1, 2, 3, 4\}$ which is cohesive on all the attributes and $\mathcal{C}_4 = \{5, 6, 7, 8, 9\}$ which is cohesive on the attributes age and sport time. Compared with clusters $\mathcal{C}_1$ and $\mathcal{C}_2$, clusters $\mathcal{C}_3$ and $\mathcal{C}_4$ are more cohesive. Although the normalized cut value increases a little bit from 0.536 to 0.559, the *unimodality compactness* (see Section 5.2) value of attributes dramatically decreases from 3.289 to 1.230. The *unimodal normalized cut* (see Section 5.2) value of the partition by the blue dashed line is 0.895 and that of the partition by the orange dashed line is 1.913. Thus, we prefer clusters $\mathcal{C}_3$ and $\mathcal{C}_4$ to clusters $\mathcal{C}_1$ and $\mathcal{C}_2$.

The contributions can be summarized as follows:

- **We introduce the univariate statistic hypothesis test called Hartigans' dip**

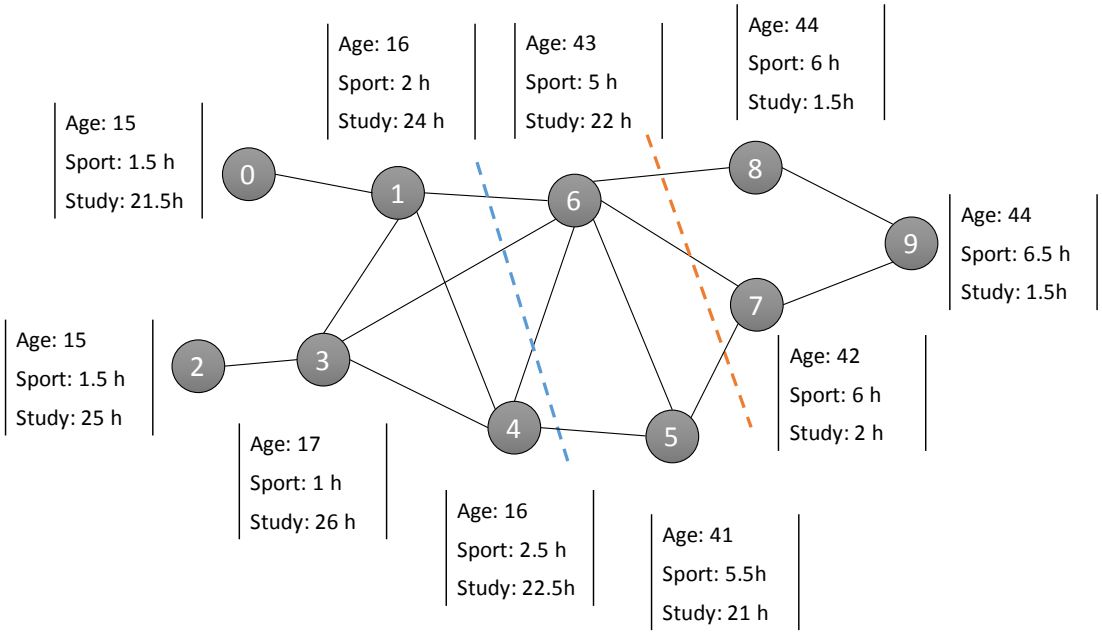


Figure 5.1: An example social network.

test [51] to the problem of attributed graph clustering.

- **We achieve the cohesive cluster detection by developing an objective function which integrates the proposed measure *unimodality compactness* with the *normalized cut***. The *unimodality compactness* takes advantage of Hartigans' dip test to measure the degree of the unimodality of attributes in a graph cluster.

- **We show the effectiveness and efficiency of our method UNCut by conducting extensive experiments on synthetic and real-world graphs**.

5.2 Unimodal Normalized Cut

Our objective is to detect cohesive graph clusters which have densely connected edges (low *normalized cut* value) and have as many homogeneous (unimodal) attributes as possible (low *unimodality compactness* value). To achieve the goal, we need to take both the edge structure and attribute information into account. If we eigen-decompose the Laplacian

matrix associated with the edge structure to generate n eigenvectors, the k eigenvectors associated with the k smallest eigenvalues near optimally partition the graph into k subgraphs. However, the procedure does not consider the attribute information. Since each eigenvector bisects the graph into two clusters, our idea is to develop a measure to simultaneously evaluate the density of the edge structure and the homogeneity of vertex attributes of a graph cluster derived from the eigenvector. To this end, we first propose a measure called *unimodality compactness* to assess the homogeneity of attributes of a graph cluster. Then we integrate it with the *normalized cut* and call the combination *unimodal normalized cut*. We select k eigenvectors associated with the k smallest *unimodal normalized cut* values to partition the graph. In the following, we describe our idea in detail. But first let us give the definitions as follows:

Definition 5 *A **unimodal graph cluster** is defined as a set of vertices with at least one attribute following unimodal distributions.*

To compute the degree of the unimodality of a graph cluster, we devise a measure called *unimodality compactness* using the dip test on each attribute of the cluster.

Definition 6 *Given a cluster of vertices $\mathcal{S}$ with number $c > 0$ of unimodal attributes, the unimodality compactness is defined as,*

$$UC(\mathcal{S}) = \log_2 \frac{d}{c} + \frac{1}{c} \sum_{i=1}^{c} D(F_i) \tag{5.1}$$

where d is the number of attributes, F_i is the empirical distribution function of the i-th unimodal attribute of $\mathcal{S}$ and $D(F_i)$ is the dip test of F_i.

The first summand measures the number of unimodal attributes of a cluster. The second summand measures the average dip test of these unimodal attributes. This measure prefers the cluster that has more unimodal attributes with lower average dip test. Note that the multimodal (irrelevant) attributes are not considered in the computation. If a graph cluster only has one unimodal attribute, its *unimodality compactness* is close to $\log_2 d$ because the

second summand in Equation 5.1 is very low. If there is no unimodal attribute in a cluster, we simply set its *unimodality compactness* to $2\log_2 d$. When d is large and $c = 1$, the value of $\frac{d}{c}$ is also large. To reduce the effect of $\frac{d}{c}$, we introduce $\log_2$ in the definition. We do not use the sigmoid function here because its resolution is not good, for example $sigmoid(\frac{8}{1}) = 0.9997$ and $sigmoid(\frac{8}{2}) = 0.9820$. Also note that a graph cluster will be more cohesive if it has more unimodal attributes.

A cohesive graph cluster is defined as follows:

Definition 7 *A **cohesive graph cluster** is a subgraph that has densely connected edges and has as many homogeneous (unimodal) attributes as possible. The density of edges is measured by the normalized cut, and the homogeneity of attributes is measured by the unimodality compactness.*

To detect cohesive graph clusters, our objective function integrates the *normalized cut* and *unimodality compactness* in one framework which is given as follows:

$$\mathrm{UNCut}(\mathcal{S}) = (1-\omega)\cdot \mathrm{NCut}(\mathcal{S}) + \omega \cdot \mathrm{UC}(\mathcal{S}) \tag{5.2}$$

where $\omega (0 \leq \omega \leq 1)$ is a weight parameter to adjust the importance between the *unimodality compactness* value and the *normalized cut* value of a graph cluster.

As said above, we can first eigen-decompose $\mathbf{L}$ to get some eigenvectors. Then, for each eigenvector, we apply 2-means (k-means with the input cluster number 2) to bisect the graph into two clusters and compute our objective function (Equation (5.2)). Finally, we select k eigenvectors associated with the k smallest *unimodal normalized cut* values. However, the time complexity to eigen-decompose $\mathbf{L}$ is $\mathcal{O}(n^3)$ which is impractical for large-scale attributed graphs. Instead, in this work, we use the power iteration method [37] to compute a number, say $10 \cdot k$, of pseudo-eigenvectors (approximate eigenvectors) and then choose k pseudo-eigenvectors associated with the k smallest *unimodal normalized cut* values.

Note that in Chapter 2, Equation (2.5) indicates that $\mathbf{v}^0$ can be denoted by $c_1\mathbf{u}_1 + c_2\mathbf{u}_2 +$

$\cdots + c_n\mathbf{u}_n$ which is a linear combination of all the original eigenvectors. By generating different starting vectors, we can get diverse linear combinations. If we let the power iteration method run enough time, it will converge to the dominant eigenvector $\mathbf{u}_1$ which is constant and of little use in clustering. We define the velocity at t to be the vector $\boldsymbol{\delta}^t = \mathbf{v}^t - \mathbf{v}^{t-1}$ and define the acceleration at t to be the vector $\boldsymbol{\epsilon}^t = \boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}$ and stop the power iteration when $\|\boldsymbol{\epsilon}^t\|_{max}$ is below a threshold $\hat{\epsilon}$.

The Algorithm 4 gives the pseudo-code to find k clusters with the smallest k *unimodal normalized cut* values.

Algorithm 4: UNCut

Input: Adjacency matrix $\mathbf{A}$, data matrix $\mathbf{F}$ and the cluster number k
Output: Cluster indicator $\mathbf{c}$

1 $\omega \leftarrow 0.5$, $\hat{\epsilon} \leftarrow 0.001$;
2 compute the random walk transition matrix $\mathbf{P}$;
3 $iter \leftarrow 100, K \leftarrow 10 \cdot k$;
4 **for** $i \leftarrow 1$ **to** K **do**
5 $\quad t \leftarrow 0, \mathbf{v}_i^0 \leftarrow$ **randn** $(1, n)$; /* $\mathbf{v}_i \in \mathbb{R}^{1\times n}$ */
$\quad$ /* power iteration */
6 $\quad$ **repeat**
7 $\quad\quad \mathbf{v}_i^{t+1} \leftarrow \frac{\mathbf{P}\mathbf{v}_i^t}{\|\mathbf{P}\mathbf{v}_i^t\|_1}$;
8 $\quad\quad \boldsymbol{\delta}^{t+1} \leftarrow |\mathbf{v}_i^{t+1} - \mathbf{v}_i^t|$;
9 $\quad\quad t \leftarrow t + 1$;
10 $\quad$ **until** $\|\boldsymbol{\delta}_i^{t+1} - \boldsymbol{\delta}_i^t\|_{\max} \leq \hat{\epsilon}$ **or** $t \geq iter$;
11 $\quad \mathcal{S}_i \leftarrow$**2-means** $(\mathbf{v}_i^t)$;
12 $\quad \mathrm{UNCut}(\mathcal{S}_i) \leftarrow (1-\omega)\cdot \mathrm{NCut}(\mathcal{S}_i) + \omega \cdot \mathrm{UC}(\mathcal{S}_i)$;
13 select k pseudo-eigenvectors associated with the k smallest *unimodal normalized cut* values;
14 use k-means on the selected k pseudo-eigenvectors to get the cluster indicator $\mathbf{c}$;
15 **return** $\mathbf{c}$;

Complexity analysis. Lines 5–10 in the Algorithm 4 use the power iteration method to compute one pseudo-eigenvector, whose time complexity is $\mathcal{O}(m)$ [39], where m is the number of graph edges. Line 11 uses 2-means on each pseudo-eigenvector, whose time complexity is $\mathcal{O}(n)$. At line 12, we compute the *unimodal normalized cut* which is dominated by the complexity of computing the *unimodality compactness* of clusters. We first

need to sort each attribute before computing the dip test, which costs $\mathcal{O}(n \cdot \log(n))$. The computation of dip test on each attribute costs $\mathcal{O}(n)$ [51]. Thus, the time complexity of lines 4–12 is $\mathcal{O}((m + n \cdot \log(n) \cdot d) \cdot k)$. Line 13 uses k-means on the selected k pseudo-eigenvector, whose time complexity is $\mathcal{O}(n \cdot k^2)$. The total time complexity of Algorithm 4 is $\mathcal{O}(m \cdot k + n \cdot \log(n) \cdot d \cdot k + n \cdot k^2)$, which is superlinear in the number of vertices n, linear in the numbers of edges m and attributes d, and quadratic in the number of clusters k.

5.3 Experimental Evaluation

In this section, we compare our method UNCut with the state-of-the-art methods from the attributed graph clustering field. As pointed out in [85], the comparison with the overlapping clustering approaches [40, 84] would always be biased to one of the paradigms due to their completely different objective from those of paititioning clustering approaches. Thus, following [85] we only compare UNCut with the partitioning clustering methods SA-cluster [107], SSCG [85] and NNM [70]. We use the synthetic and real-world data to evaluate the clustering performance. All the experiments are run on the same machine with an Intel Core Quad i7-3770 with 3.4 GHz and 32 GB RAM. We set $\omega = 0.5$ for our method UNCut on all the synthetic and real-world data. The parameters for the competitors are set according to their original papers. For all data, we feed all methods with the same cluster number. For the evaluation of clustering on synthetic data, we use the Normalized Mutual Information (NMI), Purity, and Adjusted Rand Index (ARI) [62] as clustering quality measures. The higher these clustering measures are, the better the clustering is. Because the real-world data does not have the ground truth, it is difficult to perform a quality assessment. We use the *normalized cut* and our *unimodadility compactness* to evaluate the clustering performance. The code and all the synthetic and real-world data are publicly available at the website[1].

[1] https://www.dropbox.com/sh/xz2ndx65jai6num/AAC9RJ5PqQoYoxreItW83PrLa?dl=0

5.3.1 Synthetic Data

Cluster Quality

We generate synthetic graphs with varying number of vertices n and attributes d. For the case of varying n, we fix the attribute dimension $d = 20$. For the case of varying d, we fix the number of vertices $n = 2000$. All the graphs are generated based on a benchmark graph generator [8], which makes the degree and cluster size follow power law distributions that reflect the real properties of vertices and clusters found in real networks. To add vertex attributes, for each graph cluster, we choose 20% attributes as relevant attributes and generate their values according to a Gaussian distribution with mean value of each attribute randomly sampled from the range $[0, 100]$ and variance value of each attribute randomly sampled from the range $(0, 0.1)$. To render the other attributes of clusters irrelevant to the edge structure, we randomly permute the cluster labels and generate each cluster's irrelevant attribute values according to a Gaussian distribution with mean 0 and variance 1. For each experiment, we test all the methods on the generated ten attributed graphs differing in the edge structure and attribute values.

Figure 5.2(a), Figure 5.3(a), and Figure 5.4(a) show the performance of all the methods when varying the number of attributes, where we can see that UNCut is superior to its competitors. Compared with SA-cluster and NNM, both UNCut and SSCG exceed them with large margins. UNCut and SSCG are subspace clustering methods, while SA-cluster and NNM are full-space clustering methods which are easily deceived by "*the curse of dimensionality*". Figure 5.2(b), Figure 5.3(b), and Figure 5.4(b) present the performance of all the methods when varying the number of graph vertices. SSCG has a comparable performance when the number of vertices is 1000. However, our method UNCut beats SSCG when increasing the vertex number. Note that subspace clustering methods UNCut and SSCG are still better than the full-space clustering methods SA-cluster and NNM.

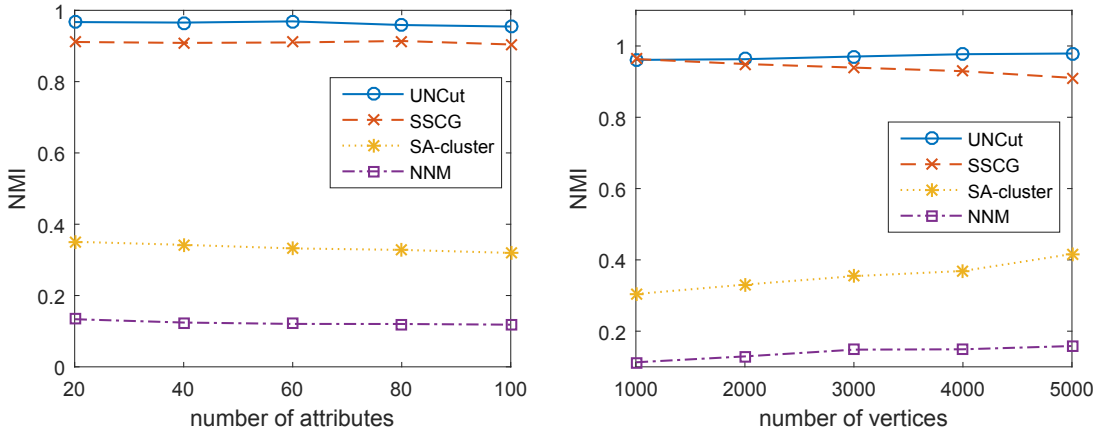


Figure 5.2: Quality evaluation (NMI).

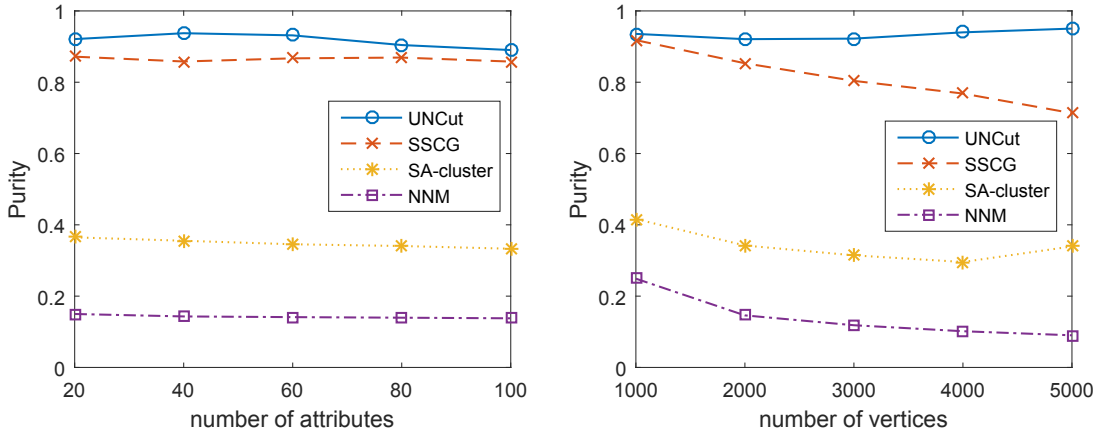


Figure 5.3: Quality evaluation (Purity).

Scalability

We still use the above attributed graph generation method to generate synthetic graphs for the evaluation of the runtime of each method. Figure 5.5(a) shows the runtime when varying the number of attributes (the number of vertices is fixed to 2000). We can see that NNM is the fastest method and SSCG is the slowest method. SSCG needs to update its subspace dependent weight matrix in every iteration, which is very time consuming. Figure 5.5(b) demonstrates the runtime when varying the number of vertices (the number of attributes is fixed to 20). NNM still performs the best and SSCG performs the worst. Our method UNCut is in the second place. Because UNCut is linear in the number of

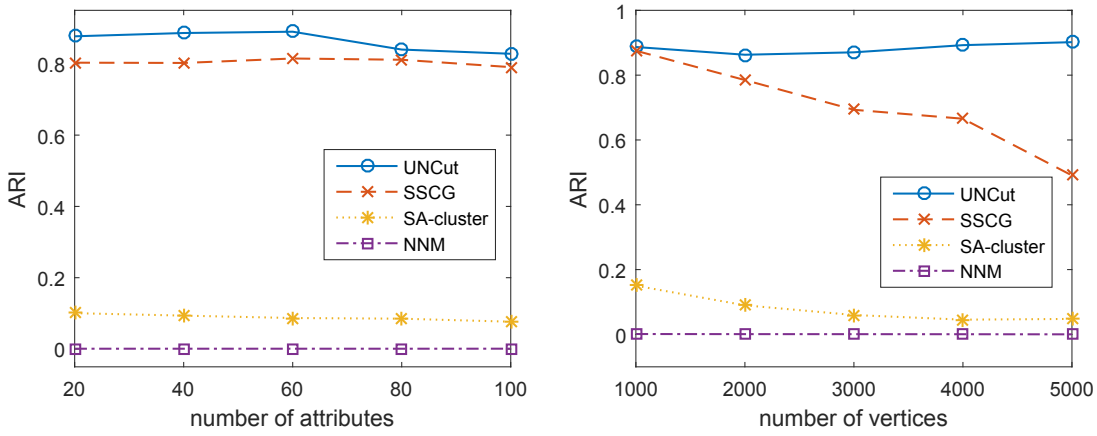


Figure 5.4: Quality evaluation (ARI).

edges, a drop in the runtime when increasing the number of vertices from 4000 to 6000 can be interpreted as caused by the drop in the number of edges.

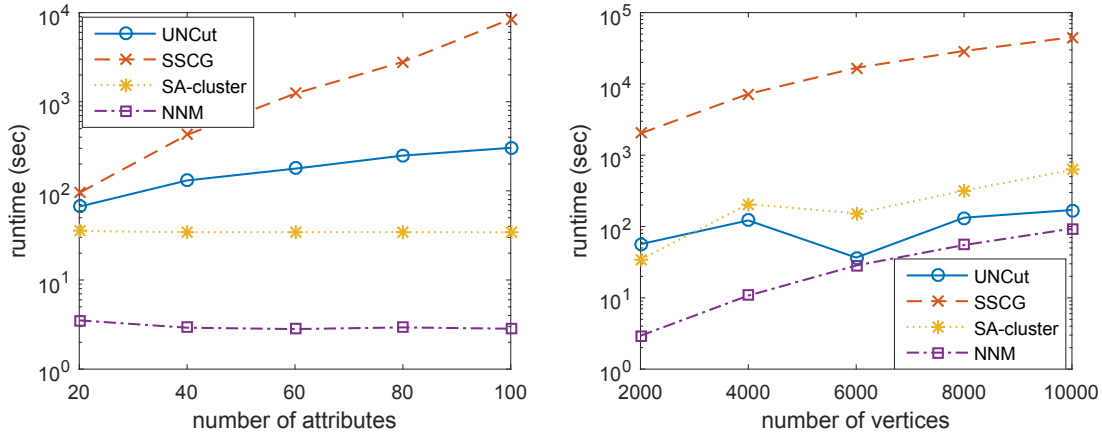
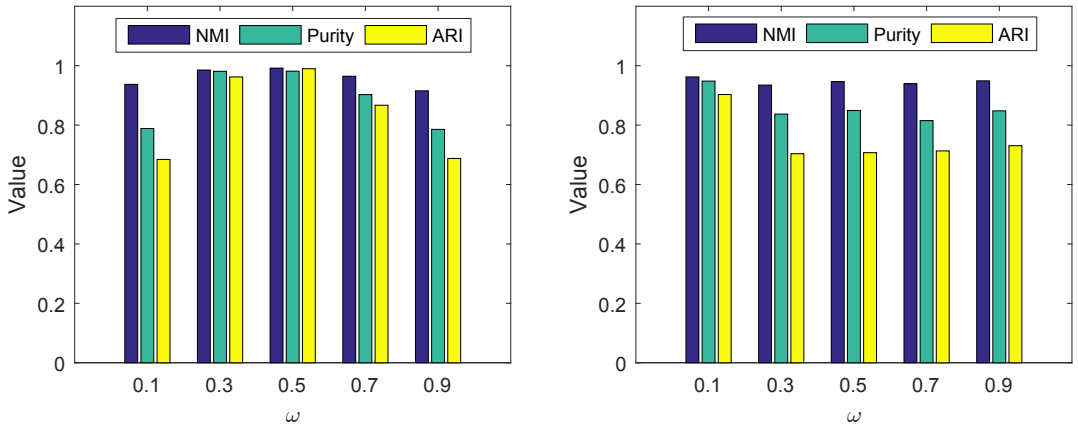


Figure 5.5: Runtime evaluation.

Stability

In this section, we study how the parameter ω affects the clustering performance. Figure 5.6(a) gives the clustering performance of UNCut on the synthetic graph with 100 attributes and 2000 vertices when varying ω. And Figure 5.6(b) gives the clustering performance of UNCut on the synthetic graph with 20 attributes and 1000 vertices when varying ω. From

Figure 5.6(a), we can see that UNCut achieves the best result when the value of ω is 0.5. From Figure 5.6(b), we can see that UNCut achieves the best result when the value of ω is 0.1. For different graphs with different edge structure and attribute values, the values of the best ω are different.



(a) the graph with 100 attributes and 2000 vertices

(b) the graph with 20 attributes and 1000 vertices

Figure 5.6: Varying the parameter ω.

5.3.2 Real-world Data

In this section, we evaluate UNCut and its competitors on six real-world datasets Disney [33], DFB [85], ARXIV [85], PolBlogs [14], 4area [14] and Patents [85]. The statistics of the real-world data are given in Table 5.1. The *normalized cut* and *unimodality compactness* values achieved by each algorithm are listed in Table 5.2.

We can see from Table 5.2 that our method UNCut achieves the best results on the datasets Disney, DFB and ARXIV in terms of both the *normalized cut* and *unimodality compactness* values. On the dataset PolBlogs, SSCG achieves the best *normalized cut* value. However, the *unimodality compactness* value achieved by UNCut is much lower than those of its competitors. On the dataset 4area, SA-cluster achieves the best results. Although SSCG is a method detecting subspace clusters, it is defeated by SA-cluster on the datasets Disney, ARXIV and 4area in terms of the *unimodality compactness* values.

For the dataset PATENTS, all the competitors fail due to their much consumption of the memory. Our method UNCut is scalable for large-scale networks. To examine whether UNCut can achieve differing results to those of its competitors, as did in [85], we compute NMI between the results of UNCut and its competitors. A low NMI value indicates that UNCut is able to detect novel cluster insights, without implying that the results of the competitors are worse or meaningless. The NMI values are given in Table 5.3. From Table 5.3, we can see that UNCut can find novel cluster insights different from the competitors, especially on the 4AREA dataset. The NMI values between the results of UNCut and its competitors are near 0, which means totally different insights. For case studies, we interprete the detected clusters of all the methods on the datasets DISNEY and POLBLOGS.

Table 5.1: Statistics of datasets.

Datasets	#vertices	#edges	#attributes	#clusters
DISNEY	124	333	28	9
DFB	100	1,106	5	14
ARXIV	856	2,660	30	19
POLBLOGS	358	1,288	44,839	10
4AREA	26,144	108,550	4	50
PATENTS	100,000	188,631	5	150

Disney. DISNEY is a subgraph of the Amazon copurchase network. Each movie (vertex) is described by 28 attributes, such as "average vote", "product group", "price" and etc. The green cluster has 14 movies, which is rated as PG (Parental Guidance Suggested) and attributed as "Action & Adventure". It contains movies such as "Spy Kids", "Inspector Gadget" and "Mighty Joe Young". The purple cluster includes 9 read-along movies, which is rated as G (General Audience) and attributed as "Kids & Family". It has movies such as "Beauty and the Beast", "Lilo and Stitch", "Toy Story 2", "The Little Mermaid", and "Monsters, Inc.". The purple cluster has three multimodal attributes "review frequency", "rating of review with most votes", and "rating of most helpful rating". In other words, the movies in the purple cluster are similar in the subspace spanned by the

Table 5.2: Normalized cut and unimodality compactness values. (N/A means the results are not available due to the runout of memory.)

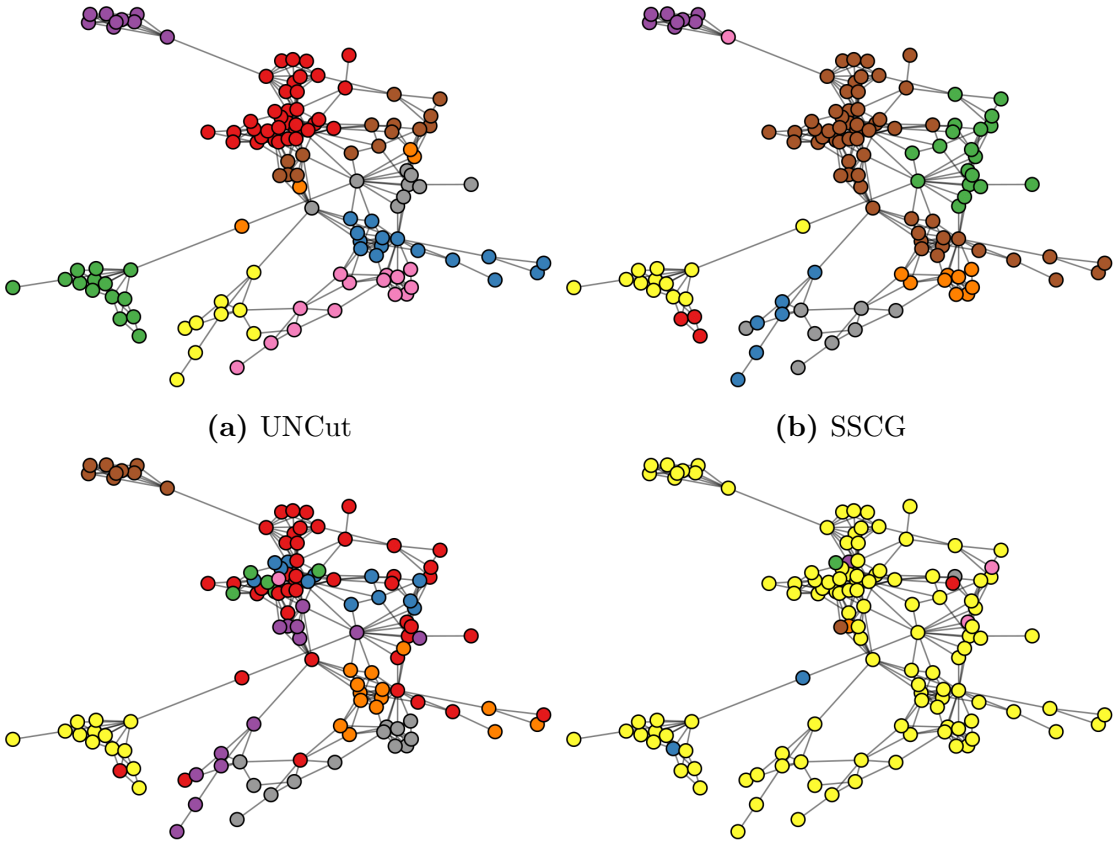
Datasets	Normalized Cut			
	UNCut	SSCG	SA-cluster	NNM
Disney	**2.702**	2.646	3.959	8.058
DFB	**10.596**	13.161	13.116	13.026
ARXIV	**1.889**	17.940	10.606	18.017
PolBlogs	7.429	**5.436**	8.181	9.071
4area	30.120	41.314	**10.813**	N/A
Patents	**31.980**	N/A	N/A	N/A
Datasets	Unimodality Compatness			
	UNCut	SSCG	SA-cluster	NNM
Disney	**1.807**	20.459	10.709	77.266
DFB	**11.541**	20.507	60.692	43.082
ARXIV	**26.621**	176.911	45.940	148.378
PolBlogs	**1.568**	155.377	217.068	124.404
4area	184.000	152.83	**37.075**	N/A
Patents	**415.941**	N/A	N/A	N/A

Table 5.3: NMI between the results of UNCut and its competitors. (N/A means the results are not available due to the runout of memory.)

Datasets	UNCut	SSCG	SA-cluster	NNM
Disney	1.000	0.724	0.597	0.164
DFB	1.000	0.298	0.246	0.272
ARXIV	1.000	0.096	0.387	0.131
PolBlogs	1.000	0.488	0.297	0.060
4area	1.000	0.027	0.043	N/A
Patents	1.000	N/A	N/A	N/A

other attributes. The clusters found by our method UNCut are subspace clusters which are cohesive on as many attributes as possible. SSCG splits our purple cluster into two clusters and our green clusters into two clusters. SA-cluster splits our green cluster into two clusters. NNM groups the most of the movies together (yellow cluster), which leads to the highest *unimodality compactness* value as shown in Table 5.2.

PolBlogs. PolBlogs is the citation network among a collection of online blogs that discuss political issues. Attributes are the keywords in their text. If a keyword appears



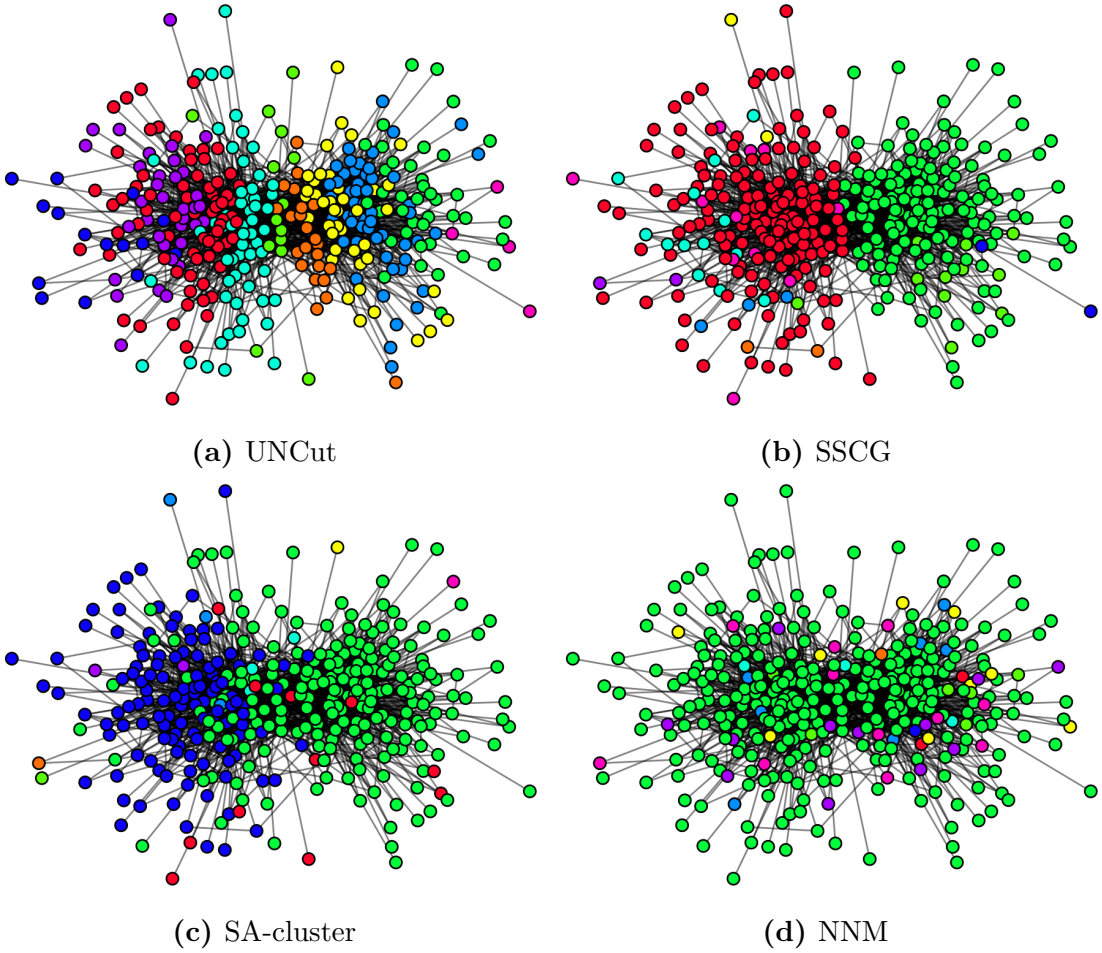
(a) UNCut **(b)** SSCG

(c) SA-cluster **(d)** NNM

Figure 5.7: Clustering results on DISNEY. (Plotted by the Python toolbox *Networkx*)

in the text, the attribute value is set to 1, otherwise 0. Thus, each attribute only has binary values. The red cluster contains 70 blogs. The top five frequent keywords of the red cluster are "London", "Iraq", "government", "work", and "American". The orange cluster contains 23 blogs. The top six frequent keywords of the orange cluster are "act", "bush", "conservative", "court", "justice", and "law". The blue cluster includes 53 blogs. The top eight frequent keywords of the blue cluster are "people", "post", "right", "political", "issue", "media", "president", and "public". For SSCG and SA-cluster, the sizes of the two main clusters are very big, i.e., the red and green clusters found by SSCG totally have 312 vertices and the blue and green clusters found by SA-cluster totally have 335 vertices. For NNM, the most of the blogs belong to the green cluster which has 306 vertices. Thus, the sizes of the most clusters detected by the competitors are small, which leads to the

high probability of having multimodal attributes as proved by the much higher *unimodality compactness* values in Table 5.2.



(a) UNCut **(b)** SSCG

(c) SA-cluster **(d)** NNM

Figure 5.8: Clustering results on PolBlogs. (Plotted by the Python toolbox *Networkx*)

5.4 Related Work and Discussion

Compared with massive works on the plain graph clustering, there are relatively less work on the attributed graph clustering. Differing from the plain graph clustering that groups vertices only considering the edge structure, the attributed graph clustering achieves grouping vertices with dense edge connectivity and homogeneous attribute values into clusters. NNM [70] first develops a measure called *normalized network modularity* and then proposes

a spectral method that combines the costs of clustering numerical vectors and *normalized network modularity* into an eigen-decomposition problem. BAGC (Bayesian Attributed Graph Clustering) [111] develops a Bayesian probabilistic model for attributed graphs, which captures both structure and attribute aspects of a graph. Clustering is accomplished by an efficient variational inference method. BAGC is only capable of categorical attributes. PICS [60] groups vertices into disjoint clusters satisfying that vertices in the same cluster exhibit similar connectivity and feature coherence. It exploits the Minimum Description Length (MDL) principle to automatically select the parameters such as the cluster number. PICS is only capable of graphs with binary feature vectors. SA-cluster [107] designs a unified neighborhood random walk distance to measure the vertex similarity on an augmented graph. It uses k-medoids to partition the graph into clusters with cohesive intra-cluster structures and homogeneous attribute values.

However, the above methods which take all attributes into consideration may fail because there may be attributes irrelevant to the edge structure. Now more researches focus on detecting subspace clusters to which only subsets of attributes are assigned. CoPaM [40] exploits various pruning strategies to efficiently find maximal cohesive patterns in the subspace of feature vectors. GAMer [84] determines sets of vertices which have high similarity in the subsets of attributes and are densely connected as well by combining the paradigms of subspace clustering and dense subgraph mining together. The twofold clusters are optimized by exploiting various pruning strategies considering the density, size and number of relevant attributes. CoPaM and GAMer exploit the notion of quasi-cliques which poses strong restrictions on the feature range and diameter of the clusters. CoPaM generates a huge number of redundant overlapping clusters. To reduce the redundancy, GAMer introduces additional parameters which are difficult to set for the real-world data. Differing from CoPaM and GAMer, our partitioning method UNCut does not suffer from redundancy. SSCG [85] presents a solution for an objective function called *Minimum Normalized Subspace Cut*, which integrates spectral clustering to the problem of subspace clustering for attributed graphs. It detects an individual set of relevant features for each cluster. Our method UNCut only considers the relevent attributes to the edge strcuture, i.e., irrelevent

attributes are excluded from the computation of the *unimodality compactness*. In other words, UNCut detects subspace clusters with as many unimodal attributes as possible.

Recently, a new research trend is to detect community outliers in attributed graphs. MAM (maximization of attribute-aware modularity) [77] develops attribute compactness to quantify the relevance of the attributes, which is then combined with the conventional modularity for the robust graph clustering with respect to irrelevant attributes and outliers. ConSub (congruent subspace selection) [78] defines a measure to assess the degree of congruence between a set of attributes and the edge structure, which is then used for the statistical selection of the congruent subspaces. FocusCO [14] defines a new graph clustering problem which incorporates the user's preference into graph mining. Given a set of examplar vertices of user's interest, FocusCO infers user's preference by applying a distance metric learning method. New vertices are carefully added to the set of examplar vertices by checking the weighted conductance. Differing from the conventional attributed graph clustering methods, FocusCO performs a local clustering of interest to the user rather than the global partitioning of the entire graph.

5.5 Summary

We have proposed UNCut to detect cohesive clusters in attributed graphs. To this end, we develop a measure called *unimodality compactness*, which is then combined with the *normalized cut* to elegantly search for cohesive clusters. Since the complexity of the eigendecomposition of the graph Laplacian matrix is high, we adopt the power iteration method to approximately compute the eigenvectors. We have tested our method UNCut on various synthetic and real-world data, which verifies that UNCut achieves better results than its competitors.

Chapter 6

Semi-Supervised Learning in Graphs

In Chapter 5, we proposed UNCut to find cohesive clusters in attributed graphs. Generally, we know the labels of some vertices in graphs. However, only very few vertices have labels compared to large amounts of unlabeled vertices. For example, in social networks, not every user provides his/her profile information such as the personal interests which are relevant for targeted advertising. Can we leverage the limited user information and friendship graph wisely to infer the labels of unlabeled users?

In this chapter, we propose a semi-supervised learning framework called weighted-vote Geometric Neighbor classifier (wvGN) to infer the labels of unlabeled vertices in sparsely labeled graphs. wvGN exploits random walks to explore not only local but also global neighborhood information of a vertex. Then the label of the vertex is determined by the accumulated local and global neighborhood information. Specifically, wvGN optimizes a proposed objective function by a search strategy which is based on the gradient and coordinate descent methods. The search strategy iteratively conducts a coarse search and a fine search to escape from local optima. Extensive experiments on various synthetic and real-world data verify the effectiveness of wvGN compared to state-of-the-art approaches.

Parts of the materials presented in this chapter have been published in [98], where Wei Ye proposed the main idea, conducted the experimental evaluation, and wrote the most parts of the paper; Linfei Zhou and Dominik Mautz helped with the derivation of the

objective function and paper writing; Christian Böhm and Claudia Plant supervised the project and wrote some parts of the paper.

"*Wei Ye, Linfei Zhou, Dominik Mautz, Claudia Plant, Christian Böhm. Learning from Labeled and Unlabeled Vertices in Networks. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2017.*"

6.1 Motivation

The problem of learning from labeled and unlabeled data, literarily attributed to semi-supervised learning, has aroused considerable interests in recent years. Generally, labeled data is scarce while unlabeled data is abundant. Labeling data can be very tedious, time-consuming and expensive because it needs the efforts of skilled human annotators. How to effectively make use of unlabeled data to improve learning performance is of great practical significance. Recently, various semi-supervised learning methods have been proposed, such as TSVM [93,96], LapSVM [67] and LGC [27]. Both LapSVM and LGC make assumption of local and global label consistency in graph and do not learn directly from data. Although LapSVM and LGC can be directly applied on sparsely labeled graphs, their performance is not satisfying when their assumption is not met.

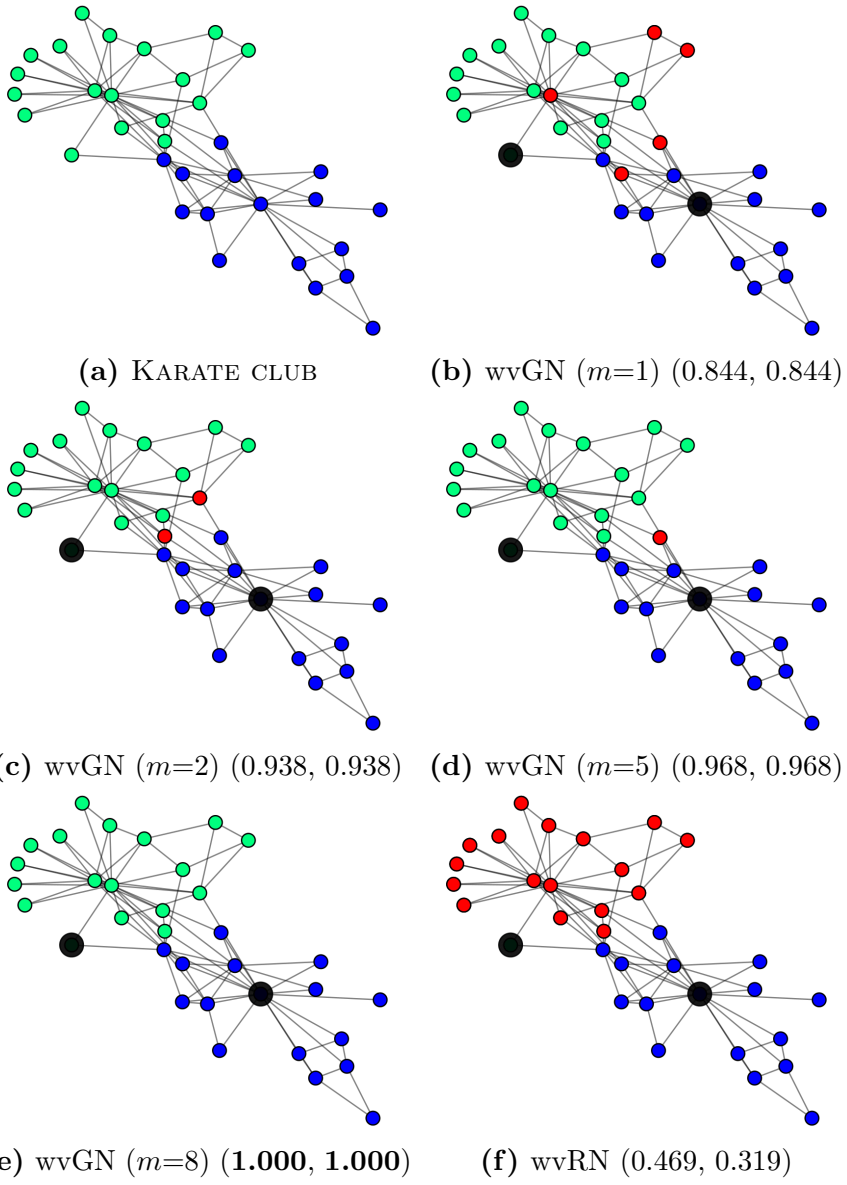
Differing from conventional data which is assumed to be i.i.d (independent and identically distributed), graph data, extracted from social media, bibliographic databases, etc., is interdependent. Vertices connected to each other are likely to have the same labels according to the principle of homophily [68]. Relational learning [61, 90] has been proposed to capture the correlations between connected vertices. It makes a first-order Markov assumption to infer labels, i.e., the label of a vertex is determined by those of its direct neighbors in the graph. In the prediction process, collective inference is used to find an equilibrium state such that the inconsistency between neighboring vertex is minimized. Relational learning methods do not perform well when unlabeled vertices have too few labeled neighbors to support learning and/or inference [11]. The question is how to infer the labels of unlabeled vertices when they have too few labeled neighbors? Our idea is to

use random walk to capture both short and long distance relationships in graphs. To be specific, if a vertex has too few labeled neighbors, we can exploit its one- to m-hop (See Section 6.2) neighbors to support the learning process.

To motivate the problem, we use a real graph called KARATE CLUB [102] shown in Figure 6.1(a). The graph records 78 pairwise interactions (links) between 34 members of a karate club who interacted outside the club. The graph is partitioned into two communities. The blue community contains 16 members and the green community contains 18 members. For the test, the labels of the two vertices in black are given and the labels of the other vertices are unknown to all the methods. Figure 6.1(b) shows the result of our method wvGN only using the geometric one-hop neighborhood information (the definition is given in Section 6.2.1). Five members are misclassified, which leads to a Micro-F1 score of 0.844; Figure 6.1(c) gives the result of wvGN using the geometric one- and two-hop neighborhood information. Two members are misclassified, which leads to a Micro-F1 score of 0.938; Figure 6.1(d) shows the result of wvGN using the geometric one- to five-hop neighborhood information. One member is misclassified, which leads to a Micro-F1 score of 0.968. When using the geometric one- to eight-hop neighborhood information, wvGN correctly classifies all the members (Figure 6.1(e)). With the accumulation of both local and global neighborhood information, the learning performance of wvGN is strengthened. Figure 6.1(f) demonstrates the result of wvRN (weighted-vote Relational Neighbor classifier) [89] (the results of LapSVM and LGC are the same and not shown here to reduce the clutter). We can see that wvRN fails in such an occasion when the graph is sparsely labeled.

To achieve better classification results, our method wvGN exploits not only local but also global neighborhood information of vertices in a graph. Specifically, the label of a vertex is jointly determined by its geometric one-hop to m-hop ($m = 1, 2, \ldots$) neighbors in the graph. The geometric m-hop neighbors of a vertex are those vertices which are m-hops away by a random walker. The contributions are as follows:

- We use random walks with arbitrary m hops to accumulate local and global neighbor-



(a) Karate club **(b)** wvGN (m=1) (0.844, 0.844)

(c) wvGN (m=2) (0.938, 0.938) **(d)** wvGN (m=5) (0.968, 0.968)

(e) wvGN (m=8) (**1.000**, **1.000**) **(f)** wvRN (0.469, 0.319)

Figure 6.1: Classification results on Karate Club. The two labeled vertices are in black and the misclassified vertices are in red. m is the number of hops of a random walker. Subcaption: Method (Micro-F1, Macro-F1).

hood information for better classification. m is implicitly determined by the power method.

- We formulate the semi-supervised learning in graphs as an optimization problem and propose an objective function based on the L2-loss SVM.

- We propose a search strategy called gradient and coordinate descent (GCD) to optimize the objective function. GCD is a combination of the gradient descent (GD) and coordinate descent (CD) methods. GD is used for a coarse search and CD is used for a fine search. GCD does not easily get trapped in local optima.
- We show the effectiveness of our method wvGN by carrying out exhaustive comparative studies with the state-of-the-art methods from various related domains.

6.2 Weighted-vote Geometric Neighbor Classifier

6.2.1 The Model

We first give the problem formulation of **Semi-Supervised Learning in Networks** as follows:

Definition 8 *Given a graph* $\mathsf{G} = (\mathcal{V}, \mathcal{E})$ *with vertices* $\{v_1, \ldots, v_l\}$ *labeled as* $\mathbf{y}_l = [y_1, \ldots, y_l], l \ll n, y_i (1 \leqslant i \leqslant l) \in \{+1, -1\}$ *and vertices* $\{v_{l+1}, \ldots, v_n\}$ *unlabeled. The goal is to learn a classifier to infer the labels* $\hat{\mathbf{y}}_u = [\hat{y}_{l+1}, \ldots, \hat{y}_n]$ *of the unlabeled vertices.*

To classify the vertices in a graph, in this work, we first transform vertices from graph space to vector space. For a graph, we define its geometric one-hop neighborhood as follows:

Definition 9 ***Geometric One-hop Neighborhood.*** *The geometric one-hop neighbors of a vertex* v_i *is defined as a set* $\mathcal{N}_i^1$ *which contains those vertex* v_j *which can be reached by a random walker from* v_i *in one step. The geometric one-hop neighborhood is defined as a set* $\mathcal{N}^1 = \bigcup_{i=1}^{l} \mathcal{N}_i^1$.

We denote a vertex v_q in the geometric one-hop neighborhood $\mathcal{N}^1$ by $\mathbf{p}_{q\cdot}$ which is the q-th row of the transition matrix $\mathbf{P}$. The class indicator score of the vertex v_q is defined as follows:

$$f(\mathbf{p}_{q\cdot}) = \mathbf{p}_{q\cdot} \cdot \mathbf{w}^\intercal + b \tag{6.1}$$

where the weight vector $\mathbf{w}$ and bias term b are the parameters to be learned.

Then the label of the vertex v_q is infered by the following formula,

$$y_q = sign(f(\mathbf{p}_{q\cdot})) = sign(\mathbf{p}_{q\cdot} \cdot \mathbf{w}^\intercal + b) \tag{6.2}$$

If we want to seek the separating hyperplane with the largest margin for the positive and negative samples, Equation (6.1) becomes the problem of linear support vector machines (SVM). However, Equation (6.1) does not take the neighborhood relationship into consideration to classify vertices in graphs, which would lead to the deterioration of the classifier. The theory of homophily [68] tells us that vertices connected to each other tend to have the same labels. For example, friends connected in a social network are likely to have similar interests; papers connected in a citation network are likely to have similar topics. Thus to classify the vertex v_q, we need to consider its neighborhood information. A simple relational neighbor classifier (RN) [89] estimates the class-membership probability of the vertex v_q by $P(\text{class}|v_q) = \frac{1}{Z}\sum_{\text{label}(v_i)=\text{class}|i\in\mathcal{N}_q^1} a_{i,q}$, where $Z = \sum_{i\in\mathcal{N}_q^1} a_{i,q}$. However, in some cases, it is still hard to determine the label of a vertex according to the theory of homophily. For example in Figure 6.2, the vertex v_5 is connected to three vertices in the green class and also connected to three vertices in the yellow class. Since $P(\text{yellow}|v_5) = P(\text{green}|v_5) = 0.5$, what is the label of the vertex v_5?

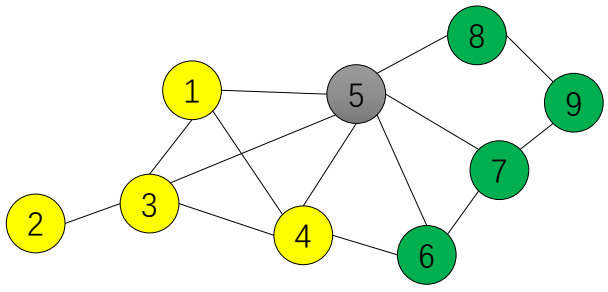


Figure 6.2: An example graph with the vertex v_5 unlabeled.

In this work, we use a weighted-vote strategy to integrate the neighborhood information

of the vertex v_q with Equation (6.1) as follows:

$$\begin{aligned} & f(\mathbf{p}_{q\cdot}) = \mathbf{p}_{q\cdot} \cdot \mathbf{w}^\intercal + b > 0, \;\; y_q = 1 \\ & \text{if } \frac{\sum_{i \in \mathcal{N}_q^1} a_{iq} f(\mathbf{p}_{i\cdot})}{\sum_{i \in \mathcal{N}_q^1} a_{iq}} \geq +1 \end{aligned} \tag{6.3}$$

Similarly, if the weighted indicator score is less than or equal to -1, we have:

$$\begin{aligned} & f(\mathbf{p}_{q\cdot}) = \mathbf{p}_{q\cdot} \cdot \mathbf{w}^\intercal + b < 0, \;\; y_q = -1 \\ & \text{if } \frac{\sum_{i \in \mathcal{N}_q^1} a_{iq} f(\mathbf{p}_{i\cdot})}{\sum_{i \in \mathcal{N}_q^1} a_{iq}} \leq -1 \end{aligned} \tag{6.4}$$

Inequalities (6.3) and (6.4) can be combined into the following inequality:

$$\begin{aligned} & y_q \cdot \frac{\sum_{i \in \mathcal{N}_q^1} a_{iq} (\mathbf{p}_{i\cdot} \cdot \mathbf{w}^\intercal + b)}{\sum_{i \in \mathcal{N}_q^1} a_{iq}} \geq 1 \\ \Rightarrow & y_q \cdot \left(\frac{\sum_{i \in \mathcal{N}_q^1} a_{iq} \cdot \mathbf{p}_{i\cdot}}{d_{qq}} \cdot \mathbf{w}^\intercal + b \right) \geq 1 \\ \Rightarrow & y_q \cdot \left(\frac{\mathbf{a}_{q\cdot} \cdot \mathbf{P}}{d_{qq}} \cdot \mathbf{w}^\intercal + b \right) \geq 1 \\ \Rightarrow & y_q \cdot (\mathbf{p}_{q\cdot} \cdot \mathbf{P} \cdot \mathbf{w}^\intercal + b) \geq 1 \end{aligned} \tag{6.5}$$

The above inequality is just for the classification of the vertex v_q using its geometric one-hop neighborhood information. However, in real graphs, the vertex may be sparsely labeled. For example in Figure 6.3, how to infer the class label of the vertex v_5 when its directly connected neighbors are unlabeled? The relational neighbor classifier (RN) [89] iteratively classifies vertices using its previously inferred labels. However, if there is an error inference, the error will be amplified in the subsequent inference procedure.

Thus, we also need to integrate the information from a vertex's geometric m-hop ($m \geq 2$) neighborhood, whose definition is as follows:

Definition 10 ***Geometric*** m***-hop Neighborhood.*** *The geometric m-hop neighbors of a vertex v_i is defined as a set $\mathcal{N}_i^m$ which contains those vertex v_j which can be reached by a*

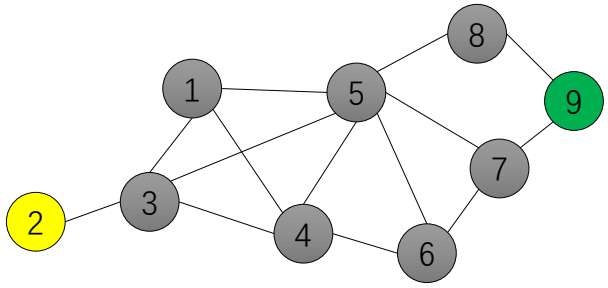


Figure 6.3: An example graph with sparsely labeled vertices.

random walker from v_i in m steps. The geometric m-hop neighborhood is defined as a set $\mathcal{N}^m = \bigcup_{i=1}^{l} \mathcal{N}_i^m$.

In the geometric m-hop neighborhood $\mathcal{N}_i^m$, v_q is denoted by $\mathbf{p}_{q\cdot}^m$ which is the q-th row of the transition matrix $\mathbf{P}^m$. If we replace $\mathbf{p}_{q\cdot}$ in the Inequality (6.5) with $\mathbf{p}_{q\cdot}^m$, we have:

$$\begin{aligned}
& y_q \cdot \frac{\sum_{i \in \mathcal{N}_q^m} a_{iq} (\mathbf{p}_{i\cdot}^m \cdot \mathbf{w}^\intercal + b)}{\sum_{i \in \mathcal{N}_q^m} a_{iq}} \geq 1 \\
\Rightarrow & y_q \cdot \left(\frac{\sum_{i \in \mathcal{N}_q^m} a_{iq} \cdot \mathbf{p}_{i\cdot}^m}{d_{qq}} \cdot \mathbf{w}^\intercal + b \right) \geq 1 \\
\Rightarrow & y_q \cdot \left(\frac{\mathbf{a}_{q\cdot} \cdot \mathbf{P}^m}{d_{qq}} \cdot \mathbf{w}^\intercal + b \right) \geq 1 \\
\Rightarrow & y_q \cdot (\mathbf{p}_{q\cdot} \cdot \mathbf{P}^m \cdot \mathbf{w}^\intercal + b) \geq 1 \\
\Rightarrow & y_q \cdot \left(\mathbf{p}_{q\cdot}^m \cdot \mathbf{P} \cdot \mathbf{w}^\intercal + b \right) \geq 1 \\
\Rightarrow & y_q \cdot \left(\mathbf{p}_{q\cdot}^{m+1} \cdot \mathbf{w}^\intercal + b \right) \geq 1
\end{aligned} \tag{6.6}$$

where $\mathbf{p}_{q\cdot}^{m+1}$ is the representation of v_q in the geometric $(m+1)$-hop neighborhood.

We can see from above that in geometric neighborhood $\mathcal{N}^1 \cup \cdots \mathcal{N}^m$, a vertex is always transformed to its next-hop neighborhood by the transition matrix $\mathbf{P}$. Similar to SVM [19,21], we introduce a positive slack variable ξ_q in the constraints and the Inequality (6.6) becomes:

$$\begin{aligned}
y_q \cdot \left(\mathbf{p}_{q\cdot}^{m+1} \cdot \mathbf{w}^\intercal + b \right) & \geq 1 - \xi_q \\
\xi_q \geq 0,\ 1 \leq q \leq l,\ m & \geq 1
\end{aligned} \tag{6.7}$$

Compared with the large amount of unlabeled data, the amount of labeled data is limited in the semi-supervised learning scenario. In addition, as we said before, it is hard to infer the label of a vertex if it has too few labeled neighbors. To alleviate these situations, we accumulate the information from every geometric neighborhood and represent the vertex v_q by $\mathbf{x}_q = \mathbf{p}_{q\cdot}^2 + \cdots + \mathbf{p}_{q\cdot}^{m+1} = \mathbf{p}_{q\cdot} \cdot (\mathbf{P} + \cdots + \mathbf{P}^m)$. The unconstrained SVM problem with L2 loss is formulated as follows:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w} \cdot \mathbf{w}^\intercal + \frac{\alpha}{l}\sum_{q=1}^{l} \max(1 - y_q \cdot f(\mathbf{x}_q), 0)^2 \tag{6.8}$$

Note that $\mathbf{x}_q$ can be considered as a random walker taking $m+1 (m \geq 1)$ hops from the vertex v_q and accumulating the neighborhood information at each hop. With higher values of m, $\mathbf{x}_q$ becomes more global. Meanwhile, $\mathbf{x}_q$ becomes identical, because the transition probability tends to converge towards the stationary distribution which is not useful in classification. To avoid the situation where all vertices have identical representation, we introduce a dampening factor at each hop such that higher hops have higher penalty and thus decay faster. Inspired from the heat kernel [35], the dampening factor for the m-th hop is defined as $\rho^m/m!$, where ρ is a non-negative parameter (considered as the temperature in the heat kernel).

In this work, we penalize $\mathbf{w}_i$ by $d_{ii}^{-\frac{1}{2}}$. For mathematical convenience, we extend each instance $\mathbf{x}_q$ as $[\mathbf{x}_q, 1]$ and $\mathbf{w}$ as $[\mathbf{w}, b]$. The objective function (6.8) now becomes:

$$\min_{\mathbf{w}} \quad F(\mathbf{w}) = \frac{\lambda}{2}(\mathbf{w} \odot \mathbf{d})(\mathbf{w} \odot \mathbf{d})^\intercal + \frac{\alpha}{l}\sum_{q=1}^{l} \max(1 - y_q \mathbf{x}_q \mathbf{w}^\intercal, 0)^2 \tag{6.9}$$

where $\odot$ means the Hadamard product, λ and α are regularization parameters, $\mathbf{x}_q = \left[\mathbf{p}_{q\cdot} \cdot \left(\frac{\rho^1}{1!}\mathbf{P} + \cdots + \frac{\rho^m}{m!}\mathbf{P}^m\right), 1\right]$, $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ and $\mathbf{d} = \left[diag(\mathbf{D}^{-\frac{1}{2}})^\intercal, 1\right]$.

6.2.2 Optimization

Stochastic gradient descent (SGD) algorithm is very successful in training large-scale SVM [92] on sparse data. However, we find it gets easily trapped in local optima because the representation of each vertex in vector space is not sparse. So do the GD and CD methods (See Section 6.3.1) because they are dependent on the starting point. In this work, we use a combination of GD and CD to optimize our objective function given in (6.9). Specifically, we first conduct a coarse search by GD owing to its fast convergence and then conduct a fine search by CD. The gradient of (6.9) with respect to $\mathbf{w}$ is:

$$F'(\mathbf{w}) = \lambda \mathbf{w} \odot \mathbf{d} - \frac{2\alpha}{l} \sum_{j \in \mathbb{I}(\mathbf{w})} y_j \mathbf{x}_j b_j(\mathbf{w}) \tag{6.10}$$

where $b_j(\mathbf{w}) = 1 - y_j \mathbf{x}_j \mathbf{w}^\intercal$ and $\mathbb{I}(\mathbf{w}) = \{j | b_j(\mathbf{w}) > 0\}$.

We iteratively update $\mathbf{w}$ as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t F'(\mathbf{w}^t) \tag{6.11}$$

where η_t is the learning rate at the t-th iteration and is chosen from $\{1, \beta, \beta^2, \dots\}$ by a line search.

We can see that the learning rate η_t for each element in $\mathbf{w}$ is the same at each iteration, which leads to GD's easily getting trapped in local optima. To let GD escape from local optima, we conduct a fine search by CD. The CD method has been successfully applied for solving large-scale L2-loss SVM [57]. The method starts from an initial vector $\mathbf{w}^0$ (the final output of GD in this work) and iteratively generates a sequence $\{\mathbf{w}^t\}$ $(t = 0, 1, 2, \dots)$. At each iteration, $\mathbf{w}^{t+1}$ is produced by sequentially updating each entry of $\mathbf{w}^t$ with other entries fixed. The process produces a sequence of vectors $\mathbf{w}^{t,i}$ $(i = 1, \dots, n+1)$, such that $\mathbf{w}^{t,0} = \mathbf{w}^t$, $\mathbf{w}^{t,n+1} = \mathbf{w}^{t+1}$ and

$$\mathbf{w}^{t,i} = \left[w_1^{t+1}, \dots, w_i^{t+1}, w_{i+1}^t, \dots, w_{n+1}^t\right]$$

Updating $\mathbf{w}^{t,i}$ to $\mathbf{w}^{t,i+1}$ becomes the following one-variable sub-problem:

$$\begin{aligned} &\min_{z} \quad F_i(w_1^{t+1}, \ldots, w_i^{t+1}, w_{i+1}^t + z, w_{i+2}^t \ldots, w_{n+1}^t) \\ \equiv &\min_{z} \quad F_i(\mathbf{w}^{t,i} + z\mathbf{e}_i) \\ = &\min_{z} \quad \frac{\lambda}{2}\left((\mathbf{w}^{t,i} + z\mathbf{e}_i) \odot \mathbf{d}\right)\left((\mathbf{w}^{t,i} + z\mathbf{e}_i) \odot \mathbf{d}\right)^\intercal + \frac{\alpha}{l} \sum_{j \in \mathbb{I}(\mathbf{w}^{t,i} + z\mathbf{e}_i))} (b_j(\mathbf{w}^{t,i} + z\mathbf{e}_i))^2 \end{aligned} \tag{6.12}$$

where $\mathbf{e}_i \in \mathbb{R}^{1\times(n+1)}$ is a vector with the i-th entry 1 and all other entries 0.

The first derivative of (6.12) with respect to z is:

$$F_i'(z) = \lambda\left(w_i^{t,i} + z\right) \cdot \mathbf{d}_i - \frac{2\alpha}{l} \sum_{j \in \mathbb{I}(\mathbf{w}^{t,i} + z\mathbf{e}_i))} (y_j x_{ji} (b_j(\mathbf{w}^{t,i} + z\mathbf{e}_i)) \tag{6.13}$$

As pointed out in [57], $F_i(z)$ is not twice differential at some j, where $b_j(\mathbf{w}^{t,i} + z\mathbf{e}_i) = 0$. Following [57, 74], we define the generalized second derivative of (6.12) with respect to z as:

$$F_i''(z) = \lambda \mathbf{d}_i + \frac{2\alpha}{l} \sum_{j \in \mathbb{I}(\mathbf{w}^{t,i} + z\mathbf{e}_i))} x_{ji}^2 \tag{6.14}$$

The Newton direction at a given z is $\frac{F_i'(z)}{F_i''(z)}$. We start from $z = 0$ and apply a line search $z = z - \eta_i \frac{F_i'(z)}{F_i''(z)}$ until $F_i(z - \eta_i \frac{F_i'(z)}{F_i''(z)}) < F_i(z)$, where η_i is the learning rate for the i-th element and is chosen from $\{1, \beta, \beta^2, \ldots\}$.

6.2.3 Implementation Details and Analysis

For each vertex, the random walk takes arbitrary m hops. In the following, we use the power method to **implicitly** decide the value of m for each vertex. Note that different vertices may have different numbers of hops and **we do not explicitly compute $\mathbf{P}^m$** owing to its high time complexity. If we denote the m-th term $\mathbf{p}_{q\cdot} \cdot \frac{\rho^m}{m!}\mathbf{P}^m$ in $\mathbf{x}_q$ by $\mathbf{v}_q^m$, then $\mathbf{v}_q^m = \frac{\rho}{m}\mathbf{v}_q^{m-1} \cdot \mathbf{P}$. We define the velocity at $m-1$ to be the vector $\boldsymbol{\delta}_q^{m-1} = \mathbf{v}_q^m - \mathbf{v}_q^{m-1}$, the acceleration at $m-1$ to be the vector $\boldsymbol{\epsilon}_q^{m-1} = \boldsymbol{\delta}_q^m - \boldsymbol{\delta}_q^{m-1}$ and stop the iteration when $\|\boldsymbol{\epsilon}_q^{m-1}\|_{max}$ is below a threshold $\hat{\epsilon}$. We set $\rho = 5$ according to [58], $\lambda = 2^{-6}$ according to [91],

and $\alpha = 1, \beta = \frac{1}{2}$ according to [57]. The pseudo-code for our binary classifier wvGN is given in Algorithm 5.

In Algorithm 5, steps 4–14 are the details of using the power method to compute the representation $\mathbf{x}_q$ of the vertex v_q. Note that this part can be parallelized since the representation computation for each vertex is independent. Step 16 uses our GCD to optimize (6.9). As described in Procedure (lines 19–25), our GCD method iteratively conducts GD and CD to escape from local optima. Given the randomly generated initial $\mathbf{w}$, we first apply a coarse search by GD. If the change of the objective (6.9) is less than a threshold (10^{-4}), we apply a fine search by CD; otherwise, we continue the coarse search by GD. We repeat the process until the maximum iteration is reached. In the optimization process, we find CD costs a lot of time. Inspired by the mini-batch SGD [92], we use CD to update only a number $k = 10\%$ of the elements randomly selected in $\mathbf{w}$. We find such a mini-batch CD method accelerates the search but does not deteriorate the results (see Section 6.3.1).

Complexity analysis. Assume that a graph has n vertices and r edges. In Algorithm 5, lines 1–3 costs $\mathcal{O}(r)$ time. Lines 4–14 adopts the power method to approximately compute the representation for each vertex. Since the time complexity of the power method is $\mathcal{O}(r)$ [39], we need $\mathcal{O}(n \cdot r)$ to finish steps 4–14. Procedure (lines 19–25) gives the pseudo-code for our GCD method. Line 22 uses GD to update $\mathbf{w}$. The time complexity of GD is bound by the complexity of computing the gradient (Equation (6.10)), i.e., $\mathcal{O}\left(l \cdot (n+1)\right)$, where $l \ll n$ is the number of the labeled vertices. Line 23 computes $\Delta F(\mathbf{w})$, whose time complexity is $\mathcal{O}\left(l \cdot (n+1)\right)$. Line 24 updates $\mathbf{w}$ by the mini-batch CD, whose time complexity is determined by the first derivative (Equation (6.13)). It costs $\mathcal{O}\left(l \cdot (n+1)\right)$ to update one element in $\mathbf{w}$. Thus the time complexity to update k elements in $\mathbf{w}$ is $\mathcal{O}\left(k \cdot l \cdot (n+1)\right)$. The total time complexity of GCD is bound by $\mathcal{O}(k \cdot l \cdot n)$. Finally, the time complexity of our method wvGN is $\mathcal{O}\left(n \cdot (r + k \cdot l)\right)$.

Algorithm 5: wvGN

Input: Affinity matrix $\mathbf{A}$ for a graph and labels $\mathbf{y}_l = [y_1, \ldots, y_l]$ for the labeled vertices $\{v_1, \ldots, v_l\}$

Output: estimated $\hat{\mathbf{y}}_u = [\hat{y}_{l+1}, \ldots, \hat{y}_n]$ for the unlabeled vertices $\{v_{l+1}, \ldots, v_n\}$

$\rho \leftarrow 5, t \leftarrow 50, \hat{\epsilon} \leftarrow 10^{-4}, \lambda \leftarrow 2^{-6}, \alpha \leftarrow 1, \beta \leftarrow \frac{1}{2}$;
compute the degree matrix $\mathbf{D}$;
compute the transition matrix $\mathbf{P} \leftarrow \mathbf{D}^{-1}\mathbf{A}$;
`/* power iteration method */`
for $j \leftarrow 1$ **to** n **do**
    $\mathbf{v}_j^1 \leftarrow \mathbf{p}_{j\cdot} \frac{\rho^1}{1!} \cdot \mathbf{P}$;
    $m \leftarrow 1$;
    $\mathbf{x}_j \leftarrow \mathbf{v}_j^1$;
    repeat
        $\mathbf{v}_j^{m+1} \leftarrow \frac{\rho}{m+1}\mathbf{v}_j^m \cdot \mathbf{P}$;
        $\mathbf{x}_j \leftarrow \mathbf{x}_j + \mathbf{v}_j^{m+1}$;
        $\boldsymbol{\delta}^m \leftarrow |\mathbf{v}_j^{m+1} - \mathbf{v}_j^m|$;
        $m \leftarrow m + 1$;
    until $\|\boldsymbol{\delta}_j^{m+1} - \boldsymbol{\delta}_j^m\|_{\max} \leq \hat{\epsilon}$ **or** $m \geq t$;
    $\mathbf{x}_j \leftarrow [\mathbf{x}_j, 1]$;
$\mathbf{X} \leftarrow [\mathbf{x}_1; \ldots; \mathbf{x}_n], \mathbf{d} \leftarrow \left[diag(\mathbf{D}^{-\frac{1}{2}})^\intercal, 1\right]$;
$\mathbf{w} \leftarrow$ `GCD(`$\mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta$`);` `/* X_l is the labeled data (training data). */`
$\hat{\mathbf{y}}_u \leftarrow sign(\mathbf{X}_u \cdot \mathbf{w}^\intercal)$; `/* X_u is the unlabeled data (test data). */`
return $\hat{\mathbf{y}}_u$;
Procedure `GCD(`$\mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta$`)`
    $iter \leftarrow 25$, $\mathbf{w} \leftarrow$`rand` (1,n), $\mathbf{w} \leftarrow \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$;
    for $i \leftarrow 1$ **to** $iter$ **do**
        $\mathbf{w} \leftarrow$`GD` $(\mathbf{w}, \mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta)$;
        if $\Delta F(\mathbf{w}) < 10^{-4}$ **then**
            $\mathbf{w} \leftarrow$`miniBatchCD` $(\mathbf{w}, \mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta)$;
    return $\mathbf{w}$;

6.3 Experimental Evaluation

Our experiments evaluate the classification performance of wvGN and its competitors on synthetic and real-world data. We compare wvGN with the state-of-the-art methods from related research fields. To be specific, the baseline methods are as follows:

- Semi-supervised learning. TSVM [93, 96], LapSVM [67] and LGC [27]. We use the rows of the transition matrix $\mathbf{P}$ to represent vertices and input them to TSVM and LapSVM.
- Relational learning. wvRN [89], SocDim [64] and SCRN [104].
- Random walk based graph learning. Deepwalk [15], node2vec [2] and SNBC [91].
- Graph diffusion based learning. Heat kernel diffusion [35]. We use the power iteration method to approximately compute the heat kernel.

For more descriptions on those competitors, please refer to our related work and their original papers. For the datasets that have more than two classes, we use the one-vs-rest [18] method to train wvGN for each class, which leads to c (the number of classes) decision values for each vertex. However, the decision values generated by each binary wvGN cannot be compared directly because they are not in the same scale. According to [91], we use Platt's Scaling [54] to transform these decision values to probability scores which are based on the same scale and can be compared directly. We assign the most probable class label to each vertex. To validate results, we use two popular evaluation measures from [80, 91, 104]: Micro-F1 score and Macro-F1 score. The higher the values of these evaluation measures, the better the classification.

We randomly sample a number of vertices with labels from each class as training data and use the rest of vertices as test data. Following the settings in [2, 15, 64], we repeat this process ten times and report the average and standard deviation of Micro-F1 score and Macro-F1 score for each method. The default parameters are adopted for the baseline methods according to their original papers. All experiments are run on the same machine

with an Intel Core Quad i7-3770 with 3.4 GHz and 32 GB RAM. All graph shown in this thesis are plotted by the python toolbox *NetworkX*. The code of wvGN and all synthetic and real-world data used in this work are available at the website[1].

6.3.1 Synthetic Data

We use the benchmark graph generator [8] to generate two synthetic graphs GRAPH1 and GRAPH2. The generator makes the vertex degree and community size follow power law distributions which reflect the real properties of vertices and communities found in real graphs. NETWORK1 has 100 vertices and 1008 edges, which are grouped into two classes (37 and 63 vertices, respectively). The average degree is 21. NETWORK2 has 120 vertices and 607 edges, which are grouped into three classes (19, 43 and 58 vertices, respectively). The average degree is 10. For each graph, we vary the number of labeled vertices in each class from one to five and report the average and standard deviation of Micro-F1 and Macro-F1 scores in Table 6.1 and Table 6.2.

From Table 6.1, we can see that our method wvGN achieves the best results compared to the baselines. wvGN achieves an average Micro-F1 score 0.987 and an average Macro-F1 score 0.986 even just given one labeled vertex in each class. In this case, wvGN achieves a gain of 1.86% (min) over node2vec and 161.1% (max) over TSVM. Thus, there is no much space left for wvGN to improve its results when given more labeled vertices. However, for its baselines, we can see that most of them have an ascending performance when given more labeled vertices because they have a relatively worse starting point. Note that the graph embedding method node2vec is very competitive. Compared with the NETWORK1, NETWORK2 is more complex. From Table 6.2, we can see that every method continuously increases their performance with increasing number of labeled vertices. When given five labeled vertices in each class, wvGN achieves a gain of 34.1% (min) over node2vec and 183.2% (max) over LGC in terms of Macro-F1 score. "wvGN (full)" means wvGN with the full use of CD, i.e., all elements in $\mathbf{w}$ are updated by CD. Table 6.1 reveals that wvGN

[1] https://github.com/yeweiysh/wvGN

achieves approximately the same performance as wvGN (full) while Table 6.2 shows that wvGN is better than wvGN (full) when the number of labeled vertices exceeds two.

Figure 6.4 and Figure 6.5 give us the intuitive demonstrations of classification on NETWORK1 and NETWORK2 (the labeled vertices are in black). To reduce the clutter, we only show the results of the top seven methods here. Note that wvGN only misclassifies one blue vertex (highlighted in red in Figure 6.4(b)). This vertex has three edges connected to the green class but only one edge to the blue class. Since the graph is unweighted, the more a vertex has edges connected to a class, the more similar the vertex to the class. According to the weighted vote strategy (see Section 6.2.1), it makes sense to classify it as the green class. Figure 6.4(c) depicts the classification result of node2vec. Three blue vertices are misclassified. Figure 6.4(d) shows that SNBC misclassifies eleven blue vertices. Figure 6.5(b)–(d) show the classification results of wvGN, node2vec and TSVM on NETWORK2. wvGN misclassifies 25 vertices. node2vec misclassifies 41 vertices. TSVM misclassifies 54 vertices. wvGN achieves the best results and has a gain of 20.9% over the second best method node2vec and 45.8% over the third best method TSVM in terms of Micro-F1 score.

Figure 6.6 (a) and (b) compare our optimization method GCD with GD, CD and SGD on NETWORK1 and NETWORK2. We can see that all GD, CD and SGD get trapped in local optimal, which leads to poor performance, especially on NETWORK1. When combining GD and CD, our method GCD improves the performance.

6.3.2 Real-world Data

For the real-world data, we use four popular relational datasets CORA, PUBMED, IMDB from [91] and WIKIPEDIA from [2]. Their statistics are given in Table 6.3.

CORA is a collection of research articles in computer science and PUBMED is a collection of research articles in diabetes. Both CORA and PUBMED are sparse citation graphs. Vertices tend to have low degree in such graphs. By using global neighborhood information, we can see from Table 6.4 and Table 6.5 that wvGN improves the classification

Table 6.1: Classification results on GRAPH1 with the varying number of labeled vertices (#LV) in each class.

	Micro-F1				
#LV	1	2	3	4	5
wvGN	**0.987±0.007**	**0.989±0.003**	**0.993±0.005**	**0.990±0.003**	**0.992±0.005**
wvGN (full)	**0.987±0.005**	**0.988±0.007**	**0.993±0.005**	**0.990±0.006**	**0.992±0.005**
node2vec	0.969±0.005	0.945±0.060	0.948±0.051	0.945±0.056	0.942±0.054
Deepwalk	0.746±0.267	0.756±0.207	0.778±0.189	0.759±0.201	0.796±0.120
SNBC	0.856±0.156	0.812±0.174	0.714±0.107	0.713±0.105	0.749±0.103
wvRN	0.779±0.192	0.666±0.185	0.763±0.211	0.867±0.177	0.926±0.073
SCRN	0.773±0.224	0.742±0.218	0.812±0.172	0.863±0.136	0.902±0.106
SocDim	0.552±0.099	0.520±0.085	0.537±0.058	0.572±0.062	0.600±0.063
HeatKernel	0.824±0.198	0.688±0.200	0.786±0.212	0.882±0.163	0.909±0.119
LGC	0.604±0.186	0.556±0.183	0.605±0.219	0.621±0.221	0.624±0.170
TSVM	0.378±0	0.375±0	0.372±0	0.370±0	0.367±0
LapSVM	0.519±0.094	0.529±0.088	0.560±0.085	0.582±0.144	0.651±0.128
	Macro-F1				
#LV	1	2	3	4	5
wvGN	**0.986±0.007**	**0.988±0.004**	**0.992±0.006**	**0.989±0.003**	**0.992±0.005**
wvGN (full)	**0.986±0.005**	0.987±0.007	**0.992±0.006**	**0.989±0.007**	**0.992±0.006**
node2vec	0.967±0.005	0.943±0.059	0.946±0.051	0.942±0.055	0.940±0.054
Deepwalk	0.743±0.269	0.745±0.224	0.770±0.199	0.750±0.212	0.787±0.124
SNBC	0.840±0.166	0.809±0.178	0.709±0.108	0.705±0.106	0.738±0.100
wvRN	0.715±0.252	0.551±0.240	0.708±0.257	0.843±0.204	0.917±0.082
SCRN	0.666±0.323	0.610±0.318	0.735±0.256	0.806±0.215	0.870±0.162
SocDim	0.518±0.102	0.497±0.075	0.524±0.046	0.549±0.058	0.584±0.054
HeatKernel	0.777±0.259	0.579±0.261	0.738±0.258	0.866±0.180	0.902±0.123
LGC	0.451±0.224	0.395±0.197	0.475±0.264	0.469±0.270	0.451±0.201
TSVM	0.274±0	0.273±0	0.271±0	0.270±0	0.268±0
LapSVM	0.416±0.089	0.456±0.099	0.497±0.082	0.529±0.167	0.624±0.124

results when the percent of labeled vertices exceeds 1%. From Table 6.4, we can see that wvRN and HeatKernel are two very competitive baselines considering their classification results and simplicity. When the percent of labeled vertices is 1%, wvRN is superior to our method wvGN in terms of Micro-F1 score. When the percent of labeled vertices is increased to 3%, wvGN achieves a gain of 334.5% (max) over Deepwalk and 6.24% (min) over wvRN in terms of Macro-F1 score. Table 6.5 demonstrates that HeatKernel is better than wvGN when the percent of labeled vertices is 1%. wvGN achieves a gain of 120.6% (max) over wvRN and 2.92% (min) over HeatKernel in terms of Macro-F1 score when the percent of labeled vertices is 3%.

Differing from CORA and PUBMED, IMDB is produced based on the vertex simi-

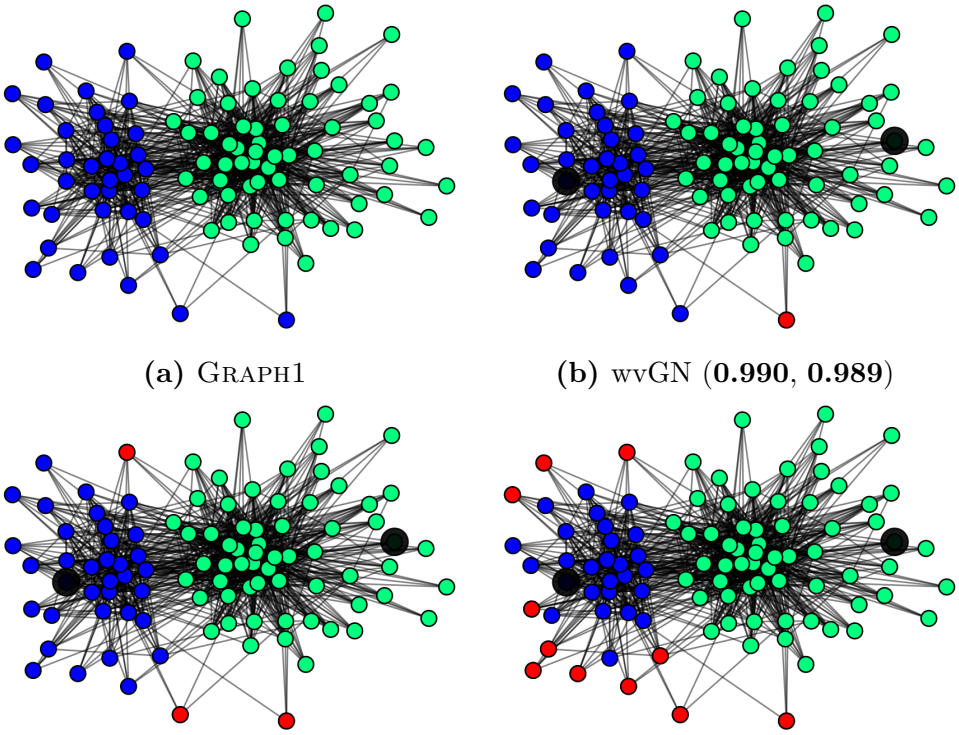
Table 6.2: Classification results on GRAPH2 with the varying number of labeled vertices (#LV) in each class.

	Micro-F1				
#LV	1	2	3	4	5
wvGN	0.667±0.114	0.718±0.110	**0.773±0.145**	**0.890±0.027**	**0.889±0.025**
wvGN (full)	**0.674±0.101**	**0.799±0.076**	0.696±0.158	0.863±0.035	0.841±0.058
node2vec	0.604±0.097	0.705±0.073	0.618±0.115	0.679±0.109	0.679±0.109
Deepwalk	0.424±0.074	0.465±0.071	0.429±0.126	0.512±0.059	0.512±0.059
SNBC	0.358±0.67	0.455±0.128	0.487±0.123	0.560±0.050	0.572±0.048
wvRN	0.450±0.121	0.560±0.157	0.569±0.113	0.681±0.075	0.681±0.075
SCRN	0.459±0.146	0.503±0.148	0.573±0.103	0.572±0.048	0.652±0.103
SocDim	0.310±0.079	0.370±0.076	0.429±0.072	0.445±0.047	0.445±0.047
HeatKernel	0.483±0.101	0.573±0.164	0.553±0.126	0.651±0.080	0.651±0.080
LGC	0.312±0.051	0.313±0.047	0.315±0.080	0.352±0.048	0.352±0.048
TSVM	0.542±0.050	0.646±0.075	0.517±0.041	0.520±0.049	0.520±0.049
LapSVM	0.376±0.132	0.494±0.107	0.583±0.105	0.631±0.085	0.631±0.085
	Macro-F1				
#LV	1	2	3	4	5
wvGN	0.677±0.122	0.733±0.116	**0.791±0.150**	**0.909±0.025**	**0.909±0.021**
wvGN (full)	**0.681±0.104**	**0.823±0.071**	0.714±0.151	0.877±0.042	0.853±0.060
node2vec	0.589±0.123	0.718±0.082	0.612±0.126	0.678±0.123	0.678±0.123
Deepwalk	0.407±0.072	0.452±0.068	0.417±0.117	0.485±0.058	0.485±0.058
SNBC	0.353±0.120	0.438±0.131	0.460±0.139	0.567±0.058	0.579±0.049
wvRN	0.362±0.139	0.494±0.178	0.539±0.149	0.648±0.116	0.648±0.116
SCRN	0.363±0.164	0.397±0.183	0.501±0.152	0.579±0.049	0.587±0.174
SocDim	0.291±0.067	0.354±0.065	0.413±0.067	0.421±0.052	0.421±0.052
HeatKernel	0.402±0.145	0.522±0.177	0.525±0.147	0.596±0.120	0.596±0.120
LGC	0.296±0.035	0.290±0.036	0.284±0.067	0.321±0.038	0.321±0.038
TSVM	0.535±0.050	0.647±0.067	0.514±0.041	0.517±0.049	0.517±0.049
LapSVM	0.234±0.086	0.391±0.112	0.569±0.123	0.660±0.079	0.660±0.079

Table 6.3: Statistics of Datasets.

Dataset	#vertices	#edges	#classes
CORA	24,519	92,207	10
PUBMED	19,717	44,324	3
IMDB	19,359	362,079	21
WIKIPEDIA	4,777	184,812	40

larity. Most of the vertices in the graph have similar degrees. Compared with CORA and PUBMED, IMDB is a more difficult graph to classify. We can see from Table 6.6 that wvGN is superior to its competitors in terms of Micro-F1 score when the percent of labeled vertices is greater than 1%. wvGN achieves a gain of 183.7% (max) over node2vec and



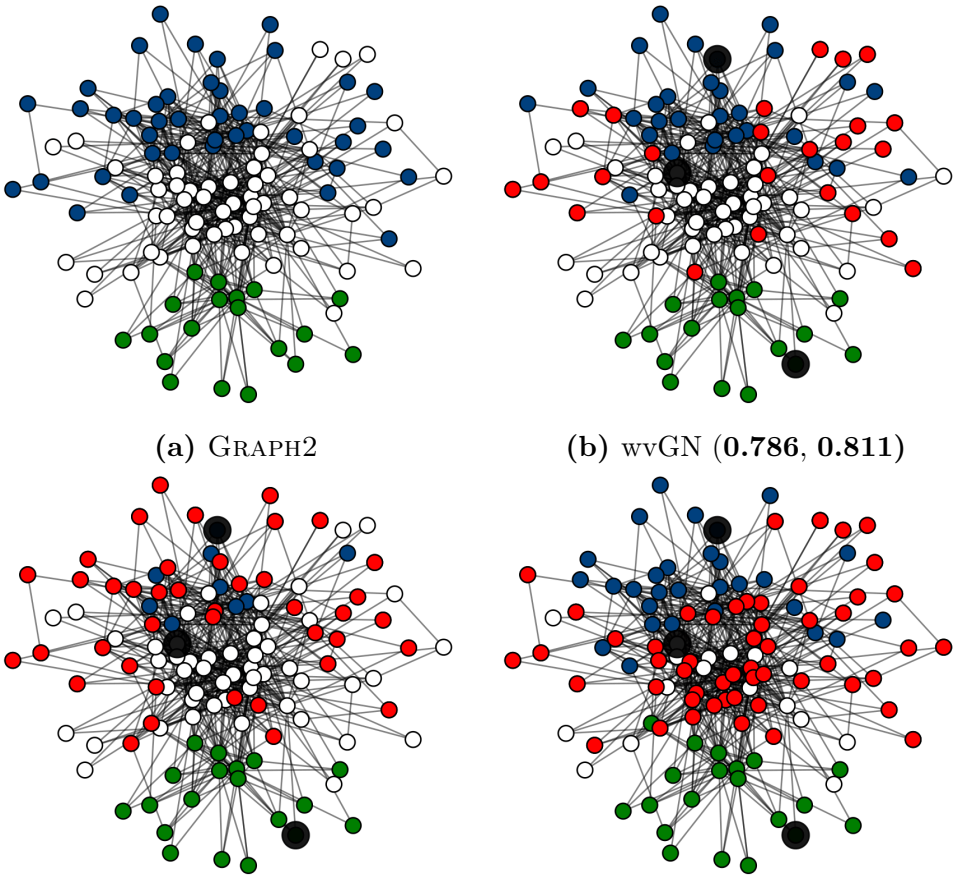
(a) Graph1 **(b)** wvGN (**0.990**, **0.989**)

(c) node2vec (0.969, 0.967) **(d)** SNBC (0.888, 0.871)

Figure 6.4: Classification results on Graph1. The two labeled vertices are in black and the misclassified vertices are in red. Subcaption: Method (Micro-F1, Macro-F1).

6.37% (min) over SocDim in terms of Micro-F1 score when the percent of labeled vertices is 7%. However, it is defeated by SocDim in terms of Micro-F1 score when the percent of labeled vertices is 1%. It is also defeated by HeatKernel and SNBC in terms of Macro-F1 score. To sum up, wvGN achieves four out of ten best results; HeatKernel achieves three; SNBC achieves two; SocDim achieves one.

Wikipedia is a co-occurrence graph of words appearing in the first million bytes of the Wikipedia dump. It is a highly noisy graph with lots of interclass edges. Our method wvGN is better than its competitors in terms of Micro-F1 score when the percent of labeled vertices exceeds 1%. To be specific, wvGN achieves a gain of 336.5% (max) over wvRN and 6.82% (min) over SNBC when the percent of labeled vertices is 7%. SNBC outperforms wvGN when the percent of labeled vertices is 1%, but the gap is very narrow, only a gain of 0.72%. In terms of Macro-F1 score, node2vec outperforms wvGN when the percent of



(a) Graph2 **(b)** wvGN (**0.786**, **0.811**)

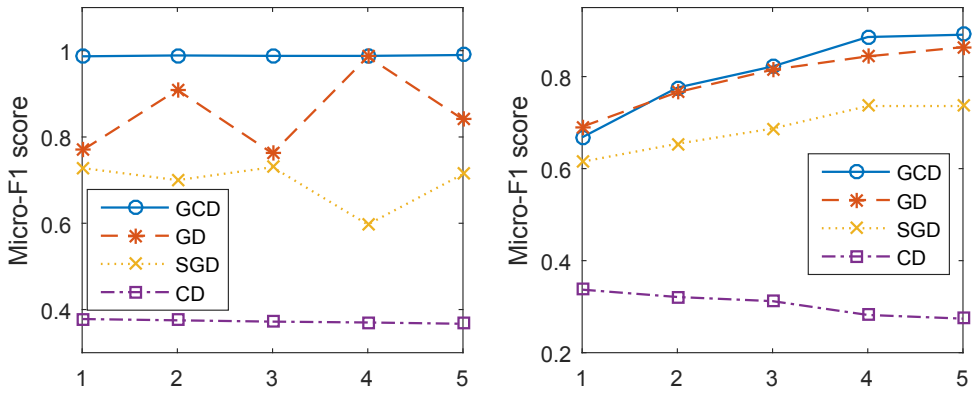
(c) node2vec (0.650, 0.622) **(d)** TSVM (0.539, 0.537)

Figure 6.5: Classification results on Graph2. The three labeled vertices are in black and the misclassified vertices are in red. Subcaption: Method (Micro-F1, Macro-F1).

labeled vertices is greater than 1%. To sum up, wvGN achieves five out of ten best results; node2vec achieves four; SNBC achieves one.

6.4 Related Work

Semi-supervised learning. Transductive SVM (TSVM) [93,96] achieves the aim of max-margin classification while ensuring that the unlabeled instances are put backward from



(a) GRAPH1 (b) GRAPH2

Figure 6.6: Optimization comparison on the synthetic data. The x-axis represents the number of labeled vertices in each class.

Table 6.4: Classification results on CoRA with the varying percent of labeled vertices (%LV). N/A means the results are not available because the algorithm is not finished in one week.

	Micro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	0.628±0.028	**0.716±0.009**	**0.742±0.006**	**0.754±0.005**	**0.758±0.004**
node2vec	0.497±0.154	0.503±0.073	0.504±0.077	0.542±0.075	0.548±0.080
Deepwalk	0.232±0.016	0.165±0.019	0.188±0.014	0.241±0.002	0.267±0.009
SNBC	0.501±0.026	0.630±0.011	0.665±0.009	0.680±0.009	0.682±0.009
wvRN	**0.652±0.014**	0.700±0.007	0.721±0.005	0.732±0.006	0.743±0.003
SCRN	0.643±0.019	0.703±0.007	0.726±0.004	0.736±0.004	0.746±0.003
SocDim	0.493±0.009	0.556±0.005	0.596±0.006	0.625±0.008	0.636±0.004
HeatKernel	0.643±0.020	0.696±0.007	0.720±0.005	0.731±0.006	0.742±0.002
LGC	0.474±0.026	0.487±0.025	0.486±0.018	0.487±0.020	0.485±0.017
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A
	Macro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	**0.528±0.020**	**0.630±0.010**	**0.661±0.010**	**0.677±0.006**	**0.683±0.008**
node2vec	0.422±0.100	0.433±0.065	0.464±0.059	0.492±0.053	0.507±0.070
Deepwalk	0.150±0.005	0.145±0.012	0.169±0.010	0.200±0.001	0.216±0.007
SNBC	0.272±0.024	0.492±0.021	0.544±0.001	0.573±0.015	0.578±0.010
wvRN	0.525±0.022	0.593±0.010	0.619±0.008	0.637±0.006	0.649±0.006
SCRN	0.508±0.030	0.592±0.009	0.622±0.007	0.639±0.005	0.652±0.005
SocDim	0.278±0.013	0.448±0.008	0.499±0.009	0.534±0.012	0.548±0.008
HeatKernel	0.517±0.031	0.591±0.010	0.620±0.009	0.637±0.006	0.649±0.004
LGC	0.231±0.003	0.245±0.029	0.231±0.027	0.233±0.016	0.223±0.023
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A

Table 6.5: Classification results on PubMed with the varying percent of labeled vertices (%LV). N/A means the results are not available because the algorithm is not finished in one week.

	Micro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	0.630±0.057	**0.758±0.012**	**0.784±0.008**	**0.790±0.006**	**0.798±0.005**
node2vec	0.640±0.040	0.632±0.061	0.645±0.055	0.656±0.057	0.611±0.062
Deepwalk	0.332±0.014	0.350±0.010	0.352±0.008	0.355±0.008	0.361±0.009
SNBC	0.521±0.072	0.713±0.018	0.723±0.006	0.784±0.006	0.795±0.005
wvRN	0.358±0.008	0.359±0.005	0.359±0.004	0.359±0.003	0.360±0.003
SCRN	0.364±0.009	0.362±0.003	0.363±0.005	0.362±0.003	0.362±0.003
SocDim	0.426±0.016	0.476±0.019	0.515±0.012	0.552±0.021	0.573±0.017
HeatKernel	**0.679±0.024**	0.733±0.011	0.765±0.006	0.777±0.006	0.788±0.004
LGC	0.622±0.100	0.715±0.063	0.756±0.023	0.766±0.016	0.766±0.017
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A
	Macro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	0.595±0.064	**0.739±0.011**	**0.767±0.010**	**0.773±0.008**	**0.782±0.005**
node2vec	0.618±0.045	0.695±0.081	0.616±0.068	0.627±0.068	0.574±0.074
Deepwalk	0.327±0.012	0.345±0.009	0.347±0.009	0.350±0.006	0.357±0.009
SNBC	0.442±0.100	0.680±0.015	0.737±0.010	0.763±0.010	0.777±0.006
wvRN	0.332±0.003	0.335±0.003	0.333±0.004	0.334±0.003	0.333±0.002
SCRN	0.331±0.003	0.336±0.002	0.334±0.004	0.334±0.004	0.333±0.003
SocDim	0.377±0.032	0.433±0.041	0.477±0.031	0.531±0.033	0.554±0.022
HeatKernel	**0.659±0.024**	0.718±0.010	0.750±0.006	0.764±0.007	0.775±0.004
LGC	0.568±0.119	0.689±0	0.727±0.031	0.741±0.018	0.741±0.018
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A

the margin as far as possible. Its objective function is as follows:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}\cdot\mathbf{w}^{\intercal} + C_1\sum_{i=1}^{l}\max(1-y_i\cdot f(\mathbf{x}_i),0) + C_2\sum_{j=l+1}^{n}\max(1-y_j\cdot f(\mathbf{x}_j),0)$$

Here C_1 and C_2 are regularization parameters to control the relative hinge-loss on the labeled and unlabeled instances. TSVM first uses SVM to label the unlabeled instances and then switches labels to improve the objective function. This process is susceptible to local optima and requires a large number of label switches before converging. Frequent label switches lead to higher training time compared to SVM. Laplacian Regularized SVM (LapSVM) [67] extends SVM by including the intrinsic smoothness penalty term $\mathbf{f}^{\intercal}\cdot\mathbf{L}\cdot\mathbf{f}$

Table 6.6: Classification results on IMDB with the varying percent of labeled vertices (%LV). N/A means the results are not available because the algorithm is not finished in one week.

	Micro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	0.245±0.077	**0.408±0.019**	**0.415±0.009**	**0.434±0.009**	**0.441±0.008**
node2vec	0.115±0.027	0.147±0.011	0.141±0.005	0.153±0.010	0.180±0.011
Deepwalk	0.152±0.017	0.133±0.007	0.159±0.002	0.248±0.019	0.328±0.012
SNBC	0.203±0.083	0.358±0.024	0.353±0.010	0.348±0.007	0.348±0.008
wvRN	0.333±0.050	0.360±0.005	0.364±0.004	0.370±0.003	0.374±0.003
SCRN	0.336±0.062	0.367±0.009	0.371±0.003	0.378±0.004	0.380±0.004
SocDim	**0.372±0.018**	0.386±0.012	0.403±0.001	0.408±0.004	0.411±0.003
HeatKernel	0.308±0.068	0.356±0.012	0.372±0.006	0.393±0.007	0.412±0.008
LGC	0.371±0.040	0.394±0.001	0.397±0.001	0.399±0.001	0.399±0.001
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A
	Macro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	0.087±0.012	0.116±0.005	0.125±0.005	0.134±0.007	0.136±0.005
node2vec	0.086±0.019	0.119±0.004	0.117±0.002	0.121±0.003	0.128±0.004
Deepwalk	0.112±0.006	0.107±0.003	0.111±0.005	0.112±0.005	0.110±0.002
SNBC	0.076±0.015	0.123±0.007	0.144±0.004	**0.156±0.004**	**0.165±0.003**
wvRN	0.100±0.007	0.104±0.003	0.103±0.002	0.104±0.002	0.103±0.002
SCRN	0.093±0.009	0.095±0.004	0.095±0.002	0.100±0.003	0.100±0.002
SocDim	0.076±0.005	0.080±0.005	0.088±0	0.092±0.004	0.095±0.003
HeatKernel	**0.117±0.014**	**0.139±0.006**	**0.147±0.005**	0.155±0.006	0.164±0.006
LGC	0.083±0.004	0.090±0.002	0.094±0.002	0.094±0.001	0.095±0.001
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A

in SVM's objective function, where $\mathbf{L}$ is the Laplacian matrix. Because LapSVM needs to compute the inverse of a dense Gram matrix, its time complexity is $\mathcal{O}(n^3)$ which is impractical for learning on large-scale graphs. Local and Global Consistency (LGC) [27] predicts the labels of unlabeled instances following the prior assumption of consistency, i.e., nearby instances tend to have the same labels, and instances on the same structure (cluster or manifold) tend to have the same labels. During each iteration, each vertex not only receives the label information from its neighbors, but also retains its initial label information. The closed form expression for the vertices is $\beta\left(\mathbf{I}-\alpha\mathbf{S}\right)^{-1}\mathbf{Y}$, where $\mathbf{S}=\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and $\mathbf{Y}$ keeps the initial label information. The closed form expression needs to invert the matrix. For learning on large-scale graphs, the inversion operation often

Table 6.7: Classification on WIKIPEDIA with the varying percent of labeled vertices (%LV). N/A means the results are not available because the algorithm is not finished in one week.

	Micro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	0.416±0.010	**0.453±0.006**	**0.449±0.010**	**0.454±0.010**	**0.454±0.007**
node2vec	0.293±0.050	0.312±0.018	0.311±0.024	0.317±0.026	0.343±0.042
deepwalk	0.177±0.015	0.140±0.012	0.135±0.014	0.138±0.018	0.157±0.018
SNBC	**0.419±0.005**	0.425±0.004	0.424±0.006	0.425±0.006	0.426±0.005
wvRN	0.016±0.011	0.042±0.025	0.078±0.040	0.104±0.035	0.130±0.066
SCRN	0.017±0.013	0.042±0.021	0.083±0.041	0.118±0.031	0.150±0.062
SocDim	0.339±0.015	0.324±0.016	0.326±0.008	0.333±0.009	0.335±0.010
HeatKernel	0.013±0.010	0.043±0.033	0.077±0.043	0.105±0.038	0.128±0.087
LGC	0.367±0.065	0.389±0.001	0.390±0.001	0.390±0.001	0.391±0.001
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	0.274±0.139	0.419±0.006	0.420±0.006	0.422±0.003	0.422±0.006
	Macro-F1				
%LV	1%	3%	5%	7%	9%
wvGN	**0.065±0.004**	0.068±0.004	0.063±0.004	0.068±0.005	0.066±0.003
node2vec	0.064±0.004	**0.071±0.002**	**0.076±0.005**	**0.080±0.006**	**0.083±0.005**
deepwalk	0.042±0.004	0.041±0.002	0.041±0.003	0.040±0.003	0.040±0.002
SNBC	0.044±0.002	0.044±0.003	0.044±0.005	0.045±0.003	0.046±0.004
wvRN	0.007±0.004	0.011±0.006	0.016±0.005	0.020±0.007	0.022±0.006
SCRN	0.007±0.004	0.011±0.005	0.017±0.005	0.021±0.005	0.024±0.005
SocDim	0.056±0.002	0.064±0.003	0.065±0.004	0.068±0.005	0.072±0.004
HeatKernel	0.006±0.004	0.010±0.007	0.015±0.006	0.017±0.006	0.020±0.007
LGC	0.030±0.003	0.030±0	0.030±0	0.030±0	0.029±0
TSVM	N/A	N/A	N/A	N/A	N/A
LapSVM	0.024±0.007	0.041±0.004	0.042±0.003	0.043±0.003	0.042±0.004

consumes a lot of time and resources. In addition, if the assumption of consistency is not met, LGC tends to fail. Compared with LGC and other label-propagation based semi-supervised learning methods, our method wvGN makes no such local and global label consistency assupmtions but directly learns geometic neighborhood information from data, which is then used to infer the labels of vertices. Linear Neighborhood Propagation (LNP) [42] assumes that each data point can be linearly reconstructed from its neighborhood and it uses this assumption to construct the graph from data. For the graph data from which we do not need to construct the graph, LNP deteriorates to LGC.

Relational learning. The Relational Neighbor (RN) [41] classifier is a simple classifier which only uses the class labels of known related instances without doing learning. RN

works by making two strong yet often reasonable assumptions: 1) some instances' class labels are known within the same linked structure and 2) instances related to each other are similar and likely belong to the same class (also called homophily [68]). However, RN may not perform well if the labeled instances in the graph are isolated. Instead of making a hard labeling during the inference process, the weighted-vote Relational Neighbor classifier (wvRN) [89] extends RN by assigning class labels to instances with some probabilities. Since the number of the labeled instances in graphs is small, both RN and wvRN need to propagate the known label information to the related instances by a collective inference procedure. The above two relational classifiers focus on the single-label classification problem. However, in many real relational graphs, each entity may belong to multiple classes. SocDim [64] first extracts latent social dimensions via the top eigenvectors of the modularity matrix and then uses them as features for discriminative learning. The extracted social dimensions describe each instance's hidden relations in the graph, which is specially useful when the graph has multiple diverse relations inside. Since the extracted latent social dimensions by SocDim are dense which is not scalable for large-scale graphs, EdgeCluster [65] partitions the edges into disjoint sets such that each set represents one latent social dimension. To achieve this, a variant of k-means is proposed to handle clustering of many edges. Then a linear SVM is adopted to classify those extracted social dimensions. SCRN [104] is a method designed for the multi-label graphs. It starts by constructing a social feature space which is an edge-centric representation of social dimensions to capture the vertex's potential affiliations. To describe each vertex's intrinsic correlation to each class, SCRN assigns each vertex a class-propagation probability. Finally, it assigns the label to a vertex considering its neighbors' class labels, the similarity to its neighbors and its class-propagation probability. ghostEdge [11] works by adding ghost edges to a graph to enable the flow of information from labeled vertices to unlabeled vertices. It combines the aspects of statistical relational learning and semisupervised learning in one framework. Within-Network Classification (WNC) [50] proposes structural-aware vertex features to deal with the situation where the theory of homophily does not hold. WNC only considers the patterns within a given radius threshold, which is incapable of capturing

long distance relationships in graphs.

Random walk based learning in graphs. Deepwalk [15] uses local information obtained from truncated random walks to learn latent representations of vertices in a graph. It models a stream of short random walks on graphs as natural language sentences, which is reasonable because both the degree distribution of a connected graph and the distribution of words in the natural language follow power law distributions. node2vec [2] is a semi-supervised method for scalable feature learning in graphs. It learns a mapping of vertices to a low-dimensional feature space, which maximizes the likelihood of preserving graph neighborhoods of vertices. To efficiently explore diverse neighborhood, a biased random walk procedure is proposed, which compromises breadth-first sampling (BFS) and depth-first sampling (DFS). SNBC [91] is a novel structural neighborhood-based learning method based on the lazy random walk. The classification of a vertex is decided based on how it is labeled in the respective k-th level neighborhood. The classification results are affected seriously by the form of the regularization on $\mathbf{w}$. Our method wvGN exploits random walks to explore geometirc one- to m-hop neighborhood information of a vertex. And the label of the vertex is determined by the accumulated geometric one- to m-hop neighborhood information. wvGN uses a proposed gradient and coordinate descent (GCD) method to optimize its objective function. GCD is robust to the starting point and does not easily get trapped in local optima.

Graph diffusion based learning. The most related graph diffusion method to our work is the heat kernel diffusion [35]. It is defined as $\mathbf{h} = \exp(-\rho)\left(\sum_{i=0}^{\infty}\frac{\rho^i}{i!}\mathbf{P}^i\right)\mathbf{s} = \exp\left(-\rho(\mathbf{I}-\mathbf{P}^{-1})\right)\mathbf{s}$, where $\mathbf{s}$ stores the initial class label information of the labeled vertices. It diffuses the class label information of the labeled vertices to the whole graph through the above formula. In the classification procedure, it first compares the amount of information a vertex receives from different classes. Then it assigns the vertex to the class which diffuses the most information to the vertex. MultiRankWalk [38] is based on the personalized pagerank diffussion and we find it is inferior to the heat kernel diffusion in the experiments.

6.5 Summary

We have proposed wvGN to tackle the problem of intra-graph classification when the number of labeled vertices is limited. wvGN is a semi-supervised learning framework which exploits both labeled and unlabeled vertices to achieve better classification. Conventional graph-based semi-supervised and relational learning methods either make assumptions of local and global label consistency or do not learn from data, and thus they do not perform well if the assumption is not met or the starting inferene procedure has some errors. To conquer these, wvGN learns geometic neighborhood information directly from data. It optimizes an objective function based on L2-loss SVM. A search strategy based on the gradient and coordinate descent methods has been developed to solve the problem of local optima. Empirical studies prove that our method wvGN is superior to state-of-the-art methods.

Chapter 7

Conclusion and Future Work

The scope of this thesis was restricted to developing unsupervised and semi-supervised learning methods on three types of complex data, i.e. numerical data, plain graph data and attributed graph data. We have advanced the state-of-the-art by proposing novel methods to address three challenges in data mining: (1) scaling up for high dimensional data and high speed data streams, (2) mining complex knowledge from complex data, and (3) data mining in a network setting: community and social networks. The proposed novel methods are based on the concepts of independence, unimodality and homophily.

To deal with the first challenge, we have proposed FUSE and ISAAC. FUSE is suitable for multi-scale data on which the normalized cut criterion tends to fail even given a suitable locally scaled affinity matrix. FUSE exploits the power iteration method to fuse cluster-separation information from all "informative" eigenvectors. Because the pseudo-eigenvectors generated by the power iteration method are redundant, ICA is adopted to reduce the redundancy. When the data dimensions become large, "the curse of dimensionality" appears and clusters only exist in subspaces. Many existing subspace clustering methods report redundant clusters which result in high runtime, low quality results and overwhelming data analysts. ISAAC was proposed to find multiple non-redundant clusters in high-dimensional data. ISAAC is based on ISA and MDL. ISA was adopted to find independent subspaces and MDL was used to automatically choose parameters that are

difficult to set.

To deal with the second challenge, we have proposed a method called UNCut to find cohesive clusters in attributed graphs that have two types of data, i.e. plain graph data and numerical data. Since not all the attributes of the numerical data are relevant to the graph structure, it is difficult to detect cohesive clusters which have densely connected edges and have as many homogeneous attributes as possible. The homogeneity of attribute is measured by the unimodality compactness that is based on the Hartigans' dip test. The assumption we make is that each cluster takes a unimodal distribution. We combined the unimodality compactness and the normalized cut as the objective function and presented an approximate solution that used the power iteration method.

To deal with the third challenge, we have proposed a method call wvGN for the semi-supervised vertex classification in graphs. wvGN is based on the theory of homophily, i.e. similar vertices tend to be connected with each other. wvGN exploits random walks to explore geometric one- to m-hop neighborhood information of a vertex. It infers the label of a vertex by the accumulated geometric neighborhood information. The form of the objective function of wvGN is close to l2-loss SVM. To get a better solution, we proposed an optimization method that combined gradient descent and coordinate descent. wvGN can be parallelized for the large-scale graphs.

From the perspective of effectiveness, the proposed four methods are empirically superior to the state-of-the-art on synthetic and real-world data with respect to appropriate quality measures. However, there are still some limitations of our methods. For example, FUSE needs the input parameters. ISAAC is quadratic in the number of data points and cubic in the number of dimensions. UNCut is only for the detection of non-overlapping cohesive clusters in attributed graphs. The graph vertex representation in the vector space is not sparse in wvGN and thus leads to the high optimization time. We will devise new strategies to conquer those limitations in the future work.

Bibliography

[1] A. Gray and A. Moore: *Nonparametric density estimation: Toward computational tractability*. In *SDM*. SIAM (2003) Pages 203–211.

[2] A. Grover and J. Leskovec: *node2vec: Scalable Feature Learning for Networks*. In *SIGKDD*. (2016) Pages 855–864.

[3] A. Hyvärinen: *Fast and robust fixed-point algorithms for independent component analysis*. In *Neural Networks, IEEE Transactions on*, Volume 10. (1999) Pages 626–634.

[4] A. Hyvärinen and P. Hoyer: *Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces*. In *Neural computation*, Volume 12. MIT Press (2000) Pages 1705–1720.

[5] A. Hyvärinen and E. Oja: *Independent component analysis: algorithms and applications*. In *Neural networks*, Volume 13. (2000) Pages 411–430.

[6] A.K. Jain and R.C. Dubes: *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[7] A. Krause and V. Liebscher: *Multimodal projection pursuit using the dip statistic*. In *Preprint-Reihe Mathematik*, Volume 13. (2005).

[8] A. Lancichinetti, S. Fortunato and F. Radicchi: *Benchmark graphs for testing community detection algorithms*. In *Physical review E*, Volume 78. APS (2008) Pages 046–110.

[9] A.Y. Ng, M.I. Jordan, Y. Weiss et al. : *On spectral clustering: Analysis and an algorithm*. In *Advances in neural information processing systems*, Volume 2. (2002) Pages 849–856.

[10] A. Strehl and J. Ghosh, *Journal of machine learning research* **3** (2002), 583.

[11] B. Gallagher, H. Tong, T. Eliassi-Rad and C. Faloutsos: *Using ghost edges for classification in sparsely labeled networks*. In *SIGKDD*. (2008) Pages 256–264.

[12] B. Liebl, U. Nennstiel-Ratzel, R. von Kries, R. Fingerhut, B. Olgemöller, A. Zapf and A.A. Roscher: *Very high compliance in an expanded MS-MS-based newborn screening program despite written parental consent*. In *Preventive medicine*, Volume 34. Elsevier (2002) Pages 127–131.

[13] B. Nadler and M. Galun: *Fandamental limitations of spectral clustering*. In *Advances in Neural Information Processing Systems*. (2006) Pages 1017–1024.

[14] B. Perozzi, L. Akoglu, P.I. Sánchez and E. Müller: *Focused clustering and outlier detection in large attributed graphs*. In *SIGKDD*. (2014) Pages 1346–1355.

[15] B. Perozzi, R. Al-Rfou and S. Skiena: *Deepwalk: Online learning of social representations*. In *SIGKDD*. ACM (2014) Pages 701–710.

[16] B.W. Silverman: *Density estimation for statistics and data analysis*, Volume 26. CRC press, 1986.

[17] C. Böhm, C. Faloutsos and C. Plant: *Outlier-robust clustering using independent components*. In *SIGMOD*. (2008) Pages 185–198.

[18] C.M. Bishop: *Pattern recognition and Machine Learning*. In *Springer*. (2006).

[19] C.J.C. Burges: *A Tutorial on Support Vector Machines for Pattern Recognition*. In *Data Min. Knowl. Discov.*, Volume 2. (1998) Pages 121–167.

[20] C.D. Correa and P. Lindstrom: *Locally-scaled spectral clustering using empty region graphs*. In *SIGKDD*. (2012) Pages 1330–1338.

[21] C. Cortes and V. Vapnik: *Support-Vector Networks*. In *Machine Learning*, Volume 20. (1995) Pages 273–297.

[22] C. Fowlkes, S. Belongie, F. Chung and J. Malik: *Spectral grouping using the Nystrom method*. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Volume 26. (2004) Pages 214–225.

[23] D. Liben-Nowell and J. Kleinberg, *journal of the Association for Information Science and Technology* **58** (2007), 1019.

[24] D. Niu, J.G. Dy and M.I. Jordan: *Multiple non-redandant spectral clustering views*. In *ICML*. (2010) Pages 831–838.

[25] D. Wagner and F. Wagner: *Between min cut and graph bisection*. In *International Symposium on Mathematical Foandations of Computer Science*. Springer (1993) Pages 744–750.

[26] D. Yan, L. Huang and M.I. Jordan: *Fast approximate spectral clustering*. In *SIGKDD*. (2009) Pages 907–916.

[27] D. Zhou, O. Bousquet, T.N. Lal, J. Weston and B. Schölkopf: *Learning with Local and Global Consistency*. In *NIPS*. (2003) Pages 321–328.

[28] E. Achtert, S. Goldhofer, H.P. Kriegel, E. Schubert and A. Zimek: *Evaluation of clusterings–metrics and visual support*. In *ICDE*. (2012) Pages 1285–1288.

[29] E. Bae and J. Bailey: *Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity*. In *ICDM*. (2006) Pages 53–62.

[30] E.G. Learned-Miller et al. : *ICA using spacings estimates of entropy*. In *The Journal of Machine Learning Research*, Volume 4. (2003) Pages 1271–1295.

[31] E. Müller, I. Assent, S. Günnemann, R. Krieger and T. Seidl: *Relevant subspace clustering: Mining the most interesting non-redandant concepts in high dimensional data*. In *ICDM*. (2009) Pages 377–386.

[32] E. Müller, S. Günnemann, I. Assent and T. Seidl: *Evaluating clustering in subspace projections of high dimensional data*. In *Proceedings of the VLDB Endowment*, Volume 2. (2009) Pages 1270–1281.

[33] E. Müller, P.I. Sánchez, Y. Mülle and K. Böhm: *Ranking outlier nodes in subspaces of attributed graphs*. In *ICDEW*. IEEE (2013) Pages 216–222.

[34] F.R. Bach and M.I. Jordan: *Kernel independent component analysis*. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '03, Hong Kong, April 6-10, 2003*. (2003) Pages 876–879.

[35] F. Chung: *The heat kernel as the pagerank of a graph*. In *Proceedings of the National Academy of Sciences*, Volume 104. National Acad Sciences (2007) Pages 19 735–19 740.

[36] F. Lin: *Scalable methods for graph-based unsupervised and semi-supervised learning*. Carnegie Mellon University, Dissertation, 2012.

[37] F. Lin and W.W. Cohen: *Power Iteration Clustering*. In *ICML*. (2010) Pages 655–662.

[38] F. Lin and W.W. Cohen: *Semi-supervised classification of network data using very few labels*. In *ASONAM*. (2010) Pages 192–199.

[39] F. Lin and W.W. Cohen: *A Very Fast Method for Clustering Big Text Datasets*. In *ECAI*. (2010) Pages 303–308.

[40] F. Moser, R. Colak, A. Rafiey and M. Ester: *Mining Cohesive Patterns from Graphs with Feature Vectors*. In *SDM*. (2009) Pages 593–604.

[41] F. Provost, C. Perlich and S.A. Macskassy: *Relational learning problems and simple models*. In *IJCAI*, Volume 11. Citeseer (2003).

[42] F. Wang and C. Zhang: *Label propagation through linear neighborhoods*. In *ICML*. (2006) Pages 985–992.

[43] G. Moise and J. Sander: *Finding non-redandant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering*. In *SIGKDD*. (2008) Pages 533–541.

[44] H. Abdi and L.J. Williams: *Principal component analysis*. In *Wiley Interdisciplinary Reviews: Computational Statistics*, Volume 2. Wiley Online Library (2010) Pages 433–459.

[45] H. Huang, S. Yoo, D. Yu and H. Qin: *Diverse Power Iteration Embeddings and Its Applications*. In *ICDM*. (2014) Pages 200–209.

[46] H.P. Kriegel, P. Kröger and A. Zimek: *Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering*. In *TKDD*, Volume 3. ACM (2009) Seite 1.

[47] I. Assent, R. Krieger, E. Müller and T. Seidl: *INSCY: Indexing subspace clusters with in-process-removal of redandancy*. In *ICDM*. (2008) Pages 719–724.

[48] J.M. Geusebroek, G.J. Burghouts and A.W. Smeulders: *The Amsterdam library of object images*. In *International Journal of Computer Vision*, Volume 61. Springer (2005) Pages 103–112.

[49] J. Han, J. Pei and M. Kamber: *Data mining: concepts and techniques*. Elsevier, 2011.

[50] J. Han, J.R. Wen and J. Pei: *Within-network classification using radius-constrained neighborhood patterns*. In *CIKM*. (2014) Pages 1539–1548.

[51] J.A. Hartigan and P. Hartigan: *The dip test of unimodality.* In *The Annals of Statistics.* JSTOR (1985) Pages 70–84.

[52] J. Hu, Q. Qian, J. Pei, R. Jin and S. Zhu: *Finding Multiple Stable Clusterings.* In *ICDM.* IEEE (2015) Pages 171–180.

[53] J. Moody: *Race, school integration, and friendship segregation in America.* In *American journal of Sociology.* Volume 107. (2001) Pages 679–716.

[54] J. Platt et al. : *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods.* In *Advances in large margin classifiers*, Volume 10. Cambridge, MA (1999) Pages 61–74.

[55] J. Rissanen: *Information and complexity in statistical modeling.* Springer Science & Business Media, 2007.

[56] J. Shi and J. Malik: *Normalized cuts and image segmentation.* In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Volume 22. (2000) Pages 888–905.

[57] K.W. Chang, C.J. Hsieh and C.J. Lin: *Coordinate descent method for large-scale l2-loss linear support vector machines.* In *Journal of Machine Learning Research*, Volume 9. (2008) Pages 1369–1398.

[58] K. Kloster and D.F. Gleich: *Heat kernel based community detection.* In *SIGKDD.* ACM (2014) Pages 1386–1395.

[59] K. Murphy: *Machine learning: a probabilistic perspective.* MIT press, 2012.

[60] L. Akoglu, H. Tong, B. Meeder and C. Faloutsos: *PICS: Parameter-free Identification of Cohesive Subgroups in Large Attributed Graphs.* In *SDM.* SIAM (2012) Pages 439–450.

[61] L. Getoor: *Introduction to statistical relational learning.* MIT press, 2007.

[62] L. Hubert and P. Arabie: *Comparing partitions*. In *Journal of classification*, Volume 2. Springer (1985) Pages 193–218.

[63] L.V.D. Maaten and G. Hinton, *Journal of Machine Learning Research* **9** (2008), 2579.

[64] L. Tang and H. Liu: *Relational learning via latent social dimensions*. In *SIGKDD*. ACM (2009) Pages 817–826.

[65] L. Tang and H. Liu: *Scalable learning of collective behavior based on sparse social dimensions*. In *CIKM*. ACM (2009) Pages 1107–1116.

[66] L. Zelnik-Manor and P. Perona: *Self-tuning spectral clustering*. In *Advances in neural information processing systems*. (2004) Pages 1601–1608.

[67] M. Belkin, P. Niyogi and V. Sindhwani: *Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples*. In *Journal of Machine Learning Research*, Volume 7. (2006) Pages 2399–2434.

[68] M. McPherson, L. Smith-Lovin and J.M. Cook: *Birds of a feather: Homophily in social networks*. In *Annual review of sociology*. Volume 27, (2001), Pages 415–444.

[69] M. Meila and J. Shi: *A random walks view of spectral segmentation*. In *Citeseer* (2001).

[70] M. Shiga, I. Takigawa and H. Mamitsuka: *A spectral clustering approach to optimally combining numericalvectors with a modular network*. In *SIGKDD*. ACM (2007) Pages 647–656.

[71] N.D. Thang, Y.K. Lee, S. Lee et al. : *Deflation-based power iteration clustering*. In *Applied intelligence*, Volume 39. (2013) Pages 367–385.

[72] N.X. Vinh and J. Epps: *minCEntropy: a novel information theoretic approach for the generation of alternative clusterings*. In *ICDM*. (2010) Pages 521–530.

[73] N.X. Vinh, J. Epps and J. Bailey: *Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance*. In *The Journal of Machine Learning Research*, Volume 11. (2010) Pages 2837–2854.

[74] O.L. Mangasarian: *A finite Newton method for classification*. In *Optimization Methods and Software*, Volume 17. Taylor & Francis (2002) Pages 913–929.

[75] P. Indyk and R. Motwani: *Approximate nearest neighbors: towards removing the curse of dimensionality*. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM (1998) Pages 604–613.

[76] P.F. Lazarsfeld, R.K. Merton et al. , *Freedom and control in modern society* **18** (1954), 18.

[77] P.I. Sánchez, E. Müller, K. Böhm, A. Kappes, T. Hartmann and D. Wagner: *Efficient algorithms for a robust modularity-driven clustering of attributed graphs*. In *SDM*, Volume 15. SIAM (2015).

[78] P.I. Sánchez, E. Müller, F. Laforet, F. Keller and K. Böhm: *Statistical Selection of Congruent Subspaces for Mining Attributed Graphs*. In *ICDM*. (2013) Pages 647–656.

[79] Q. Yang and X. Wu, *International Journal of Information Technology & Decision Making* **5** (2006), 597.

[80] R.E. Fan and C.J. Lin: *A study on threshold selection for multi-label classification*. In *Department of Computer Science, National Taiwan University*. (2007) Pages 1–23.

[81] R. Fan, K. Chang, C. Hsieh, X. Wang and C. Lin: *LIBLINEAR: A library for large linear classification*. In *Journal of machine learning research*. Volume 9. (2008) Pages 1871–1874.

[82] S. Alvarez and M. Vanrell: *Texton theory revisited: A bag-of-words approach to combine textons*. In *Pattern Recognition*, Volume 45. Elsevier (2012) Pages 4312–4325.

[83] S. Bhagat, G. Cormode and S. Muthukrishnan. In *Social network data analytics*. Springer (2011), Pages 115–148.

[84] S. Günnemann, I. Färber, B. Boden and T. Seidl: *Subspace Clustering Meets Dense Subgraph Mining: A Synthesis of Two Paradigms*. In *ICDM*. (2010) Pages 845–850.

[85] S. Günnemann, I. Färber, S. Raubach and T. Seidl: *Spectral Subspace Clustering for Graphs with Feature Vectors*. In *ICDM*. (2013) Pages 231–240.

[86] S. Günnemann, I. Färber, M. Rüdiger and T. Seidl: *SMVC: semi-supervised multi-view clustering in subspace projections*. In *SIGKDD*. (2014) Pages 253–262.

[87] S. Günnemann, I. Färber and T. Seidl: *Multi-view clustering using mixture models in subspace projections*. In *SIGKDD*. (2012) Pages 132–140.

[88] S. Kirshner and B. Póczos: *ICA and ISA using Schweizer-Wolff measure of dependence*. In *ICML*. (2008) Pages 464–471.

[89] S.A. Macskassy and F. Provost. *A simple relational classifier*. Technischer Bericht, DTIC Document, 2003.

[90] S.A. Macskassy and F. Provost: *Classification in networked data: A toolkit and a univariate case study*. In *Journal of Machine Learning Research*, Volume 8. (2007) Pages 935–983.

[91] S. Nandanwar and M.N. Murty: *Structural Neighborhood Based Classification of Nodes in a Network*. In *SIGKDD*. (2016) Pages 1085–1094.

[92] S. Shalev-Shwartz, Y. Singer and N. Srebro: *Pegasos: Primal estimated sub-gradient solver for svm*. In *ICML*. ACM (2007) Pages 807–814.

[93] T. Joachims: *Transductive Inference for Text Classification using Support Vector Machines*. In *ICML*. (1999) Pages 200–209.

[94] T. Xiang and S. Gong: *Spectral clustering with eigenvector selection*. In *Pattern Recognition*, Volume 41. (2008) Pages 1012–1029.

[95] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth: *From data mining to knowledge discovery in databases*. In *AI magazine*. Volume 17. (1996) Pages 37–52.

[96] V.N. Vapnik and V. Vapnik: *Statistical learning theory*, Volume 1. Wiley New York, 1998.

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

[98] W. Ye, L. Zhou, D. Mautz, C. Plant and C. Böhm: *Learning from Labeled and Unlabeled Vertices in Networks*. In *SIGKDD*. ACM (2017).

[99] W. Ye, L. Zhou, X. Sun, C. Plant and C. Böhm: *Attributed Graph Clustering with Unimodal Normalized Cut*. In *ECML-PKDD*. Springer (2017).

[100] W. Ye, S. Goebl, C. Plant and C. Böhm: *FUSE: Full Spectral Clustering*. In *SIGKDD*. ACM (2016) Pages 1985–1994.

[101] W. Ye, S. Maurus, N. Hubig and C. Plant: *Generalized Independent Subspace Clustering*. In *ICDM*. IEEE (2016) Pages 569–578.

[102] W.W. Zachary: *An information flow model for conflict and fission in small groups*. In *Journal of anthropological research*, Volume 33. University of New Mexico (1977) Pages 452–473.

[103] X.H. Dang and J. Bailey: *A hierarchical information theoretic technique for the discovery of non linear alternative clusterings*. In *SIGKDD*. (2010) Pages 573–582.

[104] X. Wang and G. Sukthankar: *Multi-label relational neighbor classification using social context features*. In *SIGKDD*. ACM (2013) Pages 464–472.

[105] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick and J. Han: *Personalized entity recommendation: A heterogeneous information network approach.* In *Proceedings of the 7th ACM international conference on Web search and data mining.* ACM (2014) Pages 283–292.

[106] Y. Cui, X.Z. Fern and J.G. Dy: *Non-redandant multi-view clustering via orthogonalization.* In *ICDM.* (2007) Pages 133–142.

[107] Y. Zhou, H. Cheng and J.X. Yu: *Graph Clustering Based on Structural/Attribute Similarities.* In *PVLDB*, Volume 2. (2009) Pages 718–729.

[108] Z. Li, J. Liu, S. Chen and X. Tang: *Noise robust spectral clustering.* In *ICCV.* (2007) Pages 1–8.

[109] Z. Szabó, B. Póczos and A. Lörincz: *andercomplete Blind Subspace Deconvolution.* In *Journal of Machine Learning Research*, Volume 8. (2007) Pages 1063–1095.

[110] Z. Szabó, B. Póczos and A. Lőrincz: *Separation theorem for independent subspace analysis and its consequences.* In *Pattern Recognition*, Volume 45. (2012) Pages 1782–1791.

[111] Z. Xu, Y. Ke, Y. Wang, H. Cheng and J. Cheng: *A model-based approach to attributed graph clustering.* In *SIGMOD.* (2012) Pages 505–516.

Acknowledgements

I am greatly indebted to my supervisor, Professor Christian Böhm, for his thoughtful advices on this thesis, endless support during my study in Germany, flexibility and understanding. You never push me but encourage me and teach me how to do research, how to do presentation and how to do review-feedback. Your rigorous, patient and optimistic attitude in research tells me the characristics a researcher should have. In addition, you provide with me the opportunity to supervise student to get experience for my future research career. Thank you very much. I will never forget the fruitful discussions with you before the conference deadline. Your thought is sharp and always inspires me. It was a pleasure to be a member in your research group: Data Mining in Medicine. I am also greatly indebted to Professor Claudia Plant, for her patient supervision in the first two years of my PhD study. You led me to the research field of data mining. You offered me the opportunities to cooperate with other students and researchers. You taught me how to write a scientific paper. You showed me how to balance the work and life. I still remember you talked with me about your feedback on my KDD manuscript after 19:00 o'clock. I especially thank you for your thoughtful and constructive advices on improving all my manuscripts. I also thank you for providing me with a good research enviroment in Helmholtz Zentrum München and the opportunity to visit your data mining group at University of Vienna, Austria. To Professor Michael Ewers at Klinikum der Universität München, it was a pleasure to cooperate with you on the project of Alzheimer's disease. You offered me a good research topic. I will never forget this precious research experience and time in Germany. I would also like to thank Professor Ambuj Singh and Professor

Heinrich Hußmann, for their interests in my work and their willingness to act as the second reviewer on this thesis and the chairman of the doctoral committee.

I would like to thank my colleagues and friends over the years in Germany: To Bianca Wackersreuther, Sebastian Goebl, Samuel Maurus, Nina Hubig and Dominik Mautz, it was very nice to cooperate with you on our papers and attend the premier conferences with you. You are open-minded and enthusiastic and I am very appreciated that I know you guys. To Linfei Zhou and Liang Jin, I feel lucky to be your friends. We came together to Germany to pursuit our PhD degree. When I have any problems, you can always give me some suggestions. To other colleagues Xiao He, Jing Feng, Son Mai Thai, Annika Tonch, Jinyi Ren and Sahar Behzadi Soheil, thank you very much for your advice and help. All of you played positive roles in my development towards a PhD degree. To Susanne Grienberger, Sandra Mayer and Franz Krojer, I thank you for your kindly background and technical supports. I would also like to thank China Scholarship Council and Ludwig-Maximilians-Universität München, Munich, for offering me the financial support (CSC-LMU joint Scholarship) for my PhD study in Germany.

Last but not least, I would like to express my deepest gratitude to my parents, brother and girlfriend. You are amazing, encouraging, brave and persistent. You are always be there supporting me, encouraging me and advising me when there was a failure. You are always be there celebrating with me when there was a success. Without your endless love, understanding, encouragement and support, this thesis would never have seen its conclusion.

Curriculum Vitae

EDUCATION

Ludwig-Maximilians-Universität München — Munich, Germany
Ph. D. in Computer Science — *09.2013-01.2018*

Shanghai University — Shanghai, China
M. Eng. in Control Science and Engineering — *09.2010-04.2013*

Shanghai University — Shanghai, China
B. Eng. in Automation — *09.2006-06.2010*

RESEARCH INTERESTS

data mining and machine learning. Specifically, clustering, semi-supervised learning and graph mining.

ACADEMIC HONORS

- KDD 2017 Student Travel Award, ACM SIGKDD and NSF, 2017.
- ACM student scholarship for the 50th celebration of Turing Awards.
- DAAD Conference Travel Award, German Academic Exchange Service (DAAD), 2016.
- ICDM 2016 Student Travel Award, IEEE TCII, 2016.

- KDD 2016 Student Travel Award, ACM SIGKDD and NSF, 2016.
- Excellent Master Degree Thesis, Committee of Education, Shanghai, 2015.
- Graduate National Scholarship, Ministry of Education, China, 2012.
- Second Prize in the 8th National Post-Graduate Mathematical Contest in Modeling, National Mathematical Modeling Contest Committee, 2011.
- National Scholarship for Encouragement, Ministry of Education, China, 2009.
- National Scholarship for Encouragement, Ministry of Education, China, 2008.
- National Scholarship for Encouragement, Ministry of Education, China, 2007.

PROFESSIONAL ACTIVITIES

- Guest PhD student at Integrative Knowledge Discovery and Data Mining Group, Helmholtz Zentrum Munich, Germany, 10.2013-12.2015.
- Visiting student at Signal and Image Processing Group, Instituto Superior Técnico, Lisbon, Portugal, 09.2012-12.2012.
- External reviewer at international conferences KDD 14, KDD 15, KDD 16, KDD 17 and IEEE BigData Conference.
- TKDD journal reviewer.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Ye, Wei

Name, Vorname

München, 06.07.2017

Ort, Datum

Unterschrift Doktorand/in

Formular 3.2