Sparse Graph Representation and Its Applications

A Dissertation presented

by

Shuchu Han

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2017

(include this copyright page only if you are selecting copyright through ProQuest, which is optional)

Copyright by
Shuchu Han
2017

Stony Brook University

The Graduate School

Shuchu Han

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation

Hong Qin - Dissertation Advisor
Professor, Department of Computer Science

Fusheng Wang - Chairperson of Defense
Assistant Professor, Department of Computer Science

Dimitris Samaras
Associate Professor, Department of Computer Science

Francesco Orabona
Assistant Professor, Department of Computer Science

Robert J. Harrison
Professor, Director of Institute for Advanced Computational Science

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Sparse Graph Representation and Its Applications

by

Shuchu Han

Doctor of Philosophy

in

Computer Science

Stony Brook University

2017

The structure of real-world data (in the form of feature matrix) includes crucial information relevant to the performance of machine learning and data mining algorithms. The structure could be local manifold structure, global structure or discriminative information based on the requirements of learning or mining tasks. To model this intrinsic structure, an effective graph representation like k-nearest neighbor graph is necessary. Considering the increasing data size in this digital era, efficient sparse graph representations without parameter tuning are very demanding.

In this thesis, we build novel sparse and nonparametric graph representation algorithms for unsupervised learning. The theory foundation of our research works is the similarity graph of Sparse Subspace Clustering. Our research works focus on: (1) alleviate the negative impacts of losing subspace structure assumption about the data: remove non-local edges and generate consistent edge connections, (2) solve the scalability issue for large size data: apply greedy algorithm with ranked dictionaries, (3) applications in unsupervised learning: redundant feature removal for high dimensional data.

Moreover, this thesis includes graph structure analysis which connects to the quality of graph following Dense Subgraph theory: (1) data label estimation through dense subgraphs for semi-supervised learning, (2) graph robustness which can differentiate the topology and scale of subgraphs.

To my wife, Ying, and my family, for their endless love and support.

Contents

List of Figures

List of Tables

Acknowledgements

Before the completion of this thesis, I received tremendous help and support from many individuals, to whom I want to express my sincere gratitude here.

I would like to express my sincere gratitude to my thesis advisor, Hong Qin, for being an excellent mentor during my studies. He showed such kindness and patience that I can not imagine myself completing this dissertation without his inspiration, encouragement and guidance.

I am grateful to my thesis committee members Fusheng Wang, Dimitris Samaras, Francesco Orabona and Robert Harrison for their time and effort in providing me with invaluable feedback in putting together and improving my thesis. I particularly thank Francesco for giving me many suggestions and feedbacks about my thesis.

I am honored to have this unique opportunity of studying at Stony Brook University on this dissertation. I am grateful to many professors from whom I have learned invaluable new knowledge and numerous skills, including: Steven Skiena, Leman Akoglu, Francesco Orabona, Mike Ferdman, Ker-I Ko, David (Xianfeng) Gu and Himanshu Gupta. I am particularity indebted to Steve, who advised me on algorithm and social science research at Yahoo Labs. I cannot thank Steve enough for his advising, help and life wisdom shared with me. I also feel extremely fortunate to work with Leman, for teaching me graph mining, showing me how to approach a problem and how to precisely formulate and write it down, as well as mentoring me to accomplish high quality research. Moreover, I want to thank Cynthia Scalzo for helping life run smoothly at SBU, and handling all administrative work seamlessly.

My internship at Yahoo Labs gives me the opportunity to work closely with top researchers in industry. I wholeheartedly thank my mentor Yifan Hu, for offering me this chance and advising me on data mining and scientific visualization research. I also thank my friends at Yahoo Labs for their kindness to me, including: Baris Coskun, Meizhu Liu, Francesco Orabona, Guy Halawi, Ruggiero Cavallo, Alina Beygelzimer, David Pal, Zohar Karnin, Justin Thaler, Edo Liberty, Maxim Sviridenko, Joel Tetreaul and Amanda Stent. The time I spent on the 14th floor is an indelible memory in my life.

I would also like to thank Computational Science Center of Brookhaven National Lab for funding my PhD research. I thank Dantong Yu for offering me a job which supports my study. I met many great colleagues over there, including: David Stampf, Shinjae Yoo, Yan Li, Wei Xu, Dimitri Katramantos, Nicholas D'lmperio, Mike McGuigan, Lauri Peragine and Robert Harrison. I also thank Rafael Perez and Robert Riccobono from Information Technology Division, for those days we spent together at the noisy data center.

It has also been a great pleasure knowing many friends at Stony Brook University, especially Hao Huang, Ming Zhong, Ming Chen, Chia-Che Tsai, William (Bill) Jannen, Junting Ye, Rui Shi, Yufen Ren, Jin Xu, Li Shi, Zhenzhou Peng, Tan Li, Shun Yao and Cheng Chang. I particularity thank Hao Huang for sharing his knowledge and experience of machine learning research with me.

Before join Stony Brook University, I was introduced to the joys of research by several advisors, including: Ying He and Changyun Wen from Nanyang Technological University, and Chengjing Zhang from Shandong University. Without their support and advising, I definitely cannot move so far in my research career.

Last, but certainly not least, I am grateful to my family. They have been so selfless in supporting me at all stages of my life. Particularly, I thank my wife, Ying Zhou, who has sacrificed so much for me. This dissertation is dedicated to them.

Publications

- **Shuchu Han**, Yifan Hu, Steven Skiena, Baris Coskun, Meizhu Liu and Hong Qin. "Capturing Properties of Names with Distributed Representations", (in preparation)
- **Shuchu Han**, Hao Huang, Hong Qin, "Automatically Redundant Features Removal for Unsupervised Feature Selection via Sparse Feature Graph". (submitted to ACM TKDD 2017)
- Junting Ye, **Shuchu Han**, Yifan Hu, Baris Coskun, Meizhu Liu, Hong Qin and Steven Skiena, "Nationality Classification using Name Embeddings". (submitted to ACM KDD 2017)
- **Shuchu Han**,Yifan Hu, Steven Skiena, Baris Coskun, Meizhu Liu. "Generating Look-alike Names via Distributed Representations". Yahoo Tech Pulse, 2016.
- **Shuchu Han**, Hong Qin. "A Greedy Algorithm to Construct Sparse Graph by Using Ranked Dictionary." International Journal of Data Science and Analytics, 2016.
- **Shuchu Han**, Hong Qin. "A Greedy Algorithm to Construct L1 Graph with Ranked Dictionary." Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2016.
- **Shuchu Han**, Hong Qin. "Structure Aware L1 Graph for Data Clustering." Thirtieth AAAI Conference on Artificial Intelligence. 2016. (student abstract)
- Chan, Hau, **Shuchu Han**, Leman Akoglu. "Where graph topology matters: the robust subgraph problem." Proceedings of the 2015 SIAM international conference on data mining, SDM. Vol. 15. 2015.(Best Research Paper Award)

- **Shuchu Han**, Hao Huang, Hong Qin. "Locality-preserving l1-graph and its application in clustering." Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, 2015.
- **Shuchu Han**, Hong Qin, and Dantong Yu. "An Improved Ratio-Based (IRB) Batch Effects Removal Algorithm for Cancer Data in a Co-Analysis Framework." Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference on. IEEE, 2014. (Best Student Paper Award)

Chapter 1

Introduction

1.1 Problem Statement

Graph-based algorithms have played an important role in machine learning and data mining research, for example, semi-supervised learning, transductive learning, spectral clustering and unsupervised feature selection. All of them require a graph representation which models the structure of data as input. This can be illustrated by Figure 1.1. How to generate a quality graph representation from the input data is still an open problem and remains to be solved [7]. This challenge is caused by the lack of theory support [8] and very few well accepted metrics to measure the quality [9]. Moreover, in most applications, graphs are constructed based on the user's own experience and judgment after considering the goal of learning and mining tasks. As a result, most graph representations are very arbitrary, and the quality of them is not guaranteed. With this observation, we find out that the quality of graph representation becomes the performance bottleneck for many machine learning and data mining algorithms.

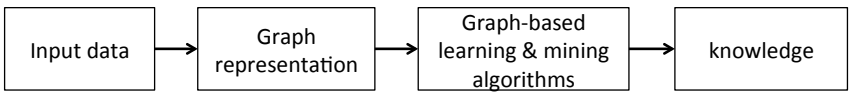


Figure 1.1: The framework of this research work. The problem we are solving is in the second block from left.

In general, the process of graph construction includes two steps: (1) define a distance metric for data vectors (we assume data samples are represented by real value data vectors in this thesis). (2) define a rule to generate edges (or which connects which in plain language). Based on the type of input data, existing graph representation methods can be classified into two groups:

(1) labeled data methods and (2) unlabeled data methods. For labeled data, the distance metric, or weight of edges, will be learned from data, including: information-theoretic metric learning (ITML) [10], large margin nearest neighbor (LMNN) [11], inference driven metric learning (IDML) [12], linear neighborhood [13], regular graph with b-matching [14], fitting a graph to vector data [15] and graph kernel [16]. For unlabeled data, global neighborhood methods are used, for example, k-nearest neighbor (kNN) graph, ϵ-ball graph, kernel graph, empty region graph [17], relative neighbor graph [18], Gabriel graph [19], β-skeletons graph [20], σ-local graph [21], $\mathcal{L}_1$ graph [22] and etc.

In this thesis, we study the problem of **how to represent the structure of unlabeled data with sparse graph**. Ideally, we hope our new graph generation algorithms could have the following three properties: (1) sparsity: for computational efficiency. (2) scalability: for big data. (3) accuracy: for exploring the structure of data. To satisfy these requirements, $\mathcal{L}_1$ graph becomes our best candidate as it is born with sparsity naturally and robustness to data noise. $\mathcal{L}_1$ graph is proposed by Cheng et al. [22] and attracts much attention of researchers in computer vision research. It seeks a sparse linear reconstruction of each data vector with other data vectors by exploiting the sparse property of lasso penalty [23]. In theory, $\mathcal{L}_1$-graph is the similarity graph of sparse subspace clustering (SCC) [24] [25]. It is constructed on a modified sparse representation framework [26], and based on a group of mixed theories including sparse linear representation algorithms from statistical signal processing community [27] [28] [29] [30] and compressive sensing [31] from signal processing research.

The construction of $\mathcal{L}_1$ graph includes n times of optimization processes, where the value n equals to the number of data samples (vectors) in input data. Given data: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}] \in \mathbb{R}^{d \times n}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, the optimization process of $\mathcal{L}_1$ graph is:

$$\min_{\boldsymbol{\alpha_i}} \|\boldsymbol{\alpha_i}\|_1 \text{ subject to } \boldsymbol{x_i} = \boldsymbol{\Phi}^i \boldsymbol{\alpha_i}, \tag{1.1}$$

where dictionary $\boldsymbol{\Phi^i} = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_{i-1}}, \boldsymbol{x_{i+1}}, \cdots, \boldsymbol{x_n}]$, $\boldsymbol{\alpha_i} \in \mathbb{R}^{n-1}$ is the sparse code of $\boldsymbol{x_i}$ and ϵ_i is the approximation error. These sparse codes are the edge weights of resulted $\mathcal{L}_1$ graph. As we can see from minimization (1.1), the neighbors of vertex $\boldsymbol{x_i}$ are sparse as a result of ℓ_1 norm constraint. Another observation is that the minimization (1.1) looks for a linear construction of $\boldsymbol{x_i}$ by using all other data vectors. This phenomenon is called "data self-representation". One advantage of this is that the neighborhood of each datum will adapt to the data structure itself.

1.2 Research Challenges

The original $\mathcal{L}_1$ graph is the similarity graph of sparse subspace clustering algorithm. It claims to have sparsity character and a nonparametric graph generation algorithm. Several advantages of $\mathcal{L}_1$ graph are [22]: (1) Robustness to data noise comparing to graphs that are constructed by using pair-wise distance, such as kNN graph, (2) Sparsity, and (3) Datum-adaptive neighborhood. The success of $\mathcal{L}_1$ graph requires the input data to have *subspace structure*. Several type of data like image data or rigid motion data may satisfy this requirement but other types may not. Since we lose this subspace assumption, the constructed sparse graph may include lots of meaningless edge connections (or linear construction).

Recently, several challenges of $\mathcal{L}_1$ graph when applying it to general data are discussed, there are:

1. Existence of non-local edge connections. The local manifold structure of data is ignored [32] [1] as it only captures subspace structure.

2. Lack of scalability by its high computational cost [22]. As we can see from Equation (1.1), for each data vector, it solves a ℓ_1-norm minimization problem which is an iterative optimization process and very time consuming.

3. Inconsistent edge connections and edge weights. While calculating the sparse representation for each data vector, ℓ_1-minimization solver will pick one representation (atom) randomly if the dictionary exists a group of highly correlated atoms (data vectors) [33]. Moreover, If there are duplicate data vectors, the solver will return only one edge connection to one of its duplications [22].

4. The success of $\mathcal{L}_1$ graph is based on the assumption that data has subspace structure. For data without this assumption, the linear sparse representation (reconstruction) returned from ℓ_1-minimization solver is wrong and the edge connections and weights are meaningless. As a result, noisy edge connections will exist in the generated graph.

1.3 Research Contributions

In this dissertation, we present novel sparse graph representations to model the structure of data. Our proposed algorithms don't make any assumption about the input data comparing to the original $\mathcal{L}_1$ graph which requires the

data to have subspace structure. Particularly, the contributions of this thesis are summarized as follows:

1. We first alleviate the existing of non-local edges problem by limiting the dictionary of sparse representation to its nearest neighbors under Euclidean distance. With this "hard" constraint, the edge connections are forced to occur within local neighbors. Our observation from experiment results shows that the locality of data is well preserved by adding this constraint. Moreover, with a small-sized dictionary, the construction of $\mathcal{L}_1$ graph becomes more efficient. However, we bring an additional parameter about the size of dictionary into original $\mathcal{L}_1$ graph construction.

2. Selecting dictionary locally based on Euclidean distance is suitable for data that has convex cluster boundary. However, for data with non-convex cluster shapes, Euclidean distance has a risk to bring data vectors (atoms) belonging to other clusters into the current dictionary for sparse coding. Here, the manifold structure of data becomes critical. We then propose diffusion distance to capture the geometry shape of input data. This structural aware approach is proved to be very efficient for clustering data with explicit non-convex geometry shapes.

3. Scalability is an urgent and not yet solved problem for original $\mathcal{L}_1$ graph construction algorithm as it involves many (linearly increasing with data size) optimization (sparse coding) processes which are time consuming. With recent research works in subspace learning about greedy ℓ_1 minimization solver, we propose a greedy algorithm based on orthogonal Matching Pursuit (OMP) solver and ranked dictionaries to accelerate the construction of $\mathcal{L}_1$ graph. The advantages of our algorithm is that it not only speeds up the construction but also solves the inconsistent edge connection problem.

4. We also invest our research effort in graph structure analysis and apply it into downstream applications. We propose a graph-based algorithm for one computational biology application. The goal is to remove Batch Effect which exists among Microarray experiment data from different sources. A sparse graph is first constructed from the data and then we use the dense subgraphs extracted from the data. Moreover, we propose robust local subgraph by using robustness as density measurement. Comparing to the dense subgraphs defined by classical edge density, the robustness metric not only can measure the difference of subgraph topologies, but also can differentiate the subgraph size.

5. We successfully apply our sparse graph representation works to high dimensional data which has more features than samples. The goal is to remove the redundant features existed in high dimensional data. The proposed *sparse feature graph* is a natural way to encode the group redundancy among features. This group redundancy is always neglected by pairwise redundancy which is more popular in machine learning research. Our research work combines the sparse graph representation and dense subgraph mining techniques, and demonstrates to be a very efficient tool for redundant feature removal.

1.4 Dissertation Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review different graph construction methods in machine learning research and the ℓ_1 minimization problem. Moreover, we review the dense subgraph mining techniques that are related to our future research on graph structure analysis. In Chapter 3, we propose an improved version of $\mathcal{L}_1$ graph with locality preserved. At the same time, we also evaluate the quality of generated graph for spectral clustering by using different distance metrics. In Chapter 4, we introduce a greedy algorithm to construct sparse graph with ranked dictionary. In Chapter 5, we present an application in computational biology by using graph algorithm to remove batch effects among Microarray experiment data from different sources. In Chapter 6, we use robustness metric to define the edge density of dense subgraphs, and a heuristic algorithm to search those robustness subgraphs. In Chapter 7, we propose sparse feature graph to model the feature redundancy existed in high dimensional data, and present a dense subgraph based approach to locating the redundant feature groups. In Chapter 8, we introduce a graph embedding research work in social science research. Finally, we conclude this thesis and outline some future research directions in Chapter 9.

Chapter 2

Background Review

Our research works are based on $\mathcal{L}_1$ graph from sparse subspace clustering, $\mathcal{L}_1$ minimization, spectral embedding and clustering and dense subgraph theory. In this chapter, we briefly review the basic ideas of related techniques and analyze their properties.

2.1 Graph Construction Methods for Similarity Measures

For graph-based learning algorithms, a graph is required to represent the similarity among data vectors (here we assume each data sample is represented by a real value data vector). The "Similarity" and "Distance" are reversed relationship: high similarity means short distance. Given a set of data vectors, and a distance metric in this vector space, a graph representation can be constructed by following a special edge construction rule. And with different rules, we have different graph construction methods. In this section, we briefly introduce several well-known graph construction methods.

Assume the input data is: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}]$, where $\boldsymbol{x_i} \in \mathbb{R}^d$, and a distance metric $d(\boldsymbol{x_i}, \boldsymbol{x_j})$ is defined over the space $\mathbb{R}^{n \times d}$, then we can construct different graph representations by following methods:

kNN graph. This graph connects each data sample to its first k nearest neighbors based on distance $d(\boldsymbol{x_i}, \boldsymbol{x_j})$.

ϵ-ball graph. This graph selects edge/no-edge between two data vectors by their distance: $d(\boldsymbol{x_i}, \boldsymbol{x_j}) \leq \epsilon$.

$\mathcal{L}_1$ graph. $\mathcal{L}_1$ graph seeks a sparse linear reconstruction for each data vector with all other data vectors by exploiting the sparse property of the Lasso penalty [23]. This is fundamentally different from the traditional ones as the edge connections and edge weights are pure numerical results form $\mathcal{L}_1$ minimization solver.

The $\mathcal{L}_1$ graph construction algorithm [22] can be described by:

Algorithm 1: $\mathcal{L}_1$-Graph

Input : Feature matrix: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}] \in \mathbb{R}^{d \times n}$, where $\boldsymbol{x_i} \in \mathbb{R}^d$.
Output: Adjacency matrix W of $\mathcal{L}_1$ graph.

1 **Normalization:** normalize each data vector $\boldsymbol{x_i}$ to has unit length: $\|\boldsymbol{x_i}\|_2 = 1$;

2 $\mathcal{L}_1$ **minimization:** For each vector $\boldsymbol{x_i}$, its sparse coding coefficients are calculate by solving the following optimization:

$$min_{\boldsymbol{\alpha}^i} \|\boldsymbol{\alpha}^i\|_1, \quad s.t. \quad \|\boldsymbol{x}_i - \boldsymbol{\Phi^i \alpha^i}\|_2 \leq \epsilon_i,$$

where matrix $\boldsymbol{\Phi^i} = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_{i-1}}, \boldsymbol{x_{i+1}}, \cdots, \boldsymbol{x_n}] \in \mathbb{R}^{d \times (n-1)}$, $\boldsymbol{\alpha^i} \in \mathbb{R}^{n-1}$ and ϵ_i is the approximation error;

3 **Graph edge weight setting:** Denote $W = (V, E)$, where V is the set of data vectors as graph vertices, and E is the set of weighted edges. We set edge weight from $\boldsymbol{x_i}$ to $\boldsymbol{x_j}$ by $\boldsymbol{\alpha}^i(j)$, where $1 \leq j \leq n, j \neq i$. (non-negativity constraints may be imposed for $\boldsymbol{\alpha}^i(j)$ in optimization if for similarity measurement). If $i < j$, edge weight of $\boldsymbol{(x_i, x_j)}$ is: $E(i, j) = \boldsymbol{\alpha}^i(j-1)$;

As we can see, for each data vector, we need to solve a ℓ_1 minimization problem. This optimization process can be solved in polynomial time by standard linear programming method.

2.2 $\mathcal{L}_1$ Minimization.

$\mathcal{L}_1$ minimization is a classical problem in optimization and signal processing communities. In compressive sensing theory, it has been shown to be an efficient approach to recover sparest solutions to certain under-determined systems of linear equations. Comparing to Equation 1.1, the more general $\mathcal{L}_1$ minimization problem solves the following convex program:

$$\min \|\boldsymbol{x}\|_1, \ \textit{subject to } \boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}, \tag{2.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{d \times n}$ is an under-determined ($d \ll$n) full-rank matrix. Assume $\boldsymbol{x}_0 \in \mathbb{R}^n$ is an unknown signal, and $\boldsymbol{b}$ is the observation of $\boldsymbol{x}_0$ through matrix $\boldsymbol{A}$, the compressive sensing theory try to discover whether the solution of Equation 2.1 can recover signal $\boldsymbol{x}_0$.

Coherence. Compressive sensing theory shows that if $\boldsymbol{x}_0$ sparse enough and the sensing matrix $\boldsymbol{A}$ is *incoherent* with the basis under which $\boldsymbol{x}_0$ is sparse, $\boldsymbol{x}_0$ can be recovered exactly [34] [28]. The sensing matrix $\boldsymbol{A}$ is also called as "Dictionary" and *coherence* [35] is used to measure the correlation among atoms of dictionary. The coherence is defined as:

$$\mu = \max_{i \neq j} | < \boldsymbol{\psi}_i, \boldsymbol{\psi}_j > |, \tag{2.2}$$

where $\boldsymbol{\psi}_\cdot$ is the column of matrix $\boldsymbol{A}$. In words, the coherence is the cosine of the acute angle between the closest pair of atoms. Informally, a dictionary is *incoherent* if the value μ is smaller than a threshold.

Minimization solvers. In practical, the equation $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$ is often relaxed to take into account the existence of measurement error in the recovering process: $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$. Particularly, if the error term $\boldsymbol{e}$ is assumed to be white noise such that $\|e\|_2 \leq \epsilon$, the ground truth signal $\boldsymbol{x}_0$ can be well approximated by the *basis pursuit denoising*(BPDN) [36].

$$\min \|\boldsymbol{x}\|_1 \text{ subject to } \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \epsilon. \tag{2.3}$$

The methods that solver the above minimization problem include but not limit to: gradient projection [37], homotopy [38], iterative shrinkage-thresholding [39], proximal gradient [40], and augmented Lagrange multiplier [41].

In our research works, we use the *truncated Newton interior-point method (TNIPM)* [37] as our optimization solver. The object function 2.3 is rewritten as below by using Lagrangian method:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \tag{2.4}$$

where λ is the Lagrangian multiplier. The TNIPM transfers the above object function into a quadratic program with inequality constraints:

$$\min \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \sum_{i=1}^{2} \boldsymbol{u}_i, \text{s.t.} -\boldsymbol{u}_i \leq \boldsymbol{x}_i \leq \boldsymbol{u}_i, \quad i = 1, \cdots, n. \tag{2.5}$$

Then a logarithmic barrier for the constraints $-\boldsymbol{u}_i \leq \boldsymbol{x}_i \leq \boldsymbol{u}_i$ can be constructed:

$$\Phi(\boldsymbol{x}, \boldsymbol{u}) = -\sum_i \log(\boldsymbol{u_i} + \boldsymbol{x_i}) - \sum_i \log(\boldsymbol{u_i} - \boldsymbol{x_i}), \tag{2.6}$$

Over the domain of $(\boldsymbol{x}, \boldsymbol{u})$, the central path consists of the unique minimizer $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ of the convex function

$$F_t(\boldsymbol{x}, \boldsymbol{u}) = t(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \sum_{i=1}^{2} \boldsymbol{u}_i) + \boldsymbol{\Phi(x, u)}, \tag{2.7}$$

where the parameter $t \in [0, \infty)$. The function can then be minimized by standard interior-point algorithms.

2.3 Spectral Embedding and Clustering

The goal of clustering is to partition data into different subsets, such that the data within each subset are similar to each other. The spectral clustering [42] algorithm show its elegant over other clustering algorithms by its ability to discover embedding data structure. Spectral clustering algorithm has strong connection with graph cut, i.e., it uses eigenspace to solve a relaxed form of the balanced graph partitioning problem [43]. It has advantage on capturing nonlinear structure of data with using nonlinear kernels, which is difficult for k-means [44] or other linear clustering algorithms. The spectral clustering algorithm can be described as following:

In the above spectral clustering algorithm 2, the affinity matrix $\boldsymbol{W}$ can be seen as a weighted undirected graph, and this graph encode the local information about the data. The weight of graph is calculated from certain similarity kernels such as Gaussian kernel. When apply $\mathcal{L}_1$ graph as the input of spectral clustering, we use a math trick: $\boldsymbol{W} = (|\boldsymbol{W}| + |\boldsymbol{W}|)/2$ to symmetrize the matrix $\boldsymbol{W}$.

2.4 Dense Subgraph

Dense subgraph problem is a fundamental research in learning the structure of graph. Given a graph $G = (V, E)$, if the edges are weighted, we use $w(u)$ to represent the weight of edge u. Unweighted graphs are the special case where all weights are equal to 1. Let S and T be subsets of V. For an undirected graph, $E(S)$ is the set of induced edges on S : $E(S) = (u, v) \in E | u, v \in S$ Then H_S is the induced subgraph $(S, E(S))$. Similarly, $E(S, T)$ designates the

Algorithm 2: SpectralClustering($\boldsymbol{X}, c$)

Input : $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ where n is #instances, m is #features, and c is #clusters.

Output: Cluster assignments of n instances.

1 Construct the affinity matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$;

2 Compute the diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ where $D(i,i) = \sum_{j=1}^{n} \boldsymbol{W}(i,j)$ and $D(i,j) = 0$ if $i \neq j$;

3 Apply the graph Laplacian $\boldsymbol{L} = \mathbb{R}^{n \times n}$ using $\boldsymbol{L}_{nn} = \boldsymbol{D} - \boldsymbol{W}$, $\boldsymbol{L}_{nn} = \boldsymbol{I} - \boldsymbol{D^{-1}W}$ or $\boldsymbol{L}_{sym} = \boldsymbol{I} - \boldsymbol{D^{-1/2}WD^{1/2}}$ where $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is an identity matrix;

4 Extract the first c nontrivial eigenvectors $\boldsymbol{\Psi}$ of $\boldsymbol{L}$, $\boldsymbol{\Psi} = \{\boldsymbol{\psi_1}, \boldsymbol{\psi_2}, \cdots, \boldsymbol{\psi_c}\}$;

5 Re-normalize the rows of $\boldsymbol{\Psi} \in \mathbb{R}^{n \times c}$ into $\boldsymbol{Y}_i(j) = \boldsymbol{\psi}_i(j)/(\sum_l \boldsymbol{\psi}_i(l)^2)^{1/2}$;

6 Run k-means with c and $\boldsymbol{Y} \in \mathbb{R}^{n \times c}$;

set of edges from S to T. $H_{S,T}$ is the induced subgraph $(S, T, E(S,T))$. S and T are not necessarily disjoint from each other.

For a subgraph S, the density $den(S)$ is defined as the ratio of the total weight of edges in $E(S)$ to the number of possible edges among $|S|$ vertices. If the graph is unweighted, then the numerator is simply the number of actual edges, and the maximum possible density is 1. if it is weighted, the maximum density is unbounded. The number of possible edges in a graph of size n is $\binom{n}{2} = n(N-1)/2$. Several **edge density** definitions are:

$$den(S) = \frac{2|E(S)|}{|S|(|S|-1)}, \tag{2.8}$$

$$den_w(S) = \frac{2 * \sum_{u,v \in S} w(u,v)}{|S|(|S|-1)}, \tag{2.9}$$

$$den_{avg}(S) = \frac{2|E(S)|}{|S|}, \tag{2.10}$$

where $den(S)$ is for unweighted graph, $den_{w(S)}$ is for weighted graph and $den_{avg}(S)$ is the average edge density for unweighted graph.

Subgraphs have different forms (or names) by considering its structure property. In the following we introduce several important forms that related to our research works.

Clique. a clique is a subgraph which all its vertices are connected to each other. A *maximum clique* of a graph is a clique having maximum size and its size is called the graph's clique number. A *maximal clique* is a clique that is not a subset of any other clique.

Densest subgraph. The *densest-subgraph* problem is to find a set S that maximizes the average degree. Finding the densest subgraph in a given graph is a P problem by solving a parametric maximum-flow problem [45]. However, if we put size restriction on $|S|$, this problem becomes **NP**-hard [46].

Quasi-clique. A set of vertices S is an α-quasi-clique if $E[S] \geq \alpha \left(\frac{|S|}{2}\right)$, i.e., if the edge density of the subgraph exceeds a threshold parameter $\alpha \in (0, 1)$.

Chapter 3

Locality-Preserving and Structure-Aware $\mathcal{L}_1$ Graphs

In this chapter, we propose two types of improved $\mathcal{L}_1$ graphs. The first one is a Locality-Preserving $\mathcal{L}_1$ graph (LOP-$\mathcal{L}_1$), which preserves higher local-connections and at the same time maintains sparsity. The second one is a structure aware $\mathcal{L}_1$ graph by encoding the intrinsic manifold structure of data. The difference with previous one is that it ranks a data point's nearest neighbors by manifold ranking score which takes the data's geometry structure into account. Comparing with original $\mathcal{L}_1$ graph and its other regularization-based versions, these two methods require less amount of running time in the scalability test. We evaluate the effectiveness of them by applying it to clustering application, which confirms that the proposed algorithms outperform related methods.

3.1 Chapter Introduction

Among many techniques used in the machine learning society, graph-based mining mainly tries to accommodate the so-called cluster-assumption, which says that samples on the same structure or manifold tend to have large weight of connections in-between. But most of the time there is no explicit model for the underlying manifolds, hence most methods approximate it by the construction of an undirected/directed graph from the observed data samples. Therefore, correctly constructing a good graph that can best capture essential data structure is critical for all graph-based methods [47].

Ideally, a good graph should reveal the intrinsic relationship between data samples on manifold, and also preserve the strong local connectivity inside neighborhood (called as locality in the following sections). Traditional meth-

ods (such as k-nearest neighbors (kNN) [48], ϵ-neighborhood [48] and Gabriel graph (GG) [49]) mainly rely on pair-wise Euclidean distances to construct the locally-connected graph. The obtained graphs oftentimes fail to capture local structures and cannot capture global structures of the manifold [47]. Besides, these methods either cannot provide datum-adaptive neighborhoods because of using fixed global parameters [49], or are sensitive to the parameter setting or local noise especially on high-dimensional datasets [50].

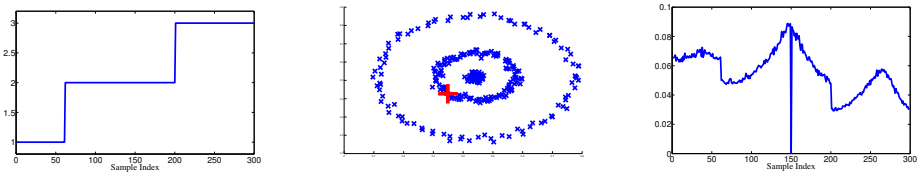
Recently, Cheng et al. [22] proposed to construct an $\mathcal{L}_1$ graph via sparse coding [26] by solving an $\mathcal{L}_1$ optimization problem. $\mathcal{L}_1$ graph is derived by encoding each datum as a sparse representation of the other samples (treated as basis or dictionary pool), and automatically selecting the most informative neighbors for each datum. The nice properties of $\mathcal{L}_1$ graph include: 1) sparsity, which leads to fast subsequent analysis and low requirement for storage [26], 2) datum-adaptive neighborhoods and 3) robustness to data noise as claimed in [22].

However, the constructing of classic $\mathcal{L}_1$ graph suffers from the loss in the locality of the samples to be encoded, which is a fundamental drawback from sparse coding [51]. Usually, the number of samples is much greater than the number of manifold dimensions, which means that the basis pool is "overcomplete" during the construction of $\mathcal{L}_1$ graph. Samples may be encoded with many basis (samples) with weak correlations with the object samples under such "overcomplete" basis pool. Thus, it results in the inaccuracy of $\mathcal{L}_1$ graph, and therefore impedes the quality of the consequent analysis tasks. As an illustration, Fig.3.1(e) shows that under classic $\mathcal{L}_1$ graph construction, the code of a sample point p (red cross in Fig.3.1(b)) involves many basis (samples) that do not belong to the same cluster with p. Such instability may hinder the robustness of the $\mathcal{L}_1$ graph based data mining applications, as shown in Fig.3.1(f). To address this issue, we propose a Locality-Preserving $\mathcal{L}_1$ graph (LOP-$\mathcal{L}_1$) to learn more discriminative sparse code and preserve the locality and the similarity of samples in the sparse coding process, and therefore the robustness of the data analysis result is enhanced. Our contributions are as follows:

1. LOP-$\mathcal{L}_1$ **preserves locality** in an datum-adaptive neighborhood, and at the same time maintains sparsity from classic $\mathcal{L}_1$.

2. The computation of LOP-$\mathcal{L}_1$ is **more scalable** than classic $\mathcal{L}_1$ graph and the succeeding regularization-based techniques.

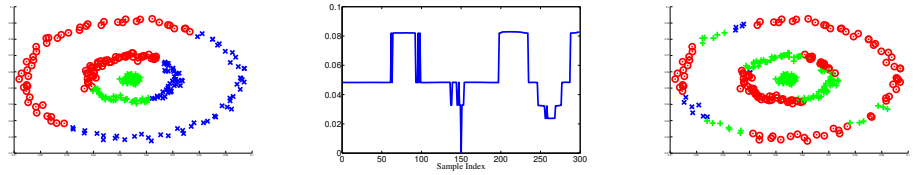
3. We confirm the **effectiveness** of LOP-$\mathcal{L}_1$ in the application of clustering.



(a) Cluster labels

(b) Original data, point p marked as red cross

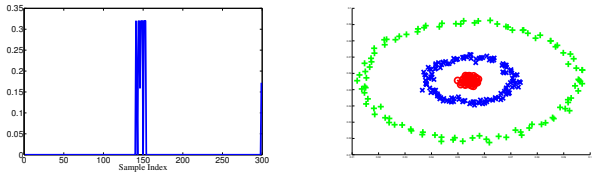
(c) Coding on Gaussian graph



(d) Clustering results on Gaussian graph

(e) Coding on $\mathcal{L}_1$ graph

(f) Clustering results on $\mathcal{L}_1$ graph



(g) Coding on LOP-$\mathcal{L}_1$

(h) Clustering results on LOP-$\mathcal{L}_1$

Figure 3.1: Illustration of LOP-$\mathcal{L}_1$ effectiveness compared with Gaussian (similarity) graph and classic $\mathcal{L}_1$. The labels of sample in the original dataset (Fig.3.1(b)) are showed in Fig.3.1(a), and in this example we only focus on the coding of point p (the 150-th sample, marked as red cross in Fig.3.1(b)). Coding (similarity) of p on Gaussian graph (Fig.3.1(c)) is built upon Euclidean space, which leads to manifold non-awareness (Fig.3.1(d)). Classic $\mathcal{L}_1$ graph coding (Fig.3.1(e)) results in the loss of locality and therefore instable clustering result (Fig.3.1(f)). Comparatively, our LOP-$\mathcal{L}_1$ coding on p (Fig.3.1(g)) shows strongly locality-preserving characteristic and has the best performance in clustering, as shown in Fig.3.1(h).

3.2 Related Works

$\mathcal{L}_1$ graph is an informative graph construction method proposed by Cheng et al. [22]. It represents the relations of one datum to other data samples by using the coefficient of its sparse coding. The original $\mathcal{L}_1$ graph construction algorithm is a nonparametric method based on the minimization of a $\mathcal{L}_1$ norm-based object function.

The advantages of $\mathcal{L}_1$ graph are summarized as follows: (1) robustness to data noise; (2) sparsity for efficiency; and (3) datum-adaptive neighborhood. Because of these virtues, $\mathcal{L}_1$ graph has been applied to many graph based learning applications [22], for example, subspace learning [22], image classification [47] and semi-supervised learning [52] etc. However, classic $\mathcal{L}_1$ graph [22] is a purely numerical solution without physical or geometric interpretation of the data set [53]. Therefore, to better exploit the structure information of data, many research works have been proposed by adding a new regularization term in addition to the original Lasso penalty, for example, the elastic net regularization [53], OSCAR regularization [53] and graph-Laplacian [3].

Another research focus of $\mathcal{L}_1$ graph is to reduce its high computational cost. For each datum, the $\mathcal{L}_1$ graph need to solve an $\mathcal{L}_1$ minimization problem within a large basis pool which is very slow. To reduce the running time, Zhou et al. [1] proposed a kNN Fused Lasso graph by using the k-nearest neighbors idea in kernel feature space. With a similar goal, Fang et al. [53] proposed an algorithm which firstly transfers the data into a reproducing kernel Hilbert space and then projects to a lower dimensional subspace. By these projections, the dimension of dataset is reduced and the computational time decreased.

In our research we evaluate the performance of different graph constructions in terms of clustering. Specifically we integrate the constructed graph into the framework of spectral clustering, due to its popularity and its ability to discover embedding data structure. Spectral clustering starts with local information encoded in a weighted graph on input data, and clusters according to the global eigenvectors of the corresponding (normalized) affinity matrix. Particularly, to satisfy the input of spectral clustering algorithm, we transform the adjacency matrix of $\mathcal{L}_1$ graph into a symmetry matrix manually.

3.3 LOP-$\mathcal{L}_1$ Graph

The construction of classic $\mathcal{L}_1$ graph [22] is a global optimization which is short of local-structure awareness. Moreover, it has a high time complexity, since for each datum it needs to solve a $\mathcal{L}_1$-minimization problem 2.3. For each sample x_i, the global optimization aims at selecting as few basis functions as possible from a large basis pool, which consists of all the other samples (basis), to linearly reconstruct x_i, meanwhile keeping the reconstruction error as small as possible. Due to an overcomplete or sufficient basis pool, similar samples can be encoded as totally different sparse codes, which may bring about the loss of locality information of the samples to be encoded. To preserve such locality information, many researches add one or several regularization terms to the object Eq. 2.3 as in [32] [1] and etc. However, there is a lack of generality for

these methods and the regularization-based approaches are, as widely known, very time consuming.

Here, we propose a much more general and concise approach, called Locality-Preserving $\mathcal{L}_1$-Graph (LOP-$\mathcal{L}_1$), by limiting the basis pool in a local neighborhood basis of the object sample. Our algorithm only uses the k nearest neighborhoods of the object sample as the basis pool, and the definition of the object function minimization is as follows:

Definition 1. *The minimizing object function of LOP-$\mathcal{L}_1$ is defined as:*

$$\min_{\boldsymbol{\alpha}_i} \|\boldsymbol{\alpha}_i\|_1, \quad s.t. \quad x_i = \boldsymbol{\Gamma^i}\boldsymbol{\alpha_i},\ \boldsymbol{\alpha_i} \geq 0, \tag{3.1}$$

where $\boldsymbol{\Gamma}^i = [\boldsymbol{x_1^i}, \boldsymbol{x_2^i}, \cdots, \boldsymbol{x_k^i}]$ is the k-nearest neighbors of $\boldsymbol{x}_i$ in the data set, with the constraint that all the elements in $\boldsymbol{\alpha}_i$ are nonnegative.

The weights of edges in the LOP-$\mathcal{L}_1$ graph are obtained by seeking a non-negative low-rank and sparse matrix that represents each data sample as a linear combination of its constrained neighborhood. The constructed graph can capture both the global mixture of subspaces structure (by the coding process) and the locally linear structure (by the sparseness brought by the constrained neighborhood) of the data, hence is both generative and discriminative. Furthermore, by introducing such a locality preserving constraint to the sparse coding process, the similarity of sparse codes between similar local samples can be preserved. Therefore, the robustness of the subsequent data analysis task (e.g. spectral clustering) is enhanced. Limiting the size of basis pool also leads to a benefit of reducing the running time of $\mathcal{L}_1$ graph construction.

The details of our proposed LOP-$\mathcal{L}_1$ is described in Algorithm 3. It is worth to point out that our proposed LOP-$\mathcal{L}_1$ doesn't prevent users to add specific regularization terms during the optimization for a special application.

In our implementation, we select one gradient-project-based method called *truncated Newton interior-point method* (TNIPM) [37] as the $\mathcal{L}_1$ minimization solver, which has $O(N^{1.2})$ empirical complexity where N is the number of samples. The $\mathcal{L}_1$-minimization object function we used is:

$$\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda\|\boldsymbol{x}\|_1, \tag{3.2}$$

where λ is the Lasso penalty parameter. We choose $\lambda = 1$ in our experiments as many methods also choose.

Analysis of Time Complexity. Here we analyze the time efficiency of LOP-$\mathcal{L}_1$ by comparing its running time with classic $\mathcal{L}_1$ graph. $\mathcal{L}_1$ graph with

Algorithm 3: LOP-$\mathcal{L}_1$-Graph

Input : Data samples $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}]$, where $\boldsymbol{x}_i \in \mathbb{R}^m$; Parameter t for scaling k-nearest neighborhood, where $k = t * m$ (check Section 3.5.1 for more details).

Output: Adjacency matrix $\boldsymbol{W}$ of $\mathcal{L}_1$ graph.

Normalize the data sample $\boldsymbol{x}_i$ with $\|\boldsymbol{x}_i\|_2 = 1$;
for $\boldsymbol{x}_i \in \boldsymbol{X}$ **do**
  Find k-nearest neighbors of x_i:$\boldsymbol{\Gamma}^i = [x_1^i, \cdots, x_k^i]$;
  Let $\boldsymbol{B}^i = [\boldsymbol{\Gamma}^i, I]$;
  Solve: $\min_{\boldsymbol{\alpha}_i} \|\boldsymbol{\alpha}_i\|_1, \quad s.t. \quad \boldsymbol{x}_i = \boldsymbol{B^i \alpha_i}$, and $\boldsymbol{\alpha_i} \geq 0$;
end
for $i = 1 : N$ **do**
  for $j = 1 : N$ **do**
    /* get the sparse code for each x_i */
    if $x_j \in \Gamma^i$ **then**
      /* $pos(x_j)$ is the position of x_j in nb^i */
      $W(i,j) = \alpha_i(pos(x_j))$
    else
      $W(i,j) = 0$
    end
  end
end

TNIPM solver has $O(N^{1.2})$ [54] empirical complexity. Our LOP-$\mathcal{L}_1$ algorithm reduces the size of basis pool from N to $k = t * m$, so the empirical complexity will be $O(Nk^{1.2})$. To demonstrate the time reduction, we test the CPU time of LOP-$\mathcal{L}_1$ and (classic) $\mathcal{L}_1$ over a series of random data sets which have 50 attributes and sample size from 10^1 to 10^4. The result is presented in Fig.2, which also proves that LOP-$\mathcal{L}_1$ can work just like the LOP-$\mathcal{L}_1$.

Analysis and Connections. We now justify the LOP-$\mathcal{L}_1$'s utility by briefly documenting its theoretic connections with a few existing methods, which also lays a solid foundation for LOP-$\mathcal{L}_1$'s attractive properties in practical use.

LOP-$\mathcal{L}_1$ vs Classic kNN-Graph. Compared with our proposed LOP-$\mathcal{L}_1$, the classic kNN graph [48] can be generated very fast, but they achieve this with a sacrifice on the quality. Classic kNN graph-based methods can be easily affected by noises, especially those samples which are not in the same structure while being very close in the misleading high-dimensional Euclidean space. The fundamental difference between classic kNN graph and our proposed LOP-$\mathcal{L}_1$ is that the former is highly dependent on the pre-specified

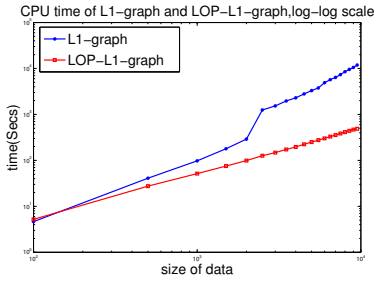


Figure 3.2: Scalability comparison between LOP-$\mathcal{L}_1$ graph and classic $\mathcal{L}_1$ graph.

sample-sample similarity measure used to identify the neighbors, whereas the later generates an advanced similarity matrix W by solving the optimization problem of Equation 3.2. In this way, W can potentially encode rich and subtle relations across instances that may not be easily captured by conventional similarity metrics. This is validated by the experimental results in Section 5 that show the LOP-$\mathcal{L}_1$ substantially outperforms classic kNN graph in clustering application.

LOP-$\mathcal{L}_1$ vs Classic $\mathcal{L}_1$-Graph. Our proposed LOP-$\mathcal{L}_1$ is built upon classic $\mathcal{L}_1$, but has unique theoretical contributions and huge improvement on performance. As we mentioned earlier, the coding process of $\mathcal{L}_1$ suffers from the "overcomplete" basis pool. The optimization of $\mathcal{L}_1$ is solved by a straightforward numerical solution: every time the $\mathcal{L}_1$-minimization picks up the basis randomly from a group of "highly similar data samples" [33]. However, if the sample dimension is high, the similarity evaluation on Euclidean space would be highly misleading, which is a well-known problem. Therefore, together with a large-size basis pool, the basis $\mathcal{L}_1$ picks up are not guaranteed to be in the same manifold with the object sample. In our proposed LOP-$\mathcal{L}_1$, we restrain the coding process from picking up those samples outside certain neighborhood. In other words, the samples/basis are locally coded, and LOP-$\mathcal{L}_1$ brings a dramatic improvement of performance and stability on the subsequent analysis step. We will further confirm this in the Experiment Section 5.

LOP-$\mathcal{L}_1$ vs Regularization-based $\mathcal{L}_1$-Graph. Specifically, the idea of our LOP-$\mathcal{L}_1$ is close to the *kNN Fused Lasso graph* proposed by Zhou et al. [1]. However, our algorithm is different at: (1) there is no regularization term in our $\mathcal{L}_1$ minimization; (2) we process the data samples at original data space instead of at kernel feature space. Generally speaking, our LOP-$\mathcal{L}_1$ is designed in a more concise and efficient way compared with the regularization-based techniques such as [32] [1].

LOP-$\mathcal{L}_1$ vs Recommender Systems and Collaborative Filtering. Similar to the linear coding used in our proposed LOP-$\mathcal{L}_1$, Paterek [55] introduced a recommender system that linearly models each item for rating prediction, in which the rating of a user u_j on an item v_k is calculated as the aggregation of the ratings of u_j on all similar items (given by kNN graph). Intuitively, in our LOP-$\mathcal{L}_1$ we can treat $W(i,j)$ as a rating of sample x_i to sample x_j, which is derived by a subset of x_i's nearest neighbors, and prediction of $W(i,j)$ is generated based on a weighted aggregate of their ratings. In other words, LOP-$\mathcal{L}_1$ realizes the concept of collaborative filtering [55] within a constraint neighborhood that brings locality-preserving property, of which advantages in recommender systems has been analyzed and confirmed in [56].

3.4 SA-$\mathcal{L}_1$ Graph

In this section, we propose a Structure Aware (SA) $\mathcal{L}_1$ graph to improve the data clustering performance by capturing the manifold structure of input data. We use a local dictionary for each datum while calculating its sparse coefficients. SA-$\mathcal{L}_1$ graph not only preserves the locality of data but also captures the geometry structure of data. The experimental results show that our new algorithm has better clustering performance than $\mathcal{L}_1$ graph.

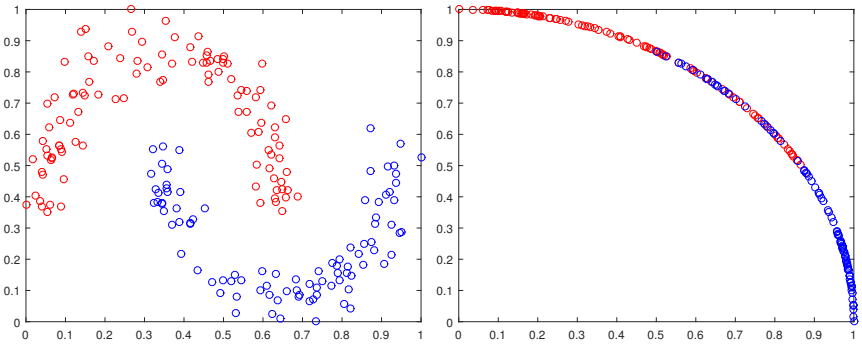


Figure 3.3: Dictionary normalization of *two moon* dataset. The red and blue points represent different clusters. Left: before normalization, right: after normalization. We can see that the neighborhood information is changed after normalization.

One less attractive aspect of $\mathcal{L}_1$ graph construction algorithm is the *normalization* of dictionary. While calculating the sparse coefficient (or $\mathcal{L}_1$ minimization), it requires all dictionary atoms (or data sample) to have unit length. Usually, we use $\mathcal{L}_2$ normalization. This normalization process project all atoms to unit hypersphere and eliminates the *locality* information of data as show by figure 3.3. As we can see, the neighborhood information is changed after

normalization.

Comparing to the strategy of adding regularization terms, we choose to search a local dictionary for each data sample while calculating the sparse coefficients. Unlike the method described in [4] which use the k-nearest neighbor as dictionary, we select atoms following the intrinsic manifold structure of data. The advantage of our selection is that it not only preserves the locality, but also captures the geometry structure of data (figure 3.4). As pointed out by [3], in many real applications, high-dimensional data always reside on or close to an intrinsically low dimensional manifold embedded in the high-dimensional ambient space. This is the fundamental assumption of manifold learning and also emphasizes the importance of utilizing manifold structure in learning algorithms. Our proposed algorithm has a user specific parameter k which leads to the lost of parametric-free characteristic. But our experiment results show that it increases the clustering performance and reduces the running time.

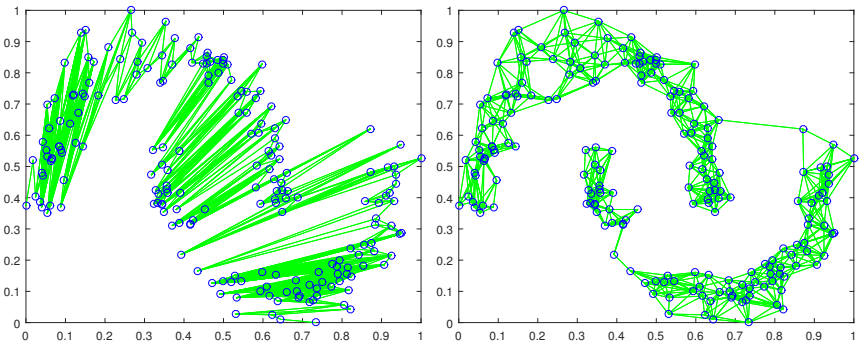


Figure 3.4: $\mathcal{L}_1$ graph (Left) and SA-$\mathcal{L}_1$ graph (Right,$K = 10$) of "two moon" dataset.

The basic idea of $\mathcal{L}_1$ graph is to find a sparse coefficient (or coding) for each data sample. Given dataset $\mathbf{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}]$, where $\boldsymbol{x_i} \in \mathbb{R}^m, i \in [1, \cdots, n]$ is a vector which represents a data sample. The sparse coefficient $\boldsymbol{\alpha}_i \in \mathbb{R}^{n-1}$ of $\boldsymbol{x_i}$ is calculated by following $\mathcal{L}_1$ minimization process.

$$\min_{\boldsymbol{\alpha_i}} \|\boldsymbol{\alpha_i}\|_1 \text{ subject to } \boldsymbol{x_i} = \boldsymbol{\Phi}^i \boldsymbol{\alpha_i}, \boldsymbol{\alpha_i} \geq 0. \tag{3.3}$$

We put constraint $\boldsymbol{\alpha_i} \geq 0$ here to let coefficients have physical meaning of *similarity*. In original $\mathcal{L}_1$ graph construction algorithm, the dictionary $\boldsymbol{\Phi}^i = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_{i-1}}, \boldsymbol{x_{i+1}}, \cdots, \boldsymbol{x_n}]$. Here, we select K atoms $\hat{\boldsymbol{\Phi}}^i = [\hat{\boldsymbol{x}}_1, \cdots, \hat{\boldsymbol{x}}_K]$ from $\boldsymbol{\Phi}^i$ by using manifold ranking scores [57] [58]. The algorithm can be described as Algorithm 4.

We use the closed form solution to calculate the manifold ranking scores

for all data samples:

$$\mathbf{F} = (I - \beta S)^{-1}, \tag{3.4}$$

where S is the Graph Laplacian matrix and we use Gaussian Kernel (parameter σ is configured as the mean distance) here. Each column of F is the relative manifold ranking scores of data sample $\mathbf{x}_i$.

Algorithm 4: SA-$\mathcal{L}_1$ graph

Input : Data samples $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}]$, where $\mathbf{x_i} \in \mathbf{X}$;
Parameter K;

Output: Adjacency matrix W of sparse graph.

Calculate the manifold ranking score matrix $\mathbf{F}$;
Normalize the data sample x_i with $\|x_i\|_2 = 1$;
for $\mathbf{x}_i \in X$ **do**
    Select top K atoms from $\mathbf{F}(i)$, and build $\hat{\mathbf{\Phi}}^i$;
    Solve: $\min_{\alpha_i} \|\alpha_i\|_1, \quad s.t. \quad x_i = \hat{\mathbf{\Phi}}^i \alpha_i, \alpha_i \geq 0$;
    $\mathbf{W}(i,:) = \alpha_i$;
end
return $\mathbf{W}$;

3.5 Experiments

3.5.1 Experiment Setup

Dataset. To demonstrate the performance of our proposed LOP-$\mathcal{L}_1$ graph and structure aware SA-$\mathcal{L}_1$ graph. we evaluate our algorithm on seven UCI benchmark datasets including three biological data sets (Breast Tissue(BT), Iris, Soybean), two vision image data set (Vehicle, Image,) and one chemistry data set (Wine) and one physical data set (Glass), whose statistics are summarized in Table 3.1. All these data sets have been popularly used in spectral clustering analysis research. These diverse combination of data sets are intended for our comprehensive studies.

Baseline. To investigate the quality of the generated LOP-$\mathcal{L}_1$ graph, we compare its performance on spectral clustering applications with $\mathcal{L}_1$ graph. At the sample time, we also select a full-scale Gaussian similarity graph (Gaussian graph), and a kNN Gaussian similarly graph (kNN graph) as our competitors to understand the quality of LOP-$\mathcal{L}_1$ graph better. Since we have ground truth

Name	#samples	#attributes	#clusters
Iris	150	4	3
BT	106	9	6
Wine	178	13	3
Glass	214	9	6
Soybean	307	35	19
Vehicle	846	18	4
Image	2000	19	7

Table 3.1: Datasets Statistics.

of labels for each data, we evaluate the spectral clustering performance with Normalized Mutual Information (NMI) and Accuracy (AC).

Parameter Setting. For LOP-$\mathcal{L}_1$ graph, the algorithm has one parameter named as *basis pool scaling parameter* t. It controls how many neighborhoods should be selected to the basis pool for each sample coding. We set t as a multiple value of attribute (or features) size w.r.t the data set.

$$2 \leq t \leq \frac{N}{m}, \tag{3.5}$$

where N is the number of samples and m is the sample dimensions. The reason we scale kNN neighborhood with Eq.3.5 is that we want to make it more adaptive to different context. In our experiments, we assign $t = 2, 3, 4$ and report the clustering performance results respectively. We will further analyze our selection of t in Section 3.5.2.

For Gaussian graph, the scaling parameter σ is configured as $\sigma = 0.1, 0.5, 1.0$. For kNN graph, we assign value of k as the size of basis pool of LOP-$\mathcal{L}_1$ graph with different t setting respectively. To obtain a fair comparison, we apply the same spectral clustering to measure their performance.

For SA-$\mathcal{L}_1$ graph, we select $\beta = 0.99$ for manifold ranking, and value K of kNN graph with Gaussian similarity (parameter σ equals to mean value) equals to 10%,20% and 30% percent of total number of data samples.

3.5.2 Analysis of Basis Pool Scaling

In our algorithm we argue that a constrained neighborhood as basis pool is not only enough but also provide locality property for the $\mathcal{L}_1$ graph construction. On the other hand, one of the most serious problem for kNN-based method is the over-sparsity where each sample has only a small amount of connected neighbors, which often results in that the derived graph is bias to some closely-

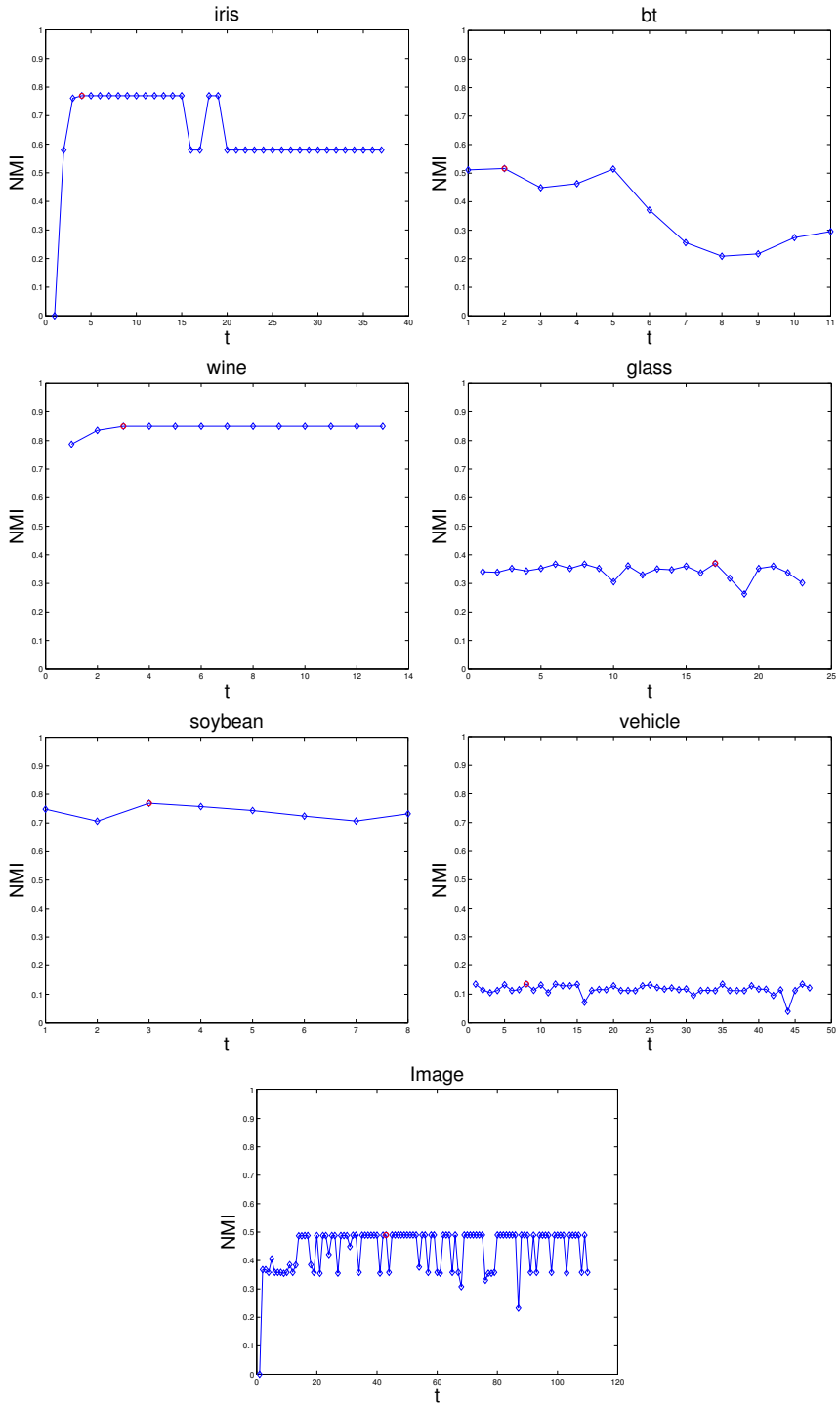


Figure 3.5: The change of NMI values w.r.t different selection of parameter t. Red dot in each subplot represents the maximal NMI. These experiments confirm that a basis neighborhood with certain size (with smaller t) provides better (or at least similar) performance than the overcomplete basis pool (with the maximal t in each subplot).

connected "cliques" and the subsequent analysis is therefore unreliable.

We confirm the effectiveness of our strategy by recording the trend of NMI value with increasing size of t (up to the maximal t w.r.t each dataset) in Fig. 3.5 across different dataset. It once again confirms that we don't need all remain samples as the basis pool to construct an informative yet stable $\mathcal{L}_1$ graph.

3.5.3 Performance of LOP-$\mathcal{L}_1$ Graph

In this section, we evaluate our proposed LOP-$\mathcal{L}_1$ graph algorithm and other three graph construction algorithms. Table 3.2 and Table 3.3 document the comparison results (in NMI and AC) of clustering performance.

LOP-$\mathcal{L}_1$ graph vs $\mathcal{L}_1$-Graph. LOP-$\mathcal{L}_1$ graph has better average performance than $\mathcal{L}_1$ graph. LOP-$\mathcal{L}_1$ graph has average NMI value 0.5032 and AC value 0.5852 while $\mathcal{L}_1$ graph has average NMI value 0.4611 and AC value 0.5643. For each specific data set, the clustering performance of $\mathcal{L}_1$ graph beats average performance of LOP-$\mathcal{L}_1$ graph on Iris, BT, Image but lose on others. Moreover, we observe that the highest NMI value between them occurs at a specific t value of LOP-$\mathcal{L}_1$ graph, for example, the highest NMI values of Image data set is at $t = 2, 3$ of LOP-$\mathcal{L}_1$ graph.

LOP-$\mathcal{L}_1$ graph vs kNN-Graph. The average clustering performance of kNN graph is the lowest one among Spectral Clustering with Gaussian similarity graph, $\mathcal{L}_1$ graph and LOP-$\mathcal{L}_1$ graph. Comparing to LOP-$\mathcal{L}_1$ graph, kNN graph only have better performance (NMI: 0.4739, AC: 0.5346) than LOP-$\mathcal{L}_1$ graph (NMI: 0.4328, AC: 0.5189) on BT data set.

LOP-$\mathcal{L}_1$ graph vs Gaussian Similarity Graph. The spectral clustering with Gaussian similarity graph (fully connected graph) has lower average performance than LOP-$\mathcal{L}_1$ graph in our experiments. However, for specific data set, the maximum values of NMI and AC not always belong to LOP-$\mathcal{L}_1$ graph. For example, the highest NMI value for Iris data set is Gaussian similarity graph with $\sigma = 0.1$. The reason is that the spectral clustering based on Gaussian similarity graph is parameter sensitive. To obtain the best result, the user has to tune the parameter σ.

3.5.4 Performance of SA-$\mathcal{L}_1$ Graph

Comparing to LOP-$\mathcal{L}_1$ graph, SA-$\mathcal{L}_1$ graph shows overall better performance than Gaussian similarity graph and original $\mathcal{L}_1$ graph as show by Table 3.4.

Name	Gaussian graph			kNN graph			$\mathcal{L}_1$ graph	LOP-$\mathcal{L}_1$ graph		
	$\sigma = 0.1$	$\sigma = 0.5$	$\sigma = 1.0$	$t = 2$	$t = 3$	$t = 4$		$t = 2$	$t = 3$	$t = 4$
Iris	**0.8640**	0.5895	0.7384	0.4831	0.5059	0.3139	0.7523	0.5794	0.7608	0.7696
BT	0.4933	0.4842	0.4691	0.4731	**0.5335**	0.4150	0.3660	0.3912	0.4536	0.4536
Wine	0.4540	0.7042	0.6214	0.6647	0.7471	0.7031	0.6537	0.8358	**0.8500**	**0.8500**
Glass	0.3535	0.2931	0.3289	0.2584	0.3475	0.3114	0.3416	0.3533	**0.3575**	0.2988
Soybean	0.6294	0.6814	0.6170	0.6291	0.6120	0.5835	0.7004	**0.7265**	0.7180	0.7267
Vehicle	0.1248	0.0976	0.0958	0.1101	0.0779	0.0667	0.0726	**0.1352**	0.1019	0.1106
Image	**0.4800**	0.4678	0.4740	0.3256	0.4434	0.4548	0.3410	0.3678	0.3678	0.3582

Table 3.2: NMI comparison of LOP-$\mathcal{L}_1$ graph and other three graph construction methods.

Name	Gaussian graph			kNN graph			$\mathcal{L}_1$ graph	LOP-$\mathcal{L}_1$ graph		
	$\sigma = 0.1$	$\sigma = 0.5$	$\sigma = 1.0$	$t = 2$	$t = 3$	$t = 4$		$t = 2$	$t = 3$	$t = 4$
Iris	0.9600	0.7267	0.8600	0.7533	0.6670	0.5800	0.8867	0.6400	**0.9933**	0.9000
BT	0.5472	0.4906	0.5189	0.4717	**0.6038**	0.5283	0.4434	0.4623	0.5472	0.5472
Wine	0.6292	0.8876	0.8820	0.8483	0.9101	0.9101	0.8652	0.9551	**0.9607**	**0.9607**
Glass	0.4112	0.3972	0.4299	0.4299	**0.5000**	0.4860	0.4579	0.4673	0.4907	0.4299
Soybean	0.5081	0.5668	0.4300	0.5049	0.4853	0.5016	0.5244	0.5700	0.5668	**0.6059**
Vehicle	0.3818	0.3582	0.3605	0.3806	0.3475	0.3381	0.3771	**0.3936**	0.3593	0.3676
Image	**0.5467**	0.5124	0.5076	0.4600	0.4838	0.4781	0.3952	0.3919	0.3919	0.3881

Table 3.3: Accuracy comparison of LOP-$\mathcal{L}_1$ graph and other three graph construction methods.

Name	Metric	$\mathcal{L}_1$	kNN Graph			SA-$\mathcal{L}_1$ graph		
			K:10%	K:20%	K:30%	K:10%	K:20%	K:30%
Iris	NMI	0.3615	0.4765	0.3883	0.4200	0.4287	**0.6103**	0.5827
	AC	0.6900	0.5133	0.6800	0.6933	0.7133	**0.8067**	0.6800
BT	NMI	0.4055	0.4839	0.4749	0.5178	**0.5436**	0.5524	0.4702
	AC	0.5283	0.5189	0.5189	0.5377	**0.6604**	0.6321	0.5755
Wine	NMI	0.7717	0.8897	0.8897	0.8897	**0.9209**	0.8946	0.8043
	AC	0.9326	0.9719	0.9719	0.9717	**0.9775**	0.9663	0.9382
Glass	NMI	0.3794	0.3642	0.3763	0.2572	0.3746	**0.3998**	0.3715
	AC	0.4486	0.5140	0.5187	0.4439	0.4486	**0.4579**	0.4533
Soybean	NMI	0.6531	0.6509	0.7022	0.6884	0.6858	0.7096	**0.7192**
	AC	0.4984	0.4625	0.5505	0.5212	0.5179	0.5179	**0.5505**
Vehicle	NMI	0.1424	0.0802	0.0806	0.0814	0.1173	0.1127	**0.1651**
	AC	0.3747	0.3664	0.3676	0.3582	0.3818	0.3818	**0.3830**
Image	NMI	0.5658	0.5514	0.5454	0.5699	0.5034	**0.5877**	0.5694
	AC	0.6271	0.4752	0.5286	0.5505	0.5443	**0.6467**	0.6133

Table 3.4: Clustering performance of SA-$\mathcal{L}_1$ graph construction algorithms. $\mathcal{L}_1$ graph is the baseline.

3.6 Chapter Summary

Classic $\mathcal{L}_1$ graph exhibits good performance in many data mining applications. However, due to the over-complete basis and the following lack of coding focus, the locality and the similarity among the samples to be encoded are lost. To preserve locality, sparsity and good performance in a concise and efficient way, we propose a Locality-Preserving $\mathcal{L}_1$ graph (LOP-$\mathcal{L}_1$). By limiting the coding process in a local neighborhood to preserve localization and coding stability, our proposed LOP-$\mathcal{L}_1$ alleviates the instability of sparse codes and outperforms the existing works.

LOP-$\mathcal{L}_1$ graph use the Euclidean distance to search the dictionary for each datum. As a result, the manifold structure hidden behind the input data is ignored. To exploit the geometry structure of data, we propose the structure aware (SA) $\mathcal{L}_1$ graph by using manifold ranking technique.

We apply our proposed methods on clustering application and the experiment result confirm the effectiveness of our proposed method.

Chapter 4

Greedy Sparse Graph by Using Ranked Dictionary

In this chapter, we propose a greedy algorithm to speed up the construction of ℓ_1 norm based sparse graph. Moreover, we introduce the concept of "Ranked Dictionary" for ℓ_1 minimization. This ranked dictionary not only preserves the locality but also removes the randomness of neighborhood selection during the process of graph construction. To demonstrate the effectiveness of our proposed algorithm, we present our experimental results on several commonly-used datasets using two different ranking strategies: one is based on Euclidean distance, and another is based on diffusion distance.

4.1 Chapter Introduction

As mentioned before, $\mathcal{L}_1$ graph has several disadvantages when apply to general dataset without the assumption of subspace structure. Motivated by these limitations, many research works have been proposed in machine learning and data mining research area. Without lost of generality, we would like to classify those algorithms into two categories: *soft-modification* and *hard-modification*.

1. *Soft-modification* algorithms. Algorithms in this category usually add one or more regularization terms to the original $\mathcal{L}_1$ minimization objective function in Eq. (1.1). For example, the structure sparsity [1] preserves the local structure information of input data, the auto-grouped sparse regularization [2] adds the group effect to the final graph, and the Graph Laplacian regularization [59] [3] lets the closed data samples have similar sparse coding coefficients (or $\boldsymbol{\alpha}_i$).

2. *Hard-modification* algorithms. These algorithms define a new dictionary

for each data sample during $\mathcal{L}_1$ minimization. By reducing the solvers' solution space for each data sample into a local space, the locality of input data is preserved and the computational time of $\mathcal{L}_1$ minimization (Eq. (1.1)) is reduced. For example, the locality preserved (LOP) $\mathcal{L}_1$ graph described in Section 3.3 is utilizing k-nearest neighbors as dictionaries.

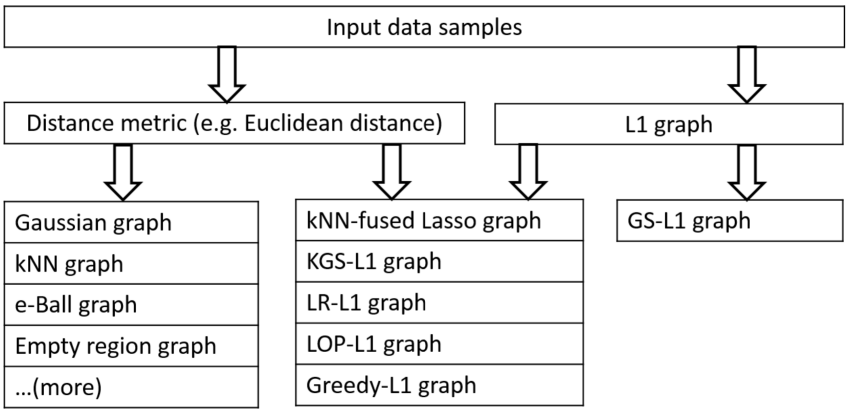


Figure 4.1: Connection of Greedy $\mathcal{L}_1$ graph to other graphs. Several of them are: kNN-fused Lasso graph [1], Group Sparse (GS) $\mathcal{L}_1$ graph, Kernelized Group Sparse (KGS) $\mathcal{L}_1$ graph [2], Laplacian Regularized (LR) $\mathcal{L}_1$ graph [3] and Locality Preserving (LOP) $\mathcal{L}_1$ graph [4].

The *soft-modification* algorithms preserve the nonparametric feature and improve the quality of $\mathcal{L}_1$ graph by exploiting the intrinsic data information such as geometry structure, group effects, etc. However, those algorithms still have high computational cost. This is unpleasant for the large-scale dataset in this "Big-data" era. To improve, in this chapter we propose a greedy algorithm to generate $\mathcal{L}_1$ graph. The generated graphs are called **Greedy-$\mathcal{L}_1$** graphs. Our algorithm employs greedy $\mathcal{L}_1$ minimization solvers and is based on non-negative orthogonal matching pursuit (NNOMP). Furthermore, we use ranked dictionaries with reduced size K which is a user-specified parameter. We provide the freedom to the user to determine the ranking strategy such as nearest neighbors, or diffusion ranking [60]. Our algorithm has significant time-reduction about generating $\mathcal{L}_1$ graphs. Comparing to the original $\mathcal{L}_1$ graph construction method, our algorithm loses the nonparametric characteristics and is only offering a sub-optimal solution comparing to solutions that use non-greedy solvers and deliver global optimal solution. However, our experimental results show that the graph generated by our algorithm has equal (or even better) performance as the original $\mathcal{L}_1$ graph by setting K equals to the length

of data sample. Our work is a natural extension of existing $\mathcal{L}_1$ graph research. A concise summary of the connection between our proposed Greedy-$\mathcal{L}_1$ graph and other graphs is illustrated in Figure 4.1.

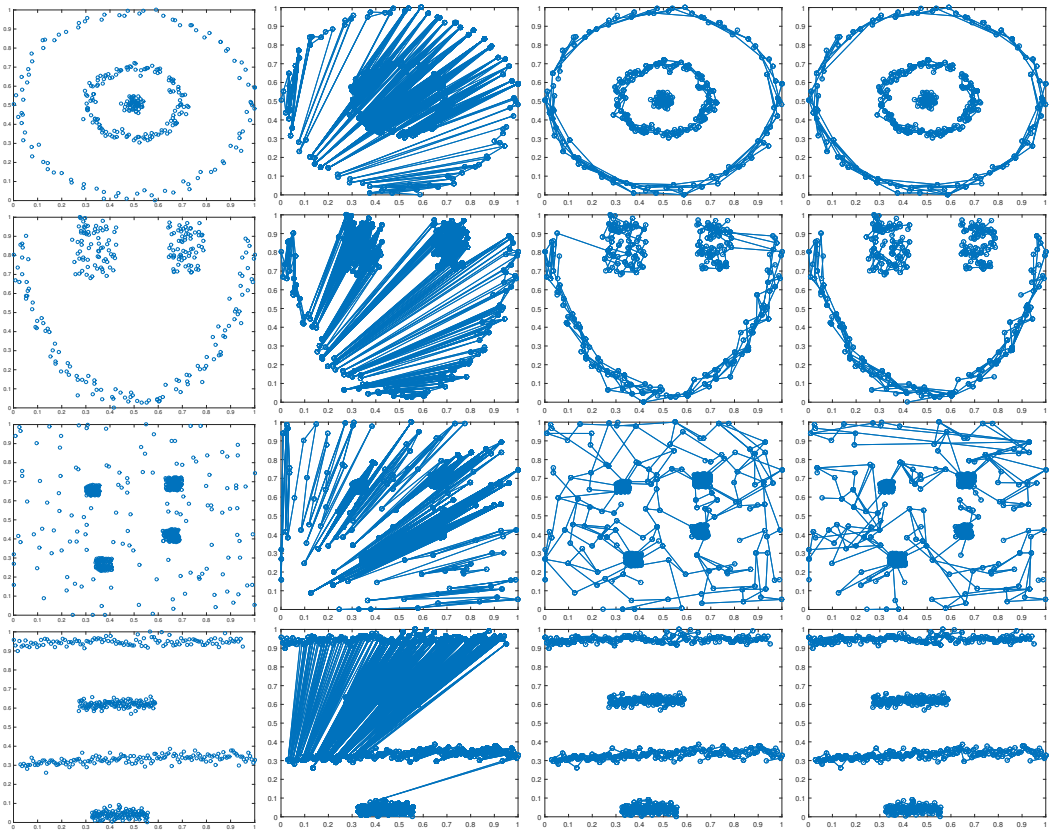


Figure 4.2: $\mathcal{L}_1$ graphs generated by different construction algorithms. From left to right: 2D toy dataset; $\mathcal{L}_1$ graph; Greedy-$\mathcal{L}_1$ graph with Euclidean metric (K=15); Greedy-$\mathcal{L}_1$ graph with Diffusion metric (K=15).

The organization of this chapter is as follows. First, the unstable solutions caused by different $\mathcal{L}_1$ solvers will be presented in Section 4.2. Second, we will introduce our proposed greedy algorithm in Section 4.3. After that, we will give a review of existing works on how to improve the quality of $\mathcal{L}_1$ graph. Finally, we will present our experimental results in Section 4.4 and draw conclusion in Section 4.5.

4.2 Unstable Solutions caused by Different $\mathcal{L}_1$ Solvers

To solve the optimization problem (2.3), we need a numerical solver. There are many popular ones with various minimization methods [61] such as *gradient projection, homotopy* and *proximal gradient*. Moreover, all these solvers have their own special parameter settings. As a result, if we choose different parameters, the numerical results will be not same. Also, several To illustrate this phenomenon, we exam the UCI Image dataset with "spams-matlab" software [62] and "l1_ls" software [37]. For each solver, we set the parameter λ to different values as: $[0.01, 0.001, 0.0001]$. For the experiment, we select the first sample of Image dataset as source sample, and others as dictionary. To see the unstable solutions, we list the top five neighbors (Index) and its corresponding weights (Value). The result is show in below table: As we can see,

Solver	λ	Index(Value)
L1-ls	0.01	5(0.2111),14(0.4449),17(0.2718),38(0.0418),575(0.0163)
Spams-matlab	0.01	5(0.2632),13(0.0044),14(0.3525),17(0.2819)
L1-ls	0.001	5(0.0771),14(0.4540),17(0.3005),38(0.0715),575(0.0908)
Spams-matlab	0.001	5(0.2851),14(0.3676),17(0.3142),38(0.0043)
L1-ls	0.0001	14(0.3861),17(0.4051),32(0.0292),36(0.0211),575(0.1413)
Spams-matlab	0.0001	5(0.2621),14(0.4171),17(0.2744),38(0.0346),225(0.0068)

Table 4.1: The effect of unstable solutions caused by using different solvers or with different parameters.

.

the majority neighbors between "spams-matlab" and "l1_ls" are same except some minor difference. However, the weights are very different and unstable. This unstable situation is not only with different parameter λ, but also with different solvers. This is a disadvantage for using $\mathcal{L}_1$ graph as similarity graph for graph oriented machine learning tasks.

4.3 Algorithm

In this section, we introduce the concept of ranked dictionary and two different ranking strategies: Euclidean distance ranking and Diffusion ranking. These different ranking methods are proposed for different type of data. For example, Diffusion ranking is suitable for data with manifold structure,and Euclidean distance is the popular choice for general data. Obviously, there are many other distance choices such as cosine distance could be used for ranking, and

it's upon user's judgment for the right choice. Furthermore, we present a greedy algorithm at the end of this section.

4.3.1 Ranked Dictionary

We propose a "ranked dictionary" to substitute the original dictionary Φ^i in equation (1.1). Our claim it that the "ranked dictionary" not only preserves the locality of data, which is important for clustering applications, but also solve the "curse of dictionary normalization" dilemma. The idea of "ranked dictionary" is to rank the neighborhood information following a given distance metric such as Euclidean distance in vector space. By selecting the top K nearest neighbors as dictionary, the new dictionary Φ^i_K keeps the order of nearest neighbors and captures the local distribution of data samples. Moreover, Φ^i_K has smaller size comparing to $n-1$ while n equals to the number of data samples.

There is a subtle difference between k value of popular k-nearest neighbor (kNN) graph and the K value in our proposed "ranked dictionary" Φ^i_K. Usually, the users set the value k of kNN graph in the order of $\log(n)$ which is the asymptotic connectivity result [63] that makes the kNN graph to be connected. For K value of Φ^i_K, it needs to be larger than d which is the dimension of vector x_i. This requirement is to increase the feasibility of finding successful sparse linear representation (or signal recover).

The using of truncated version of dictionary Φ is proved to success in building quality $\mathcal{L}_1$ graph for clustering application [4]. However, it can not solve the dilemma that there might exist data samples with the same direction but different length in input data. The dictionary normalization process will project them onto to the same location at hypersphere. Since they have the same values, the $\mathcal{L}_1$ minimization solver will choose one of them randomly. To avoid this randomness, we need to rank those atoms (or data samples) of dictionary.

Euclidean Distance Ranking. Using Euclidean metric to rank atoms of dictionary is quite straightforward. We rank them by distance. The shorter distance will have a higher rank score. The Euclidean distance is defined as:

$$dist(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = (\sum_{k=1}^{n} |x_i(k) - x_j(k)|^2)^{1/2}. \tag{4.1}$$

Diffusion Distance Ranking. As pointed out by Yang et al. [3], many real-world datasets are similar to an intrinsic low dimensional manifold embedded in high dimensional ambient space, and the geometry structure of manifold can

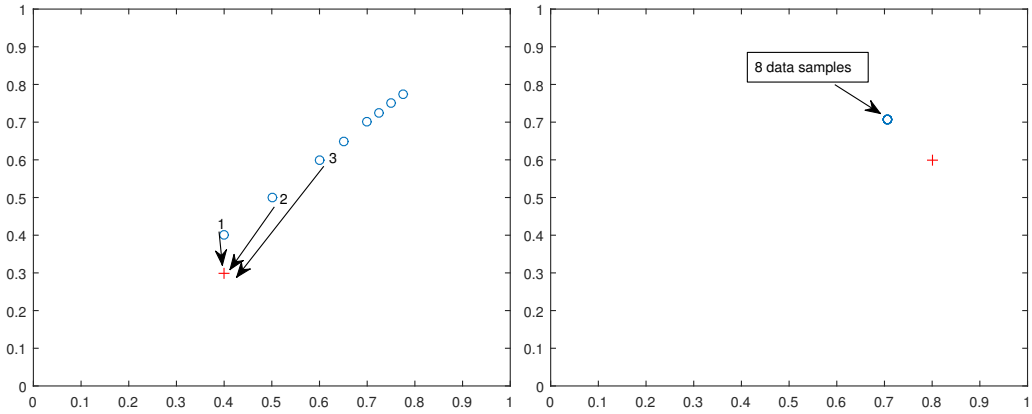


Figure 4.3: Ranked dictionary. Left: eight data samples have the same direction but with different length. Red cross is the target data sample for calculating sparse coefficients. Right: after normalization, those eight data samples have the same location.

be used to improve the performance of learning algorithms. we now present a strategy to search dictionaries following the geometry structure of input data. Based on the diffusion theory [60] [64], we rank the atoms of dictionary through diffusion matrix. A diffusion process has three stages [64]: (1) initialization; (2) definition of transition matrix; (3) definition of the diffusion process. In our setting, the first stage is to build an affinity matrix $\boldsymbol{A}$ from the input dataset $\boldsymbol{X}$. We use Gaussian kernel to define the pairwise distance:

$$\boldsymbol{A}(i,j) = \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{2\sigma^2}\right), \tag{4.2}$$

where $\boldsymbol{A}(i,j)$ is the distance between data sample $\boldsymbol{x_i}$ and data sample $\boldsymbol{x_j}$, and σ is a normalization parameter. In our configuration, we use the median of K nearest neighbors to tune σ. The second stage is to define the transition matrix $\boldsymbol{P}$:

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}, \tag{4.3}$$

where $\boldsymbol{D}$ is a $n \times n$ degree matrix defined as

$$\boldsymbol{D}(i,j) = \begin{cases} \sum_{j=1}^{n} \boldsymbol{A}(i,j) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

Now the diffusion process can be defined as:

$$\boldsymbol{W}_{t+1} = \boldsymbol{P}\boldsymbol{W}_t\boldsymbol{P}', \tag{4.5}$$

where $\boldsymbol{W}_0 = \boldsymbol{A}$ and t is the number of steps for diffusion steps. Each row of $\boldsymbol{W}_t$ is the diffusion ranking scores. In this paper, we let t equal to K for the sake of simplicity. Once $\boldsymbol{W}_t$ is calculated, the first K data samples with top scores of each row is selected as dictionary. The algorithmic details can be documented as follows:

Algorithm 5: DiffusionDictionary

Input : Data samples $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}]$, where $\boldsymbol{x}_i \in \boldsymbol{X}$;
Size of dictionary: K;

Output: Diffusion dictionary index matrix $\boldsymbol{\Phi}_K$.

Calculate Gaussian similarity graph $\boldsymbol{A}$;
$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$;
`/* calculate diffusion process iteratively. */`
for $t = 1 : K$ **do**
  $\boldsymbol{W}_t = \boldsymbol{P}\boldsymbol{W}_{t-1}\boldsymbol{P}^{'}$
end
`/* sort each row in descend order. */`
for $i = 1 : n$ **do**
  sort$(\boldsymbol{W}_t(i,:))$
end
`/* fetch the index of highest` K `values in each row of` W_t `*/`
for $i = 1 : n$ **do**
  $\boldsymbol{\Phi}(i,:)$ =index$(\boldsymbol{W}_t(i, 1:k))$
end

4.3.2 Greedy $\mathcal{L}_1$ Graph

To solving the $\mathcal{L}_1$ norm minimization problem, we need an efficient solver [61]. For datasets that size are larger than $3,000$ with reasonable dimensions, greedy solver like Basic pursuit(BP) [36] [24] or Orthogonal Matching Pursuit(OMP) [65] is more suitable [66]. In this section, We propose a greedy algorithm to build $\mathcal{L}_1$ graph. Our proposed algorithm is based on OMP [66] and NNOMP [67] [68]. By using greedy solver, we switch the $\mathcal{L}_1$ minimization problem (**P1**) back to the original $\mathcal{L}_0$ optimization with(**P2**)/without(**P3**) non-negative constraints

as:

$$(\mathbf{P2}) \quad \min_{\boldsymbol{\alpha_i}} \|\boldsymbol{\alpha_i}\|_0 \text{ subject to } \boldsymbol{x_i} = \boldsymbol{\Phi}^i \boldsymbol{\alpha_i}, \boldsymbol{\alpha_i} \geq 0. \tag{4.6}$$

$$(\mathbf{P3}) \quad \min_{\boldsymbol{\alpha_i}} \|\boldsymbol{\alpha_i}\|_0 \text{ subject to } \boldsymbol{x_i} = \boldsymbol{\Phi}^i \boldsymbol{\alpha_i}. \tag{4.7}$$

The main difference between our algorithm and the original OMP and NNOMP is that the atoms of dictionary are ranked. We force the solver to choose and assign weights to atoms that are closer to source data sample before normalization. To clarify our idea, we present the improved version of NNOMP solver in Algorithm (6). For OMP solver, the idea and process are same.

4.3.3 Connection to Subspace Clustering

$\mathcal{L}_1$ graph is almost the same as the similarity graph of sparse subspace clustering (SSC) algorithm [25]. However, they have different assumptions about the data. The $\mathcal{L}_1$ graph is defined for general data and doesn't have any specific assumption about data like k-nearest neighbor graph, while the similarity graph of SSC assumes the data is lied in a union of low-dimensional subspaces [25].

The success of $\mathcal{L}_1$ graph is first applied to human face images clustering [26] [52]. Those face images data has two sufficient conditions for the success of using $\mathcal{L}_1$ graph for spectral clustering: (1) the dimension of data vector is high. (2) different human face images stay in different subspaces. However, for general data, these two conditions are not always exist. By the experiment results from research work [4], the Ng, Jordan, Weiss and et al. (NJW) spectral clustering algorithm [69] with Gaussian similarity graph has better performance than with $\mathcal{L}_1$ graph on several general datasets. So here, we argue that the power of $\mathcal{L}_1$ graph follows the assumption of sparse subspace clustering.

4.3.4 Connection to Locally Linear Embedding

The idea of "ranked dictionary" has a connection to Locally Linear Embedding(LLE) [70]. LLE solves the following minimization problem:

$$\epsilon(w) = \sum_i |\mathbf{x}_i - \sum_j \mathbf{w}_{ij}\mathbf{x}_j|^2. \tag{4.8}$$

The cost function $\epsilon(w)$ is the add up of the squared distance between all data samples (x_i) and their reconstructions $\sum_j \mathbf{w}_{ij}\mathbf{x}_j$. There are two constraints

Algorithm 6: Greedy$\mathcal{L}_1$Graph.

Input : Data sample $\boldsymbol{x}$;
Ranked dictionary $\boldsymbol{\Phi_K}$;
Residual threshold $\theta_{threshold}$

Output: Sparse coding $\boldsymbol{\alpha}$ of $\boldsymbol{x}$.

for $i = 1 : \|\boldsymbol{x}\|_1$ **do**
  if $i == 0$ **then**
    Temporary solution: $\boldsymbol{\alpha}^i = 0$;
    Temporary residual: $\boldsymbol{r}^i = \boldsymbol{x} - \boldsymbol{\Phi_K}\boldsymbol{\alpha}^i$;
    Temporary solution support: $\boldsymbol{S}^i = Support\{\boldsymbol{\alpha}^i\} = \emptyset$;
  else
    for $j = 1 : k$ **do**
      /* $\boldsymbol{\phi}_j$ is the j-th atom of $\boldsymbol{\Phi_K}$ */
      $\epsilon(j) = \min_{\alpha_j \geq 0} \|\boldsymbol{\phi_j}\boldsymbol{\alpha_j} - \boldsymbol{r}^{i-1}\|_2^2 = \|\boldsymbol{r}^{i-1}\|_2^2 - \max\{\boldsymbol{\phi}_j^T \boldsymbol{r}^{i-1}, 0\}^2$.
    end
    Find j_0 such that $\forall j \in \boldsymbol{S}^c, \epsilon(j_0) \leq \epsilon(j)$, if there are multiple j_0 atoms, choose the one with smallest index value.;
    Update support: $\boldsymbol{S}^i = \boldsymbol{S}^{i-1} \cup \{j_0\}$;
    Update solution: $\boldsymbol{\alpha}^i = \min_z \|\boldsymbol{\Phi}_K\boldsymbol{\alpha} - \boldsymbol{x}\|_2^2$ subject to $Support\{\boldsymbol{\alpha}^i\} = S^i$ and $\boldsymbol{\alpha^i} \geq 0$;
    Update residual: $\boldsymbol{r}^i = \boldsymbol{x} - \boldsymbol{\Phi_K}\boldsymbol{\alpha^i}$;
    if $\|\boldsymbol{r}^i\|_2^2 < \theta_{threshold}$ **then**
      Break;
    end
  end
end
Return α^i;

during the minimization process: (1) the $\mathbf{x}_j$ are selected to be k nearest neighbor of of $\mathbf{x}_i$, where k is a parameter set by user; (2) the row of weight matrix sum to one: $\sum_j \mathbf{w}_{ij} = 1$.

If we compare the equation 4.8 of LLE with equation 1.1 of $\mathcal{L}_1$ graph and "ranked dictionary", we can find that both of them are finding a linear representation relationship between a given data sample and its k nearest neighbors. However, $\mathcal{L}_1$ graph with "ranked dictionary" looks for a sparse reconstruction weights, and prefer to assign non-zero weights for nearest neighbors $\mathbf{x}_j$ that stay in the same subspace as the given data sample $\mathbf{x}_i$. The second difference is the unique advantage of $\mathcal{L}_1$ graph.

4.3.5 Spectral Clustering Performance

One major application of $\mathcal{L}_1$ graph is spectral clustering. Researchers use $\mathcal{L}_1$ graph as the similarity graph of spectral clustering algorithm by treating the sparse coefficients as similarity values. The similarity graph models the cluster structure of original data with pre-defined similarity metric, and has significant impact to the performance of spectral clustering algorithm. A good similarity graph should have high weights for edges within same cluster and low weights for edges between different clusters. However, there is no explicit measurement of the quality of similarity graph from theoretical research as point out by [8]. Instead, the clustering performance, like Mutual Information and Accuracy, is used to tell whether the similarity graph is in high quality or not implicitly. "Locality" is another guidance to judge the quality of similarity graph [32]. "Locality" stresses that the edges of similarity graph should connect data points locally as non-local edges will affect the result of graph cut [43] then the performance of spectral clustering [8]. In this section, we try to explain how $\mathcal{L}_1$ graph with "ranked dictionary" can generate high quality similarity graphs.

"Ranked dictionary" preserves the locality of data by only selecting k nearest neighbors as dictionary. For a given source data point, "ranked dictionary" constrains the possible candidates that it can connect to. There is a difference between k nearest neighbor of kNN graph and our proposed Greedy $\mathcal{L}_1$ graph. We show it in the Figure (4.4).

As we can see, Greedy $\mathcal{L}_1$ graph selects a larger range than kNN graph but a much smaller one than original $\mathcal{L}_1$ graph. It preserves the locality of data in a "Hard-modification" way as we introduced in the beginning of this work. And this locality preserving ability has been proved in previous research work [71].

Another important aspect of Greedy $\mathcal{L}_1$ graph is that it preserves the local subspaces through OMP solver. As the theory proof in [66], if coherence between the residual vectors (set of r^i in line 13 of algorithm (6)) and subspaces satisfies a data dependent condition, the OMP solver preserves the subspaces of input data. Based on this, we observe another difference with kNN graph: the Greedy $\mathcal{L}_1$ graph prefers to create connections between data samples within same subspace, while the kNN graph selects edges according to the given distance metric.

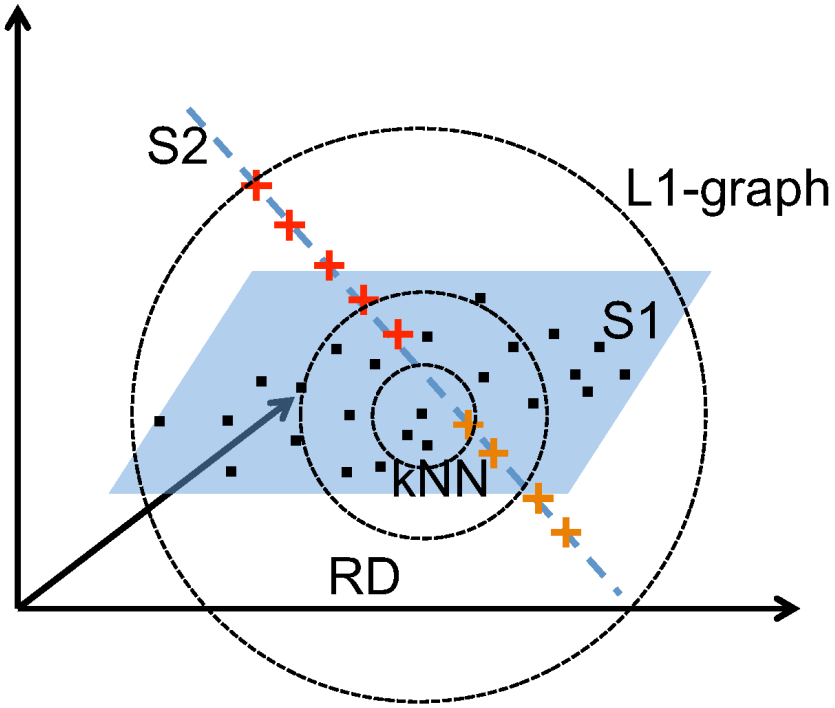


Figure 4.4: The range difference of "Ranked Dictionary"(RD), "kNN" and original "$\mathcal{L}_1$ graph". The toy dataset include two subspace **S1** and **S2**. The selection range of nearest neighbors is shown by dash circles.

4.4 Experiments

We present our experimental results in this section. The datasets in our experiments can be divided into small size data and large size data. The reason for this separation is that calculating the global optimization for $\mathcal{L}_1$ minimization is time-consuming for large size data (number of instances are larger than 3000.) For those large size data, we use an efficient OMP solver from "spams-matlab" [62]. As a consequence, the generated $\mathcal{L}_1$ graphs are not from optimal sparse coding solutions.

The effectiveness of our proposed graph construction methods is evaluated through NJW spectral clustering algorithm [69]. To satisfy the input of spectral clustering algorithm, we transform the adjacency matrix of $\mathcal{L}_1$ graph $\boldsymbol{W}$ into a symmetry matrix $\boldsymbol{W}^{'}$ by $\boldsymbol{W}^{'} = (\boldsymbol{W} + \boldsymbol{W}^{T})/2$. All analyses and experiments are carried out by using Matlab on a server with Intel 12-core 3.4GHz CPU and 64GB RAM.

Solvers. We use three solvers in our experiments. For small size dataset,

"l1-ls" is used for creating $\mathcal{L}_1$ graph, and our proposed NNOMP solver is used for Greedy $\mathcal{L}_1$ graph. For large dataset, we use "spams-matlab" software [62], which is an efficient implementation of sparse optimization by using multi-thread techniques, to build the $\mathcal{L}_1$ graph and Greedy $\mathcal{L}_1$ graph.

Evaluation Metrics. We evaluate the spectral clustering performance with Normalized Mutual Information (NMI) and Accuracy (ACC). NMI value ranges from 0 to 1, with higher values meaning better clustering performance. AC is another metric to evaluate the clustering performance by measuring the fraction of its clustering result that are correct. It's value also ranges from 0 to 1, and the higher the better.

4.4.1 Small-sized Data

Datasets. To demonstrate the performance of our proposed algorithm, we evaluate it on seven UCI benchmark datasets including three biological data sets (BreastTissue, Iris, Soybean), two vision image data sets (Vehicle, Image), one chemistry data set (Wine), and one physical data set (Glass), whose statistics are summarized in Table 4.2. All of these data sets have been popularly used in spectral clustering analysis research. The diverse combinations of data sets are necessary for our comprehensive studies.

Name	#samples	#attributes	#clusters
BreastTissue (BT)	106	9	6
Iris	150	4	3
Wine	178	13	3
Glass	214	9	6
Soybean	307	35	19
Vehicle	846	18	4
Image	2100	19	7

Table 4.2: Statistics of small-sized datasets.

Baselines and Parameters Setting. We compare the spectral clustering performance with Gaussian similarity graph and original $\mathcal{L}_1$ graph. For experiments with small size datasets, we use the *l1_ls* solver [54] for original $\mathcal{L}_1$ graph construction algorithms. We set the solver's parameter λ to 0.1. The *threshold* $\theta_{threshold}$ of Greedy solver 6 is set to $1e-5$. For Gaussian graph and Greedy-$\mathcal{L}_1$ graph, we select three different K values and document their clustering performance results respectively. The K is set to be the multiple of data attribute size. The results are documented in Table 4.3 and Table 4.4.

Name	$\mathcal{L}_1$ graph	Gaussian graph	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
BT	0.4582	0.4606	0.5473	0.4517	0.5024	0.4197	0.4073	0.3839
Iris	0.5943	0.7364	0.3950	0.4623	0.4070	0.5106	0.4626	0.4640
Wine	0.7717	0.8002	0.8943	0.9072	0.8566	0.6925	0.4291	0.6093
Glass	0.3581	0.2997	0.2569	0.3688	0.3039	0.2991	0.3056	0.2918
Soybean	0.7373	0.6958	0.6919	0.6833	0.6775	0.5788	0.5493	0.5432
Vehicle	0.1044	0.1870	0.1512	0.2121	0.2067	0.1438	0.1035	0.1244
Image	0.4969	0.4652	0.5821	0.6673	0.6649	0.4866	0.4483	0.3155
Average	0.5030	0.5207	0.5170	**0.5361**	0.5170	0.4473	0.3865	0.3903

Table 4.3: NMI comparison of graph construction algorithms. M is the number of attributes.

Name	$\mathcal{L}_1$ graph	Gaussian graph	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
BT	0.5472	0.5377	0.6698	0.4811	0.5943	0.4528	0.4906	0.4717
Iris	0.7400	0.8867	0.6933	0.7200	0.6800	0.7200	0.6533	0.6400
Wine	0.9326	0.9438	0.9719	0.9719	0.9551	0.8989	0.7865	0.8596
Glass	0.4206	0.4112	0.4579	0.4533	0.4346	0.4626	0.4813	0.5187
Soybean	0.6156	0.5440	0.5244	0.4853	0.5016	0.4430	0.3746	0.4876
Vehicle	0.3713	0.4515	0.4539	0.4243	0.4090	0.3664	0.3522	0.3605
Image	0.5629	0.4595	0.6348	0.7181	0.7043	0.5190	0.5524	0.3505
Average	0.6105	0.6049	0.6227	**0.6288**	0.6141	0.5683	0.5334	0.5362

Table 4.4: ACC comparison of different graph construction algorithms. M is the number of attributes.

Name	$\mathcal{L}_1$ graph	Gaussian graph	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
BT	0.0604	1	0.0457	0.0615	0.0705	0.0341	0.0442	0.0548
Iris	0.0403	1	0.0217	0.0288	0.0311	0.0203	0.0237	0.0265
Wine	0.0600	1	0.0413	0.0496	0.0552	0.0347	0.0409	0.0437
Glass	0.0369	1	0.0242	0.0308	0.0349	0.0188	0.0204	0.0239
Soybean	0.030	1	0.0286	0.0317	0.0346	0.0258	0.0299	0.034
Vehicle	0.0135	1	0.0104	0.0124	0.0135	0.0062	0.0074	0.0084
Image	0.0039	1	0.0034	0.004	0.0044	0.0026	0.0029	0.0027

Table 4.5: Graph sparsity comparison of different graph construction algorithms. M is the number of attributes.

Greedy-$\mathcal{L}_1$ Graph vs. Gaussian Graph. Overall, the Greedy-$\mathcal{L}_1$ graph using Euclidean metric has better average spectral clustering performance than Gaussian graphs. However, we also observer that Guassian graph has overall better performance on "Iris", "Soybean" and "Vehicle" datasets.

Greedy-$\mathcal{L}_1$ Graph vs. $\mathcal{L}_1$ Graph. Greedy-$\mathcal{L}_1$ graph has better clustering performance than $\mathcal{L}_1$ graph on average. However, for iris and soybean datasets, the $\mathcal{L}_1$ graph shows the best clustering result: Iris (NMI=0.5943, ACC=0.74); Soybean (NMI=0.7373, ACC=0.6156). The best result of Greedy-$\mathcal{L}_1$ graph

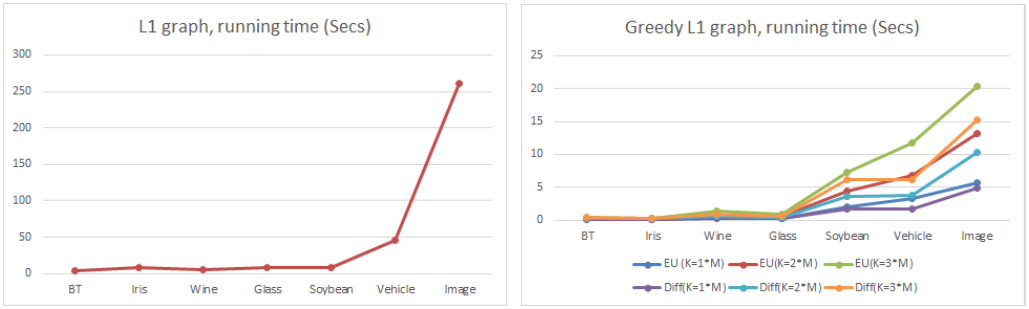


Figure 4.5: Running time of different $\mathcal{L}_1$ graph construction algorithms. Top: original $\mathcal{L}_1$ graph construction algorithm. Bottom: the construction of $\mathcal{L}_1$ graph using greedy solver.

are: Iris (NMI=0.5106, ACC=0.72); Soybean (NMI=0.6919, ACC=0.5244).

Euclidean Distance Ranking vs. Diffusion Ranking. The Euclidean distance ranking appears to have better clustering performance than that of diffusion ranking in general. This is rather a surprising result to us. Only for "Iris" dataset, the result of diffusion ranking is better than that of Euclidean distance ranking.

Running Time. We report the running time of generating $\mathcal{L}_1$ graphs using different construction algorithms. As we can see from Fig. 4.5, the Greedy-$\mathcal{L}_1$ graphs have consumed significantly less construction time than that in original $\mathcal{L}_1$ graphs.

Graph Sparsity. We check the sparsity of graphs by calculating the edge density:

$$Sparsity(G) = \frac{|E|}{|V| * (|V| - 1)}. \tag{4.9}$$

The results are reported in Table 4.5. We can see that Greedy-$\mathcal{L}_1$ graphs with diffusion distance ranking are more sparse than that with Euclidean distance ranking.

4.4.2 Large-sized Data and Multiple Classes Data

In this section, we present the experiment results of three large datasets. To keep the integrity of our experiments, two multiple classes data are also examined.

Name	#samples	#attributes	#clusters
ISOLET	1560	617	25
YaleB	2414	1024	38
MNIST4K	4000	784	9
COIL100	7200	1024	100
USPS	9298	256	10

Table 4.6: The statistics of three large datasets and two multiple classes datasets.

Datasets. We select following datasets for our experiments. Three large size datasets are: first 2k testing images of MNIST (MNIST4K), COIL 100 objects database (COIL100) and USPS handwritten digit database (USPS). Two multiple classes datasets are: isolet spoken letter recognition dataset (ISOLET), extended Yale face database B (YaleB). The statistics of selected datasets can be described by Table (4.6).

Spectral Clustering Performance. The spectral clustering performance shows in Table (4.8). As we can see, Gaussian graphs have overall better performance than different $\mathcal{L}_1$ graphs. For the performance between original $\mathcal{L}_1$ graph (with OMP greedy solver) and Greedy $\mathcal{L}_1$ graphs, the greedy version is better.

Graph Sparsity. We also check the sparsity of different similarity graphs. The result in Table (4.9) shows that Greedy $\mathcal{L}_1$ graphs with diffusion ranking are more denser than other $\mathcal{L}_1$ graphs. And the ordinary $\mathcal{L}_1$ graph (OMP) has the lowest sparsity.

It is known that the sparsity of graph will affect the performance of graph cut and then to spectral clustering. And the spectral clustering performance will drop if the sparsity is lower than a threshold [72]. Since $\mathcal{L}_1$ graph is a sparse graph in nature, we want to know the correlation between the sparsity and clustering performance. To evaluating this, we choose the "USPS" dataset, and generating graphs with different sparsity by setting the reconstruction approximation error bound to different thresholds. They are: [0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001]. For the size of "ranked dictionary", we choose size to $2M$ which is 512. The trend of spectral clustering performance with different sparsity can be show as the left subplot of Figure (4.6). We can see that when the sparsity value lower than 0.0072 , the spectral clustering performance drop catastrophically. The relationship between the approximation error and the graph sparsity is presented at the right side of Figure (4.6). By reading from the curve, we know that the ap-

Name	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
ISOLET	0.2571	0.7821	0.5501	0.4202	NA	0.1903	0.2993	NA
YaleB	0.2349	0.4219	0.2493	0.2895	NA	0.2003	0.4408	NA
MNIST4K	0.2503	0.4426	0.2679	0.1850	0.2438	0.0737	0.0333	0.0575
COIL100	0.3556	0.7726	0.7072	0.6533	0.6283	0.4044	0.4166	0.4788
USPS	0.1585	0.6580	0.6608	0.6571	0.6488	0.0360	0.0621	0.0399
Average	0.2513	**0.5457**	0.4713	0.4462	0.5070	0.1809	0.2504	0.1921

Table 4.7: NMI results of spectral clustering with different similarity graphs. M is the number of attributes.

NAME	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
ISOLET	0.2038	0.6974	0.4205	0.3327	NA	0.1705	0.2558	NA
YaleB	0.1533	0.2618	0.2067	0.2606	NA	0.1831	0.4321	NA
MNIST4K	0.2787	0.5302	0.3900	0.2755	0.3538	0.1847	0.1685	0.1845
COIL100	0.1192	0.5201	0.4746	0.4368	0.4012	0.2381	0.2326	0.2778
USPS	0.2122	0.7018	0.6723	0.6740	0.6950	0.1590	0.1778	0.1663
Average	0.1934	**0.5423**	0.4328	0.3959	0.4833	0.1871	0.2534	0.2095

Table 4.8: ACC results of spectral clustering with different similarity graphs. M is the number of attributes.

Name	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
ISOLET	0.0010	1	0.3304	0.2679	NA	0.4288	0.2804	NA
YaleB	0.0019	1	0.1968	0.1713	NA	0.3797	0.1952	NA
MNIST4K	0.0043	1	0.1022	0.0954	0.0929	0.1470	0.1267	0.1076
COIL100	0.0002	1	0.0786	0.0620	0.0574	0.1887	0.1198	0.0929
USPS	0.0003	1	0.0076	0.0072	0.0071	0.0246	0.0225	0.0214

Table 4.9: Graph sparsity results of different similarity graphs. M is the number of attributes.

Name	$\mathcal{L}_1$ (OMP)	Gaussian	Greedy-$\mathcal{L}_1$ graph (Euclidean)			Greedy-$\mathcal{L}_1$ graph (Diffusion)		
			K=1*M	K=2*M	K=3*M	K=1*M	K=2*M	K=3*M
ISOLET	243.9	1.1	202.5	310.6	NA	263.0	327.7	NA
YaleB	836.1	4.3	758.7	1187.6	NA	1097.9	1197.7	NA
MNIST4K	1435.8	9.8	814.8	1048.5	1341.9	848.9	1158.4	1412.7
COIL100	5541.3	36.1	2379.7	3225.0	5447.8	4108.5	5091.8	7475.3
USPS	2499.5	16.4	93.2	123.1	174.1	221.1	259.5	323.1

Table 4.10: Running time of different similarity graphs. M is the number of attributes.

proximation error and sparsity has a negative relationship. To maintain the Greedy $\mathcal{L}_1$ as dense as possible, we need to set a lower bound of approximation error.

Running time. We also record the running time of building different similarity graphs. From table (4.10), we see that the running time increase while

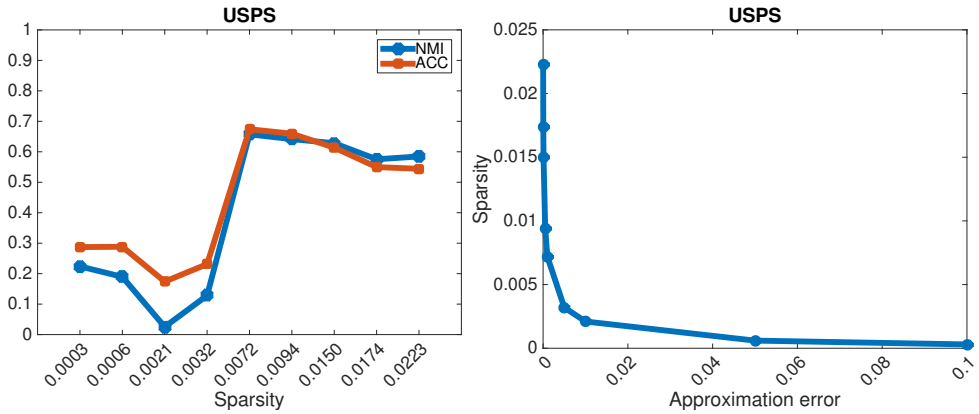


Figure 4.6: The impact of graph sparsity to spectral clustering performance. Left: graph sparsity vs. NMI and ACC. Right: $\mathcal{L}_1$ solver approximation error vs. graph sparsity.

the data size becomes larger. However, the "USPS" has lesser running time than "COIL100" even its data size is bigger. The reason is that "USPS" has smaller number of features than "COIL100" and this cause the $\mathcal{L}_1$ solver to need more computation time for finding sparse solutions.

4.5 Chapter Summary

In this chapter, we have designed a greedy algorithm to construct $\mathcal{L}_1$ graph. Moreover, we introduced the concept of "ranked dictionary", whcih not only preserves the locality but also solve the curse of normalization. Moreover, it can construct $\mathcal{L}_1$ graph efficiently for large size data (#instances $\geq$ 3000.) Except for the Euclidean metric and diffusion metric that have been discussed in this paper, the user can choose other ranking methods such as manifold ranking that could be more appropriate for specific dataset in real applications. Our greedy algorithm can generate sparse $\mathcal{L}_1$ graph faster than the original $\mathcal{L}_1$ graph construction algorithm, and the resulting graphs have better clustering performance on average than original $\mathcal{L}_1$ graph. Nevertheless, our algorithm could be generalized in a straightforward way by introducing regularization terms such as elastic net into the current solver, which would indicate the quality of generated $\mathcal{L}_1$ graphs could be further improved.

Chapter 5

Dense Subgraph based Multi-source Data Integration

In this chapter, we propose a multi-source data integration framework based on dense subgraph mining techniques. It is applied to integrate Microarray experiment data from different sources in computational biology research area. The goal is to solve the so-called "batch effect" between different experiments. Ratio-based algorithms are proven to be effective methods for removing batch effects that exist among microarray expression data from different data sources. They are outperforming than other methods in the enhancement of cross-batch prediction, especially for cancer data sets. However, their overall power is limited by: (1) Not every batch has control sample. The original method uses all negative samples to calculate subtrahend. (2) Microarray experimental data may not have clear labels, especially in the prediction application, the labels of test data set are unknown. In this chapter, we propose an Improved Ratio-Based (IRB) method to relieve these two constraints for cross-batch prediction applications. For each batch in a single study, we select one reference sample based on the idea of aligning *probability density functions (pdfs)* of each gene in different batches. Moreover, for data sets without label information, we transfer the problem of finding reference sample to the dense subgraph problem in graph theory. Our proposed IRB method is straightforward and efficient, and can be extended for integrating large volume microarray data sets. The experiments show that our method is stable and has high performance in tumor/non-tumor prediction.

5.1 Chapter Introduction

In this digital era, we have been obtaining much more biological experiment data than before. Consequently, biological scientists have collected and built many genomic knowledge database by taking the advantage of today's information technology. These large database, for example, NIH GEO [73], inSilicoDb [74], and ArrayExpress [75], not only share many experiments data from different independent studies, but also provide computing tools for researchers to analyze data. The approach of integrative analyzing multiple microarray gene expression datasets is proved to be a robust way for extracting biological information from genomic datasets [76]. Comparing with "meta-analysis" [77] which combines analysis results from many small-sized independent datasets, integrative analysis shows higher statistical relevance of results from one integrated large size dataset [78]. Nevertheless, combining or merging microarray expression data from different data sources suffers from the so-called *batch effects* [79] which is still a challenging and difficult problem to be solved in computational biology nowadays.

Batch effects are different from bias and noise. They are systematical unwanted variations existing among batches from different sources [79]. Many research works have been proposed in past decade to learn their math properties, and to reduce its impacts in microarray data analysis. Lazar et al. [78] documented a comprehensive survey about existing batch effect removal methods. In all those methods, ratio-based methods are proved to have high prediction performance by Luo et al. [80]. Moreover, ratio-based methods have low computational cost which is demanding for integrating large volume data sets. However, ratio-based methods require each batch of data to have a group of *reference samples*, which could be either control samples or negative (non-tumor) samples.

GENESHIFT is another batch effect removal method proposed by Lazar et al. [81]. It is a nonparametric algorithm and assumes that samples in different batches are from same population, which means they will have same distributions. By this assumption, GENESHIFT reduces the batch effect by aligning the *pdfs* of each gene's expression values crossing different batches. It has same expression value model as ratio-based methods. However, It does not have a clear math operation/definition about how the batch effects are neglected or removed. In this chapter, we propose an *Improved Ratio-based(IRB)* method of batch effect removal by taking the advantages of ratio-besd methods and GENESHIFT. The main contributions of our works are listed as follows:

- We show that it is better if the *pdfs* of genes are estimated from negative (non-tumor) samples instead of all samples for cancer data sets(§ 5.4.3).
- We propose a co-analysis framework (§ 5.4.4) to find reference samples

Symbol	Meaning
X^k	X: one batch; k: batch id;
X_{ij}^k	expression value of i_{th} row and j_{th} column;
$\hat{X}_{ij}^k$	expression value after batch effect removal;
b_{ij}^k	batch effect of value at (i, j) in batch k;
ϵ_{ij}^k	noise;
P_i, Q_i	*pdfs* of gene i in batch P and Q;
$G(V, E)$	graph G with vertices V and edge set E;
S	vertices of subgraph;
$e[S]$	number of edges induced by S;

Table 5.1: Frequent math notations.

for ratio-based algorithms. We define *matching score* for searching best reference samples for labeled data samples. We also propose a greedy algorithm for obtaining the local optimal solution.

- For unlabeled data samples, we transfer the reference samples searching problem to the dense subgraph problem from graph theory (§ 5.4.4) and design an searching algorithm based on bipartite graph to solve it.
- We propose an improved ratio-based method (IRB) (§ 5.4.5) by using one sample in each batch as subtrahend comparing to original method which use many. We also evaluate the prediction performance over two real cancer data sets.

In this work, we represent different batch data as $X^k, k \in \{1, \cdots, K\}$, where k is the batch ID. Each batch data has m rows and n columns. The rows represent genes(feature), and the columns represent samples. Moreover, we assume that all batches have been log-transformed and preprocessed for background correction, normalization and summarization by using either MAS5 [82], RMA [83], fRMA [84] or other preprocessing tools.

5.2 Related Works

Batch effect removal. Survey [85], [78] give detail comparison and analysis about existing batch effect removal algorithms. The most popular ones are(not

limited to): Batch Mean-Centering(BMC) [86],Gene Standardization [87],Ratio-based methods [80], Scaling relative to reference dataset [88], Empirical Bayes method [89], Cross-Platform Normalization(XPN) [90], Distance-Weighted Discrimination [91], singular value decomposition based method [92], surrogate variable analysis [93],GENESHIFT [81], remove unwanted variation 2-step [94] and etc. These methods can be separated into two groups: location-scale (LS) methods and matrix-factorization(MF) methods. LS methods assume a statistical model for the location (mean) and scale (variance) of the data within the batches and proceed to adjust the batches in order to agree with these methods. MF algorithms assume that the variation in the data corresponding to batch effects is independent to the biological variable of interest and it can be captured in a small set of factors which can be estimated through some matrix factorization methods.

Ratio-based methods. Ration-based methods [80] shift the expression value of each gene based on a set of reference samples in each batch. It is designed with two versions: *Ratio-A* and *Ratio-G*. *Ratio-A* uses arithmetic mean value as subtrahend while Ratio-G uses geometric mean value. They assume that expression value of each gene in reference samples are subjected to the same amount of batch effect as in the other samples in same batch. Then the batch effect can be removed by subtracting the mean of those reference samples. Assuming that there are r reference samples in batch X^k, method *Ratio-A* and *Ratio-G* can be described as:
Ratio-A: Arithmetic mean ratio-based method:

$$\hat{x}_{ij}^k = x_{ij}^k - \frac{1}{r}\sum_{l=1}^{r} x_{il}^k \tag{5.1}$$

Ratio-G: Geometric mean ratio-based method:

$$\hat{x}_{ij}^k = x_{ij}^k - \sqrt[r]{\prod_{l=1}^{r} x_{il}^k} \tag{5.2}$$

GENESHIFT is a high quality nonparametric method. It first estimates genewise *pdfs* for each batch using the Parzen-Rosenblatt density estimation method [95]. Secondly, it estimates the offset term by finding the best match between two *pdfs*. This algorithm processes two batch data at one time. Assume P_i and Q_i are the *pdfs* of gene i in studies of batch X and batch Y. The algorithm put P_i as being fixed, and slides Q_i step by step across the range where P_i is estimated. In each step, the algorithm computes the inner product between P_i and part of Q_i, which lays in the range where the densities are

estimated as follows:

$$M(t) = P_i * Q_i = \sum_{j=1}^{d} P_i(j) W^t_{Q_i(j)} \tag{5.3}$$

where d is number of sampling ticks of *pdf* and $W^t_{Q_i(j)}$ is given by:

$$W^t_{Q_i(j)} = \begin{cases} \omega Q^t_i, & \text{for } Q^t_i \text{ in window} \\ 0, & \text{otherwise} \end{cases}$$

with $\omega = 1$ a rectangular window defined on the support of P_i and Q^t_i is part of Q_i found in the *pdfs* estimation range at step t. The best matching between P_i and Q_i is given by $max(M)$ and the offset term is obtained by subtracting from the initial position of $Q_i(b_{ref})$, the best matching position $(b_{max(M)})$ is:

$$\delta = b_{ref} - b_{max(M)}$$

By setting the reference position to 0, the offset term becomes $\delta = -b_{max(M)}$.

Dense subgraph Dense subgraph extraction is a classic problem in Graph theory [96]. The algorithms of solving this problem have been applied to biological networks research [97] [98] [99]. Here, we want to extract a densest subgraph from defined bipartite graph. We wish the extracted subgraph has high quality and concise. To archive this goal, we apply the latest technique described in [100] to extract the *optimal quasi-clique* which is a high quality dense subgraph.

Given a graph $G(V, E)$, find a subset of vertices $S^* \subseteq V$ such that $f_\alpha(S^*) = e[S] - \alpha\binom{|S|}{2} \leq f_\alpha(S)$ for all $S \subseteq V$. The resulted set S^* is called *optimal quasi-clique* of G. We use the recommend value $\alpha = 1/3$ in this chapter.

5.3 Data

We use two real world cancer data sets to validate our proposed algorithms.

Lung cancer dataset The lung cancer dataset consists three data sets hybridized on two different Affymetrix platforms. The first lung cancer data set (GSE19804) contains 120 samples of tumor and adjacent normal tissue samples hybridized on Affymetrix HGU133plus2 expression arrays. The second data set (GSE19188) contains 94 tumor and 62 adjacent normal tissue samples hybridized on Affymetrix HGU133plus2 expression arrays. The third lung cancer data set (GSE10072) contains 58 tumor samples and 49 normal tissues

samples consists of a mix of independent controls and tumor adjacent tissues hybridized on Affymetrix HGU133A expression array.

Type	Name	NT	T	Platform
Train	GSE19804	60	60	GPL570
	GSE19188	62	94	GPL570
Test	GSE10072	49	58	GPL96

Table 5.2: Lung cancer dataset. **NT**: non-tumor, **T**: lung tumor.

Iconix dataset We use the Iconix dataset (GSE24417) from Microarray Quality Control Phase II(MAQC-II) microarray gene expression data [80]. The Iconix dataset is a toxicogenomic data set provide by Iconix Bioscience (Mountain View, CA, USA). It aimed at evaluating hepatic tumor induction by non-genotoxic chemicals after short-time exposure. The training set consists of 216 samples treated for 5 days with one of 76 structurally and mechanistically diverse non-genotoxic hepatocarcinogens and non-hepatocarcinogens. The test set consists of 201 samples treated for 5 days with one of 68 structurally and mechanistically diverse non-genotoxic hepatocarcinogens. Gene expression data were profiled using the GE Codelink microarray platform. The separation of the training set and the test set was based on the time when the microarray data were collected, also the different batches. The detail data set information is listed as follows.

Type	Batch	NT	T	Date
Train	B1	17	24	11/6/01-12/10/01
	B2	87	17	12/11/01-02/25/02
	B3	39	32	3/20/02-7/18/02
Test	B4	91	18	07/22/02-12/4/02
	B5	53	39	4/3/03-9/28/04

Table 5.3: Information of the Iconix dataset; **NT**: non-tumor, **T**: tumor.

5.4 Algorithm

In this section, we are presenting the Improved Ratio-based (IRB) method. Comparing to the original ratio-based method, we solve the problem of finding reference samples. Instead of finding reference samples in each batch separately, IRB selects reference samples by taking all batches into consideration at the same time. The outline of this section is as follows. Firstly, the expression value model of microarray data sets are defined. Secondly, we define the reference samples searching problem formally. Thirdly, we describe the assumption used in our method. In the last, we introduce a co-analysis framework for finding reference samples in labeled and unlabeled data sets separately.

5.4.1 Expression Value Model

In general, batch effect comes with multiplicative and additive form. After log-transform, these batch effects are both represented as additive terms. We assume that the expression value of feature i in sample j of batch X^k can be expressed in the following general form:

$$x_{ij}^k = x_{ij}^{'} + b^k + \epsilon_{ij}^k \tag{5.4}$$

where $x_{ij}^{'}$ is the actual feature value. b^k is the batch effect term and ϵ_{ij}^k represents noise.

Moreover, we use the same genewise density estimation method as GENESHIFT algorithm which is Parzen-Rosenblatt density estimation method[95].

5.4.2 Problem Definition

As we mentioned before, we only want to find one reference sample for each batch. The searching guideline is following the philosophy of GENESHIFT algorithm: *the inner product of each gene's pdf in different batches are maximized after integration.* Before giving the formal definition of our problem, we first define the matching score of two batches:

Definition 2. *Given two batches that have same number m of genes(or features), and with pdf P and pdf Q respectively, the matching score of them is defined as:*

$$M(P,Q) = \sum_{i=1}^{m} < P(i), Q(i) > \tag{5.5}$$

Now, our problem can be defined formally as:

Problem 1. *Given K batches of microarray expression dataset $X^k : m \times n(k)$,m genes,$n(\cdot)$ samples,$k \in \{1, \cdots, K\}$, with estimated pdfs:*

$$\boldsymbol{P} = [P^1(\boldsymbol{x}), P^2(\boldsymbol{x}), \cdots, P^K(\boldsymbol{x})]$$

where P^k is the vector of pdfs for genes $\boldsymbol{x}$ in batch X^k. P^k is a $m \times 1$ vector where each $P_i^k, i \in \{1, \cdots, m\}$ represents the pdf of i_{th}gene.The problem is to find K offset samples x_{offset}^k within each batch respectively:

$$\boldsymbol{x}_{offset} = [\boldsymbol{x}_{offset}^1, \boldsymbol{x}_{offset}^2, \cdots, \boldsymbol{x}_{offset}^K,]$$

such that the total matching score of pdfs after shifting by its offset samples respectively archives maximum:

$$max_{\boldsymbol{x}_{offset}} \sum_{i=1}^{K} \sum_{j \neq i, j=1}^{K} M(P^i(\boldsymbol{x} - \boldsymbol{x}_{offset}), P^j(\boldsymbol{x} - \boldsymbol{x}_{offset})) \tag{5.6}$$

In the above problem, x_{offset}^k is a specific sample in batch k. If we don't limit x_{offset}^k to be a specific sample in the batch and let it be a regular offset vector, the problem 1 can be seen as a generalized version of GENESHIFT which takes two batches at the same time and shift *pdfs* of every gene separately from one batch to another batch. The reason we put this constrain here is that the batch effect term b^k in equation (5.4) can be neglected by subtracting a sample, and this sample inherits the batch effect term with its true signal value. The advantage of applying this constrain is that we obtain a clear math explanation about how the batch effects are removed.

5.4.3 Assumption

In GENESHIFT, the author assumes that the expression of each gene from two different experiments(batches) can be represented accurately enough through the expression of that gene across all population if the number of samples in two microarray Gene Expression(MAGE) experiments is sufficiently high. By this assumption, a consequence conclusion is that the *pdfs* of each gene should be similar in all experiments. However, as we observed from above cancer data sets, the average similarity among the non-tumor(negative) samples is higher than the tumor(positive) samples, as show by Figure 5.1. We then argue that the similarity *pdfs* assumption of GENESHIFT holds for cancer data sets only if the *pdfs* are estimated from non-tumor samples but not from all. This argument is not only based on the observation but also based on the fact

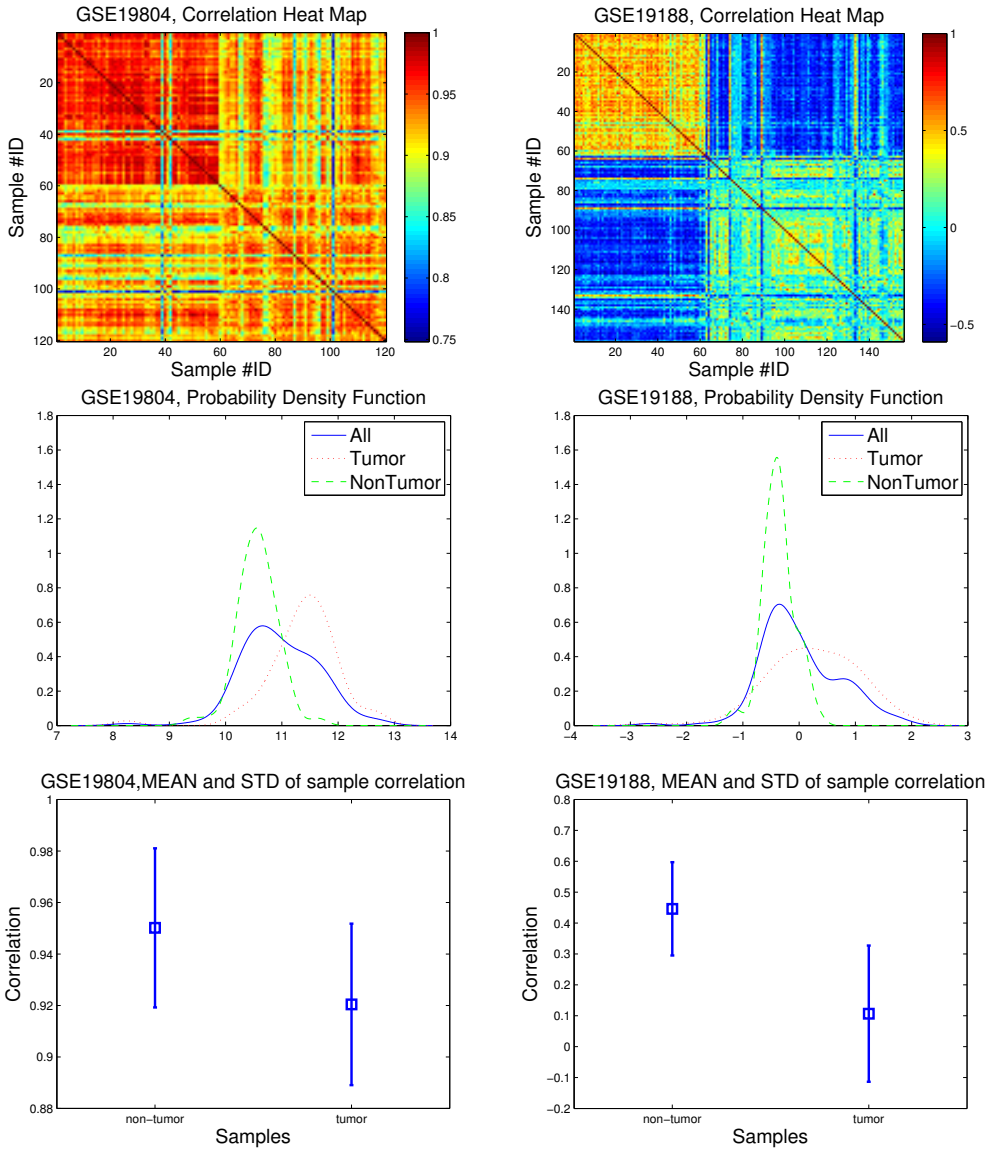


Figure 5.1: Left: GSE19804; Right: GSE19188; Top row: correlation (PCC) heat map, samples are sorted from non-tumor to tumor samples; Middle row: *pdf* of a random gene (GeneBank ID:U48705). Bottom row: correlation values distribution.

that tumors with similar histopathological appearance can follow significantly different clinical courses [101]. The assumption of IRB now can be described as following:

Assumption 1. *The pdf of a gene have similar distribution in all experiments*

iif the pdf is estimated from non-tumor samples.

5.4.4 Co-analysis Framework

In this section, we propose a co-analysis framework to find the reference samples both for labeled and unlabeled data samples. For all ratio-based methods, we need reference samples to calculate the subtrahend. Original ratio-based methods use average of all negative samples or median of them. As for our method, we only use one reference sample for each batch. Comparing to the original ratio-based methods that find reference sample independently, we take all batches into consideration at the same time. Our co-analysis framework can be described as following from labeled data sets to unlabeled data sets.

Labeled data sets. For example, the training data sets have clear labels of samples. To find the reference samples for them, we need to solve the optimization problem (1). However, the properties like convexity or non-convexity of objective function in problem (1) are uncertain. Because (1)the objection function cumulates all matching scores of genes that show very different *pdfs*;(2)the *pdf* curve could be either convex or non-convex.

To solve this problem, we propose a greedy algorithm for it. Our algorithm first select an *anchor* batch that has the largest number of non-tumor samples and shift its geometric median to axis origin. Secondly, for rest batches, we calculate the best offset vector for each of them according to this *anchor* batch. In the last step, we search a sample inside each batch that has the smallest euclidean distance to this offset vector and treat it as the reference sample that we are looking for. In the first step, we shift the geometric median of *anchor* batch to axis origin. The reason is that we want to place the median of *pdfs* of all genes around the axis origin as much as possible. However, the geometric median is not only difficult to compute but also not necessary to be an experiment sample that inherits batch effect. To solve this dilemma, we choose the sample that nearest to the geometric median as a substitute. We call this sample *approximate geometric median(GM) sample*: GM_{approx}. and the definition is as:

$$GM_{approx} = \arg\min_{y \in X} \sum_{x_j \in X \setminus y} \|x_j - y\|_2 \tag{5.7}$$

where the parameter δ controls the width of neighborhoods. Our greedy algorithm now can be described as Algorithm 7.

Algorithm 7: FindingReferenceSampleLabeled

input : Microarray experiments data: $X^k : m \times n, k \in \{1, \cdots, K\}$ with labels.

output: Reference samples: $\boldsymbol{x}_{offset} = [\boldsymbol{x}^1_{offset}, \boldsymbol{x}^2_{offset}, \cdots, \boldsymbol{x}^K_{offset},]$.

```
begin
    Find anchor batch x^anchor;
    Shift x^anchor by GM_approx;
    for batch X^k, k ≠ anchor do
        for each gene g_i, i ∈ {1, ··· , m} do
            estimates the pdf across batches: pdf_i^k;
            calculate the offset term δ_i^k;
        end
        find the closest sample x̂^k_offset to δ^k;
    end
end
```

Unlabeled data sets. For these data sets, the tumor/non-tumor labels are unknown but the batch labels are clear. We estimate the non-tumor samples of a unlabeled batch by using dense subgraph extraction algorithms. We first build a bipartite similarity graph between the known non-tumor samples and all unlabeled samples. The pearson correlation coefficient(PCC) metric, represented as $sim(\cdot)$, is used. After that, we extract a dense subgraph, called *optimal quasi-clique*, from the built graph. The nodes of the resulted subgraph that belong to the unlabeled side are treated as non-tumor samples. The algorithm of building the bipartite graph is described by algorithm 8.

The user-specific value θ will affect the output of our algorithm as the input is a completed weighted graph. In our experiments, we use the value that equals to half of the highest similarity value.

We use the *GreedyOQC* algorithm introduced in [100] to extract the *optimal quasi-clique*. An illustration of the algorithm output is as following:

5.4.5 Improved Ratio-based Method

Once we have reference sample for each batch, it's straightforward to modify the original ratio-based method and obtain our proposed IRB method as following:

$$\hat{x}^k_{ij} = x^k_{ij} - x(i)_{offset} \tag{5.8}$$

The overall **IRB** algorithm can be described by algorithm 9.

Algorithm 8: BuildBipartiteGraph

input : Non-tumor samples: L, unlabeled samples: R, User specified threshold θ

output: A unweighted undirected bipartite graph $G(V, E)$, where $L, R \subseteq V$.

```
1  begin
2      Calculate the similarity sim(l, r),where l ∈ L, r ∈ R;
3      for each pair (l, r) do
4          if sim(l, r) ≥ θ then
5              add one edge to E for nodes pair (l, r);
6          end
7      end
8      remove the nodes with zero degree;
9      return G(V, E);
10 end
```

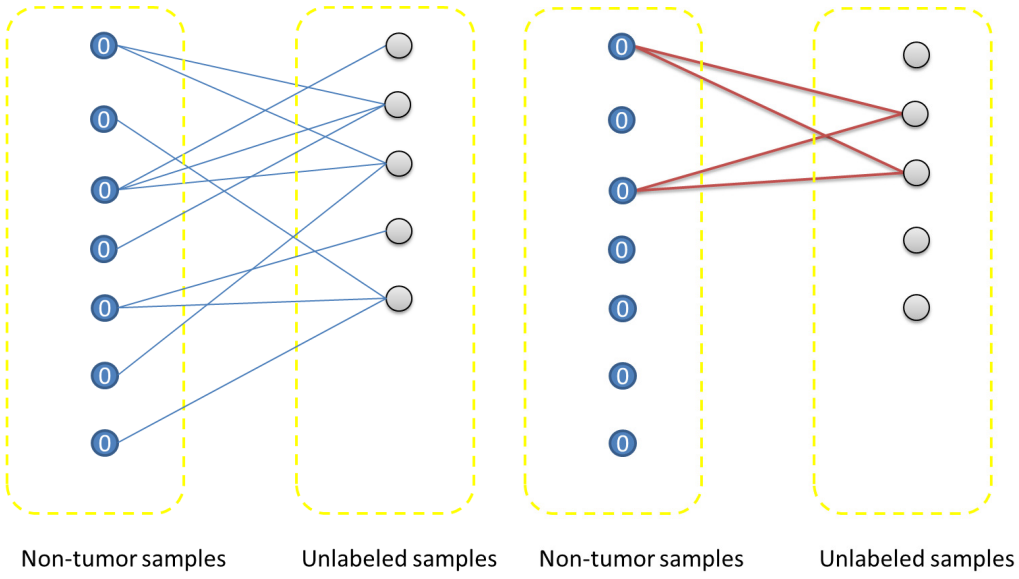


Figure 5.2: Left: Input bipartite graph; Right: extracted *optimal quasi-clique*; Blue nodes: known non-tumor samples; Gray nodes: unlabeled samples.

5.5 Validation

In this section, we demonstrate and validate our proposed co-analysis framework by using the Lung cancer dataset. Results of each step are presented here to better show the details of our proposed algorithm.

Algorithm 9: IRB

input : labeled data sets: $X^k, k \in \{1, \cdots, K\}$ with labels;
unlabeled data set: Y;
output: data sets with batch effect removed: $\hat{X}^l$ and $\hat{Y}$;

begin
  FindingReferenceSampleLabeled(X), obtain $\boldsymbol{x}_{offset}$;
  Shift all X by $\boldsymbol{x}_{offset}$, obtain $\hat{X}$;
  BuildBipartiteGraph(X,Y) and extract*optimal quasi-clique*;
  Estimate the offset of Y;
  Find reference sample $(y)_offset$;
  Shift Y;
end

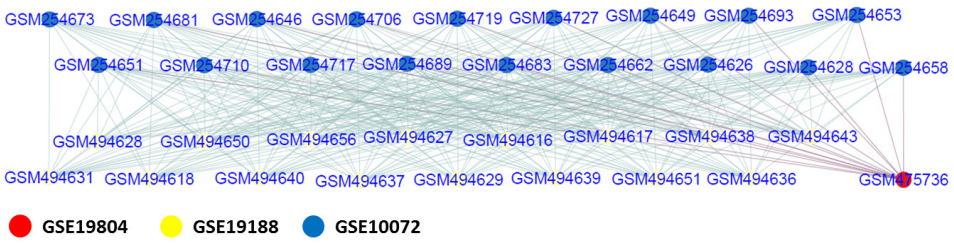


Figure 5.3: Resulted *optimal quasi-clique* of Lung cancer dataset.$G = (|V| = 35, |E| = 287)$. The top two rows list the estimated (fake) non-tumor samples found by *GreedyOQC*.

For Lung cancer dataset, we have three batches from two different gene chip platforms. The batch GSE19188 is selected as *anchor batch* since it has the largest number of non-tumor samples. The approximate geometric median sample is GSM475732. The difference of *pdf* before and after shifting (applying IRB method) shows as Figure 5.4.

Now we calculate the reference sample for second batch GSE19804 according to *anchor batch* and the changing of *pdf* is as figure 5.5.

For test data GSE10072, we build the bipartite graph and find the resulted *optimal quasi-clique* as figure 5.5. The constructed bipartite graph has 173 nodes and 747 edges. The output *optimal quasi-clique* shows as figure 5.5 and it has 35 nodes and 287 edges. Among them, 18 nodes are samples of GSE10072 and the real labels of them are non-tumor samples. The changes of *pdfs* of GSE10072 is as figure 5.6.

To check the quality of batch effect removal, we show the correlation heat map and clustering dehendragraph here. As we can see, the correlation values

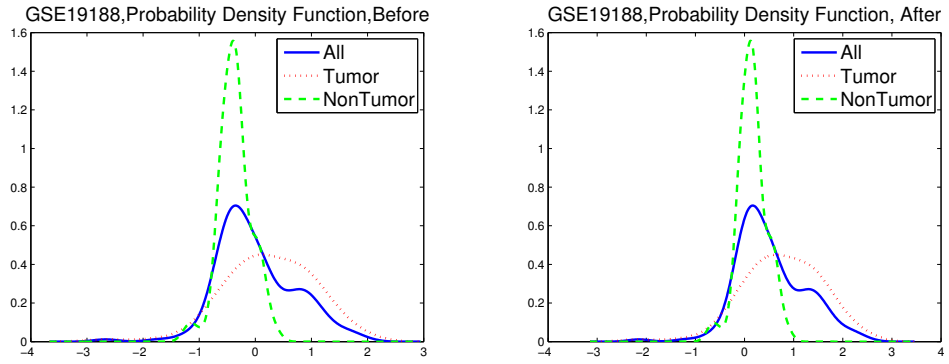


Figure 5.4: Difference of gene U48705 *b*efore (left) and after (right) applying IRB by reference sample GSM475732.

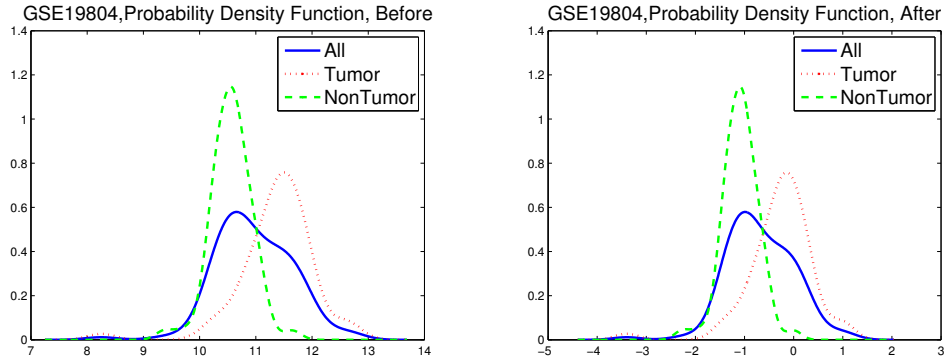


Figure 5.5: The *pdf* difference of gene U48705. *pdf* before (left) and after (right) applying IRB.The value offset is -10.4113.

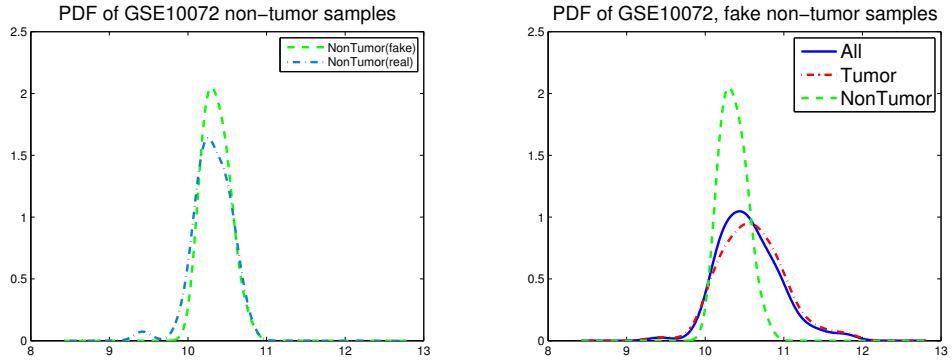


Figure 5.6: The *pdf* of GSE10072 by estimated(fake) non-tumor samples

among different batches are enhanced and more smooth. The correlation heat map before and after batch effect removal is:

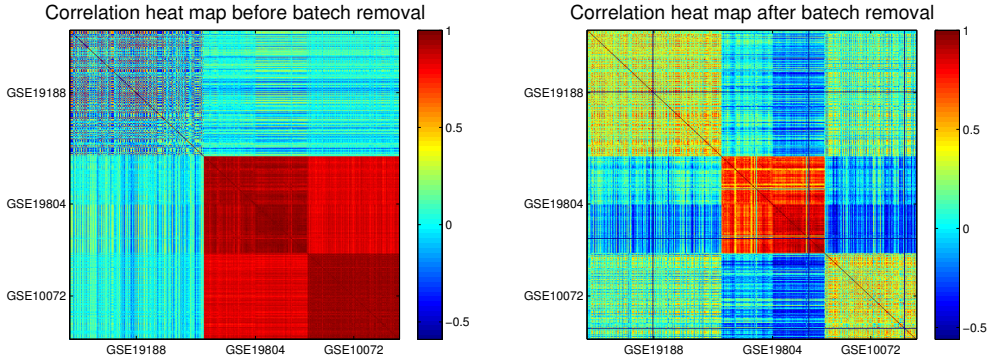


Figure 5.7: Correlation heat map of Lung cancer data. Top: original data. Bottom: after batch effect removal by IRB.

5.6 Experiments

In this section, we examine the prediction performance of our proposed algorithm comparing to original ratio-based methods and GENESHIFT. We use *Support Vector Machine*(SVM) algorithm with penalty $C = 1$, which is the setting in [80] except that we omit feature selection here. Accuracy and *Matthews correlation coefficient(MCC)* are used for our measurements.

The prediction performance of Lung Cancer data is summarized by following table: As the results show, GENESHIFT has the best prediction accuracy

Table 5.4: Prediction performance of Lung cancer dataset

Classifier	Method	Accuracy	MCC
SVM(C=1)	ratio-G	0.45	0.7829
SVM(C=1)	ratio-A	0.9629	**0.9813**
SVM(C=1)	GENESHIFT	**0.9723**	0.9803
SVM(C=1)	IRB	0.9623	**0.9813**

but ratio-A and IRB have the better MCC scores.

Also, We compare the prediction performance of Iconix data set in table 5.6. The results show that IRB obtain the best accuracy and MCC scores.

By above two experiment results, we can see that IRB method always has higher prediction performance than others. This means that IRB is a stable batch effect removal algorithm.

Table 5.5: Prediction performance of Iconix dataset

Classifier	Method	Accuracy	MCC
SVM(C=1)	ratio-G	0.72	0.1
SVM(C=1)	ratio-A	0.71	0.01
SVM(C=1)	GENESHIFT	0.68	0.04
SVM(C=1)	IRB	**0.73**	**0.15**

5.7 Chapter Summary

Batch effect removal has been a challenging research problem in computational biology while integrating large volume microarray data sets. In the past, we neither had a clear mathematical description of this problem, nor had an unique way to evaluate the performance of batch effect removal. In this work, we have generalized the idea of GENESHIFT, which is the latest batch effect removal algorithm and a non-parametric method.

Our contribution is tow-fold. First, we have solved the problem of finding reference samples for ratio-based methods from labeled data sets to unlabeled sets. The proposed co-analysis framework aligns the density function of non-tumor samples of each batch as much as possible. Comparing with the original ratio-based method which processes the batch effect less adequately, our framework takes all batches into consideration at the same time. Moreover, we applied the latest algorithm for dense subgraph problem from graph theory to solve the problem of finding reference samples for unlabeled data sets. The motivation of using the graph algorithm is that the non-tumor samples are much more similar to each other than tumor samples.

Second, our algorithm has the advantage of lowering the computational cost of both ratio-based method and GENESHIFT method. Comparing with several other batch effect removal methods, this property is valuable while integrating large volume of microarray datasets. The *GreedyOQC* has complexity $\mathrm{O}\big(|V| + |E|\big)$ for graph $G(V, E)$.

In summary, IRB solves the reference sample finding problem of the original ratio-based method. It inherits the characteristic of GENESHIFT that has little negative impact on the data distortion (only on samples). As a non-parametric method, it is stable and has high performance in prediction applications for cancer data sets. It has low computational cost and can be easy adapted to large volume data applications.

Chapter 6

Mining Robust Local Subgraphs in Large Graphs

Robustness is a critical measure of the resilience and performance of large networked systems. Most works study the robustness of graphs as a whole. In this chapter, we focus on local robustness and pose a novel problem in the line of subgraph mining: given a large graph, how can we find its most robust local subgraphs (RLS)? Robust subgraphs can be thought of as the anchors of the graph that hold together its connectivity and find several applications.

Our problem formulation is related to the recently proposed general framework [6] for the densest subgraph problem, however differs from it substantially as robustness concerns with the placement of edges, i.e. the subgraph topology, as much as the number of edges in a subgraph. We offer the following contributions: (*i*) we show that our RLS-Problem is **NP**-hard and analyze its properties, (*ii*) we propose two heuristic algorithms based on top-down and bottom-up search strategies, (*iii*) we present simple modifications of our algorithms to handle three variants of the original RLS-Problem. Extensive experiments on many real-world graphs demonstrate our ability to find subgraphs with higher robustness than the densest subgraphs [5, 6] even at lower densities, suggesting that the existing approaches are not as suitable for the new robust subgraph mining setting.

6.1 Chapter Introduction

Large complex networked systems, such as transportation and communication networks, are a major part of our modern world. The performance and function of such complex networks rely on their structural robustness, which is their ability to retain connectivity in the face of damage to parts of the net-

work [102]. There are many quantitative metrics to measure the robustness of a network. However, among other desirable properties, it is crucial for a robustness measure to emphasize the existence of alternative or back-up paths between nodes more than just the shortest paths.

In this chapter, we adopt one such measure based on the reachability (phrased as the communicability) of the nodes in the network [103]. We then introduce a novel problem related to graph robustness: Given a large graph, which sets of nodes exhibit the strongest communicability among each other? In other words, how can we identify the most robust subgraphs in a large graph?

From the practical point of view, robust subgraphs can be considered as the "anchors" of the graph, around which others are connected. They likely form the cores of larger communities or constitute the central backbones in large networks, responsible for most of the connectivity. For example, robust subgraphs can correspond to strong communities in social and collaboration networks or robust regions in the power grid. Moreover, robust interaction patterns in biological networks can help the understanding of healthy and diseased functional classes, and in financial networks the strength and robustness of the market.

While the robust subgraph mining problem has not been studied before, similar problems have been addressed in the literature (§6.2). Probably the most similar to ours is the densest subgraph mining problem, aiming to find subgraphs with highest average degree [5, 104, 105] or edge density [6, 106]. However, density (whichever way it is defined) is essentially different from robustness mainly because while the former concerns with the number of edges in the subgraph, the topology is at least as critical for the latter (§6.3.1).

The main contributions of our work are the following:

- We formulate a new problem of finding the most robust local subgraphs (RLS) in a given graph. While in the line of subgraph mining problems, it has not been studied theoretically before (§6.3).
- We show that the RLS-Problem is **NP**-hard (§6.3.2), and further study its properties (§6.3.2).
- We propose two fast heuristic algorithms to solve the RLS-Problem for large graphs. One is a top-down greedy algorithm, which iteratively removes a node that affects the robustness the least. The other algorithm is a bottom-up solution based on a meta-heuristic called the greedy randomized adaptive search procedure (GRASP) [107] (§6.4).
- We define three practical variants of the RLS-Problem; finding (*i*) the most robust global subgraph without any size constraint, (*ii*) the top-k most robust local subgraphs, and (*iii*) the most robust local subgraph

containing a set of user-given seed nodes (§6.3.2). We show how to modify our algorithms proposed for the RLS-Problem to solve its variants (§6.4).
- We extensively evaluate our proposed solutions on numerous real-world graphs. Since our RLS-Problem is a new one, we compare our results to those of three algorithms (one in [5], two in [6]) that has been proposed for the densest subgraph problem. Our results show that we find subgraphs with higher robustness than the densest subgraphs even at lower densities, demonstrating that the existing algorithms are not compatible for the new problem setting (§6.5).

6.2 Related Works

Robustness is a critical property of graphs. Thus, it has been studied extensively in various fields including physics, biology, mathematics, and networking. One of the early studies in measuring graph robustness shows that scale-free graphs are robust to random failures but vulnerable to intentional attacks, while for random networks the difference between the two strategies is small [108]. This observation has stimulated studies on the response of networks to various attack strategies [109–114]. Other works look at how to design networks that are optimal with respect to some survivability criteria [115–118].

With respect to local regions, Trajanovski *et al.* aim to spot critical regions in a graph the destruction of which would cause the biggest harm to the network [119]. Similar works aim to identify the critical nodes and links of a network [120–123]. These works try to spot vulnerability points in the network, whereas our objective is somewhat orthogonal: identify robust regions. Closest to ours, Andersen *et al.* consider a spectral version of the densest subgraph problem and propose algorithms for identifying small subgraphs with large spectral radius [124].

While having major distinctions as we illustrated in this work, robust subgraphs are related to dense subgraphs, which have been studied extensively. Finding the largest clique in a graph, well-known to be **NP**-complete [125], is also shown to be hard to approximate [126].

A relaxation of the clique problem is the densest subgraph problem. Goldberg [105] and Charikar [5] designed exact poly-time and $\frac{1}{2}$-approximate linear-time solutions to this problem, respectively, where density is defined as the average degree. This problem is shown to become **NP**-hard when the size of the subgraph is restricted [127]. Most recently, Tsourakakis *et al.* [6] also proposed fast heuristic solutions, where they define density as edge surplus;

the difference between number of edges and α fraction of maximum edges, for user-specified constant $\alpha > 0$. Likewise, Pei *et al.* study detecting quasi-cliques in multi-graphs [128]. Other definitions include k-cores, k-plexes, and k-clubs, etc. [129].

Dense subgraph discovery is related to finding clusters in graphs, however with major distinctions. Most importantly, dense subgraph discovery has to do with absolute density where there exists a preset threshold for what is sufficiently dense. On the other hand, graph clustering concerns with relative density measures where density of one region is compared to another. Moreover, not all clustering objectives are based on density and not all types of dense subgraphs can be found by clustering algorithms [129].

In summary, while similarities among them exist, discovery of critical regions, robust subgraphs, cliques, densest subgraphs, and clusters are substantially distinct graph mining problems, for which different algorithms can be applied. To the best of our knowledge, our work is the first to consider identifying robust local subgraphs in large graphs.

6.3 Robust Local Subgraphs

6.3.1 Graph Robustness

Robustness is a critical property of large-scale networks, and thus has been studied in various fields including physics, mathematics, computer science, and biology. As a result, there exists a diverse set of robustness measures, e.g., mean shortest paths, efficiency, pairwise connectivity, etc. [130].

In this work, we adopt a spectral measure of robustness, called *natural connectivity* [131], written as

$$\bar{\lambda}(G) = \log(\frac{1}{n}\sum_{i=1}^{n} e^{\lambda_i}) \ , \tag{6.1}$$

which can be thought of as the "average eigenvalue" of G, where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ denote a non-increasing ordering of the eigenvalues of its adjacency matrix $\mathbf{A}$.

Among other desirable properties [110], natural connectivity is interpretable; it is directly related to the subgraph centralities (SC) in the graph. The $SC(i)$ of a node i is known as its communicability [103], and is based on the "weighted" sum of the number of closed walks that it participates in:

$$S(G) = \sum_{i=1}^{n} SC(i) = \sum_{i=1}^{n} \sum_{k=0}^{\infty} \frac{(\mathbf{A}^k)_{ii}}{k!} ,$$

where $(A^k)_{ii}$ is the number of closed walks of length k of node i. The $k!$ scaling ensures that the weighted sum does not diverge, and longer walks count less. $S(G)$ is also referred as the Estrada index [103], and has been shown to strongly correlate with the folding degree of proteins [132].

Noting that $\sum_{i=1}^{n}(\mathbf{A}^k)_{ii} = \text{trace}(\mathbf{A}^k) = \sum_{i=1}^{n} \lambda_i^k$ and using the Taylor series of the exponential function we can write

$$S(G) = \sum_{k=0}^{\infty} \sum_{i=1}^{n} \frac{(\mathbf{A}^k)_{ii}}{k!} = \sum_{i=1}^{n} \sum_{k=0}^{\infty} \frac{\lambda_i^k}{k!} = \sum_{i=1}^{n} e^{\lambda_i} .$$

Natural connectivity is then the normalized Estrada index and quantifies the "average communicability" in a graph.

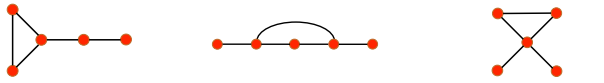
Robustness vs. Density

Graph robustness appears to be related to graph density. Here we show that although the two properties are related, there exist key distinctions between them.

Firstly, while density directly uses the number of edges e, such as $\frac{2e(G)}{|V|}$ as in average degree [5, 104, 105] or $\frac{2e(G)}{|V|(|V|-1)}$ as in edge density [6, 106], robustness follows an indirect route; it quantifies the count and length of paths and uses the graph spectrum. Thus, the objectives of robust and dense subgraph mining problems are distinct.

More notably, density concerns with the number of edges in the graph and not with the topology. On the other hand, for robustness the *placement* of edges (i.e., topology) is as much, if not more important. In fact, graphs with the same number of nodes and edges but different topologies are indistinguishable from the density point of view (Figure 6.1).

To illustrate further, we show in Figure 6.2 the robustness and density of



$\bar{\lambda} = 0.9564$ $\bar{\lambda} = 0.9804$ $\bar{\lambda} = 0.9965$

Figure 6.1: Example graphs with the same density but different robustness, i.e. topology.

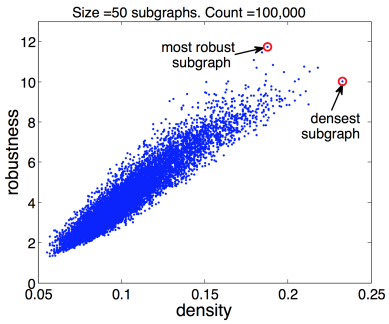


Figure 6.2: Robustness vs. density of 100,000 connected subgraphs on a real email graph.

example subgraphs, each of size 50, sampled[1] from a real-world email network (§6.5, Table 6.1). We notice that while the two properties are correlated, subgraphs with the same density do have a range of different robustness values. Moreover, among all the samples, the densest and the most robust subgraphs are distinct.

6.3.2 Problem Definition

In their inspiring work [6], Tsourakakis *et al.* recently defined a general framework for subgraph density functions, which is written as

$$f_\alpha(S) = g(e[S]) - \alpha h(|S|) \ ,$$

where $S \subseteq V$ is a set of nodes, $S \neq \emptyset$, $e[S]$ is the number of edges in the subgraph induced by S, $\alpha > 0$, and g and h are any two strictly increasing functions.

Under this framework, maximizing the *average degree* of a subgraph [5, 104, 105] corresponds to $g(x) = h(x) = \log x$ and $\alpha = 1$ such that

$$f(S) = \log \frac{e[S]}{|S|} \ .$$

In order to define our problem, we can relate the objective of our setting to this general framework. Specifically, our objective can be written as

$$f(S) = \log \frac{\sum_{i=1}^{|S|} e^{\lambda_i}}{|S|} \ ,$$

which is to maximize the *average eigenvalue* of a subgraph. As such, the objectives of the two problems are distinct, but they both relate to a more

[1] We create each subgraph using snowball sampling: we pick a random node and progressively add its neighbors with probability p, and iterate in a depth-first fashion. Connectivity is guaranteed by adding at least one neighbor of each node. We use varying $p \in (0, 1)$ that controls the tree-likeness, to obtain subgraphs with various densities.

general framework [6].

In the following, we formally define our robust local subgraph mining problem, which is to find the highest robustness subgraph of a certain size (hence the locality) in a given graph, which we call the RLS-Problem.

Problem 1 (RLS-Problem)**.** *Given a graph $G = (V, E)$ and an integer s, find a subgraph with nodes $S^* \subseteq V$ of size $|S^*| = s$ such that*

$$f(S^*) = \log \sum_{i=1}^{s} e^{\lambda_i([S^*])} - \log s \geq f(S), \quad \forall S \subseteq V, |S| = s \; .$$

S^* is referred as the most robust s-subgraph*.*

One can interpret a robust subgraph as containing a set of nodes having large communicability within the subgraph.

Problem Hardness

In this section we show that the optimal RLS-Problem is **NP**-Hard. We write the decision version of our problem as

P1. (robust s-subgraph problem RLS-Problem) is there a subgraph S in graph G with $|S| = s$ nodes and robustness $\bar{\lambda}(S) \geq \alpha$, for some $\alpha \geq 0$?

We reduce from the NP-Complete s-clique problem [125].

P2. (s-clique problem CL) is there clique of size s in G?

Proof. It is easy to see that P1 is in NP, since given a graph G we can guess the subgraph with s nodes and compute its robustness in polynomial time.

In the reduction, the conversion of the instances works as follows. An instance of CL is a graph $G = (V, E)$ and an integer s. We pass G, s, and $\alpha = \bar{\lambda}(C_s)$ to RLS-Problem, where C_s is a clique of size s. We show that a *yes* instance of CL maps to a *yes* instance of RLS-Problem, and vice versa. First assume C is a *yes* instance of CL, i.e., there exists a clique of size s in G. Clearly the same is also a *yes* instance of RLS-Problem as $\bar{\lambda}(C_s) \geq \alpha$. Next assume S is a *yes* instance of RLS-Problem, thus $\bar{\lambda}(S) \geq \bar{\lambda}(C_s)$. The proof is by contradiction. Assume S is a subgraph with s nodes that is *not* a clique. As such, it should have one or more missing edges from C_s. Let us denote by $W_k = \text{trace}(\mathbf{A}^k_{C_s})$ the number of closed walks of length k in C_s. Deleting an edge from C_s, W_k will certainly not increase, and in some cases (e.g., for $k = 2$) will strictly decrease. As such, any s-subgraph S' of C_s with missing edges will have $\bar{\lambda}(S') < \bar{\lambda}(C_s)$, which is a contradiction to our assumption that S is a *yes* instance of the RLS-Problem. Thus, S should be an s-clique and also a *yes* instance of CL. □

Properties of Robust Subgraphs

NP-hardness of the RLS-Problem suggests that it cannot be solved in polynomial time, unless P=NP. As a consequence, one needs to resort to heuristic algorithms.

Certain characteristics of hard combinatorial problems sometimes guide the development of approximation algorithms for those problems. For example, cliques display a key property used in successful algorithms for the maximum clique problem called heredity, which states that if the property exists in a graph, it also exists in all its induced subgraphs. Thanks to this property of cliques, e.g., checking maximality by inclusion is a trivial task and effective pruning strategies can be employed within a branch-and-bound framework. In this section, we study two such characteristics; subgraph monotonicity and semi-heredity for the RLS-Problem, and show that, alas, robust subgraphs do not exhibit any of these properties.

Semi-heredity: It is easy to identify α-robust graphs containing subsets of nodes that induce subgraphs with robustness less than α. As such, robust subgraphs do not display heredity. Here, we study a weaker version of heredity called semi-heredity or quasi-inheritance.

Definition 1 (Semi-heredity)**.** *Given any graph $G = (V, E)$ satisfying a property p, if there exists* some $v \in V$ *such that $G - v \equiv G[V \setminus \{v\}]$ also has property p, p is called a semi-hereditary property.*

Theorem 1. *The graph property of having at least α robustness $\bar{\lambda}_\alpha$ does not display semi-heredity. In other words, it does not hold that any α-robust graph with $s > 1$ nodes is a strict superset of* some *α-robust graph with $s - 1$ nodes.*

Proof. The proof is by counter example. In particular, robustness of cliques is not semi-hereditary. Without loss of generality, let C_k be a k-clique. Then, $\bar{\lambda}(C_k) = \ln \frac{1}{k}(e^{k-1} + (k-1)\frac{1}{e})$. *Any* subgraph of C_k with $k-1$ nodes is also a clique having strictly lower robustness, for $k > 1$, i.e.,

$$\begin{aligned}
\frac{1}{k-1}(e^{k-2} + (k-2)\frac{1}{e}) &\leq \frac{1}{k}(e^{k-1} + (k-1)\frac{1}{e}) \\
ke^{k-2} + \frac{k(k-2)}{e} &\leq (k-1)e^{(k-1)} + \frac{(k-1)^2}{e} \\
ke^{k-1} + k^2 - 2k &\leq (k-1)e^k + (k^2 - 2k + 1) \\
ke^{k-1} &\leq (k-1)e^k + 1
\end{aligned}$$

where the inequality is sharp only for $k = 1$. Thus, for $\alpha = \bar{\lambda}(C_k)$, there exists *no* v such that $C_k - v$ is α-robust. □

Subgraph monotonicity: As we defined in §6.3.1, our robustness measure can be written in terms of subgraph centrality and can be as $\bar{\lambda}(G) = \log(\frac{1}{n}S(G))$.

As $S(G)$ is the total number of weighted closed walks, $\bar{\lambda}$ is strictly monotonic with respect to edge additions and deletions. However, monotonicity is not directly obvious for node modifications due to the $\frac{1}{n}$ factor in the definition, which changes with the graph size.

Definition 2 (Subgraph Monotonicity)**.** *An objective function (in our case, robustness) R is* subgraph monotonic *if for any subgraph $g = (V', E')$ of $G = (V, E)$, $V' \subseteq V$ and $E' \subseteq E$, $R(g) \leq R(G)$.*

Theorem 2. *Robustness $\bar{\lambda}$ is not subgraph monotonic.*

Proof. Assume we start with any graph G with robustness $\bar{\lambda}(G)$. Next, we want to find a graph S with as large robustness as $\bar{\lambda}(G)$ but that contains the minimum possible number of nodes $V_{\min}$. Such a smallest subgraph is in fact a clique, with the largest eigenvalue $(V_{\min} - 1)$ and the rest of the $(V_{\min} - 1)$ eigenvalues equal to -1.[2] To obtain the exact value of $V_{\min}$, we need to solve the following

$$\bar{\lambda}(G) = \log\left[\frac{1}{V_{\min}}\left(e^{V_{\min}-1} + (V_{\min} - 1)e^{-1}\right)\right]$$

which, however, is a transcendental equation and is often solved using numerical methods. To obtain a solution quickly, we calculate a linear regression over $(\bar{\lambda}(G), V_{\min})$ samples. We show a sample simulation in Figure 6.3 for $V_{\min}$ 1 to 100 where the regression equation is

$$V_{\min} = 1.0295 * \bar{\lambda}(G) + 3.2826$$

Irrespective of how one computes $V_{\min}$, we next construct a new graph $G' = G \cup S'$ in which G is the original graph with n nodes and S' is a clique of size $V_{\min} + 1$. Let $\lambda = \bar{\lambda}(G)$ and $\lambda' = \bar{\lambda}(S')$, and as such, $\lambda < \lambda'$. Then, we can write the robustness of G' as

$$\bar{\lambda}(G') = \ln\frac{ne^{\lambda} + (V_{\min} + 1)e^{\lambda'}}{n + V_{\min} + 1} < \ln\frac{ne^{\lambda'} + (V_{\min} + 1)e^{\lambda'}}{n + V_{\min} + 1} = \lambda'$$

which shows that S', which is a subgraph of G', has strictly larger robustness than the original graph. This construction shows that $\bar{\lambda}$ is not subgraph monotonic. □

[2] Any subgraph $g(C)$ of a k-clique C has strictly lower robustness $\bar{\lambda}$. This is true when $g(C)$ also contains k nodes, due to monotonicity of $S(G)$ to edge removals (§6.3.2). Moreover, any smaller clique has strictly lower robustness, see proof for semi-heredity.

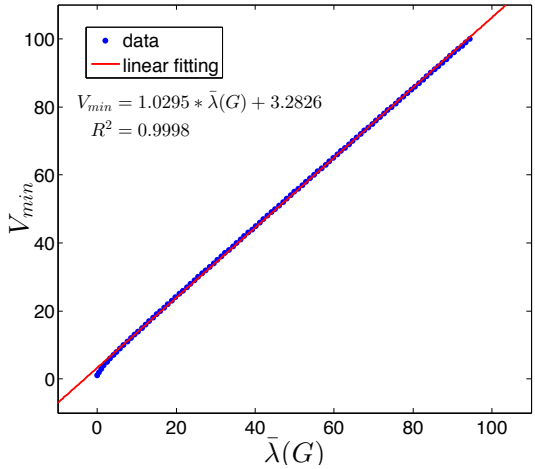


Figure 6.3: Relation between $\bar{\lambda}(G)$ and $V_{\min}$

As we mentioned for cliques before, problems that exhibit properties such as (semi-)heredity and monotonicity often enjoy algorithms that explore the search space in a smart and efficient way. For example, such algorithms employ some "smart node ordering" strategies to find iteratively improving solutions. This starts with the first node in the order and sequentially adds the next node such that the resulting subgraphs all satisfy some desired criteria, like a minimum density, which enables finding large solutions quickly.

Showing that our robustness objective displays neither characteristic implies that our RLS-Problem is likely harder to solve than the maximum clique and densest subgraph problems as, unlike robust subgraphs, (quasi-)cliques are shown to exhibit e.g., the (semi-)heredity property [106].

Problem Variants

Before we conclude this section, we introduce three variants of our RLS-Problem, that may also be practical in certain real-world scenarios.

Given that robustness $\bar{\lambda}$ is not subgraph-monotonic, it is natural to consider the problem of finding the subgraph with the maximum overall robustness in the graph (without any restriction on its size), which is not necessarily the full graph. We call this first variant the robust global subgraph problem or the RGS-Problem.

Problem 2 (RGS-Problem)**.** *Given a graph $G = (V, E)$, find a subgraph $S^* \subseteq V$ such that*

$$f(S^*) = \max_{S \subseteq V} f(S) \ .$$

S^* is referred as the most robust subgraph*.*

Another variant involves finding the top k most robust s-subgraphs in a graph, which we call the kRLS-Problem.

Problem 3 (kRLS-Problem)**.** *Given a graph $G = (V, E)$, and two integers s and k, find k subgraphs $\mathcal{S} = S_1^*, \ldots, S_k^*$, each of size $|S_i^*| = s$, $1 \leq i \leq k$ such that*

$$f(S_1^*) \geq f(S_2^*) \geq \ldots \geq f(S_k^*) \geq f(S), \;\; \forall S \subseteq V, |S| = s \; .$$

The set $\mathcal{S}$ is referred as the top-k most robust s-subgraphs.

In the final variant, the goal is to find the most robust s-subgraph that contains a set of *user-given* seed nodes.

Problem 4 (Seeded-RLS-Problem)**.** *Given a graph $G = (V, E)$, an integer s, and a set of seed nodes U, $|U| \leq s$, find a subgraph $U \subseteq S^* \subseteq V$ of size $|S^*| = s$ such that*

$$f(S^*) = \max_{U \subseteq S \subseteq V, |S| = s} f(S) \; .$$

S^* is referred as the most robust seeded s-subgraph.

It is easy to see that when $k = 1$ and $U = \emptyset$, the kRLS-Problem and the Seeded-RLS-Problem respectively reduce to the RLS-Problem, and thus can easily be shown to be also **NP**-hard. A formal proof of hardness for the RGS-Problem, however, is nontrivial and remains an interesting open problem for future research.

In the next section, where we propose solutions for our original RLS-Problem, we also show how our algorithms can be adapted to solve these three variants of the problem.

6.4 Robust Local Subgraph Mining

Given the hardness of our problem, we propose two heuristic solutions. The first is a top-down greedy approach, called GreedyRLS, in which we iteratively remove nodes to obtain a subgraph of desired size, while the second, called GRASP-RLS, is a bottom-up approach in which we iteratively add nodes to build up our subgraphs. Both solutions carefully order the nodes by their contributions to the robustness.

6.4.1 Greedy Top-down Search Approach

This approach iteratively and greedily removes the nodes one by one from the given graph, until a subgraph with the desired size s is reached. At each

iteration, the node whose removal results in the maximum robustness of the residual graph among all the nodes is selected for removal.[3]

The removal of a node involves removing the node itself and the edges attached to it from the graph, where the residual graph becomes $G[V \backslash \{i\}]$. Let i denote a node to be removed. Let us then write the updated robustness $\bar{\lambda}_\Delta$ as

$$\bar{\lambda}_\Delta = \log\left(\frac{1}{n-1}\sum_{j=1}^{n-1} e^{\lambda_j + \Delta\lambda_j}\right). \tag{6.2}$$

As such, we are interested in identifying the node that maximizes $\bar{\lambda}_\Delta$, or equivalently

$$\begin{aligned} \textbf{max}. \quad & e^{\lambda_1+\Delta\lambda_1} + e^{\lambda_2+\Delta\lambda_2} + \ldots + e^{\lambda_{n-1}+\Delta\lambda_{n-1}} \\ & e^{\lambda_1}(e^{\Delta\lambda_1} + e^{(\lambda_2-\lambda_1)}e^{\Delta\lambda_2} + \ldots + e^{(\lambda_{n-1}-\lambda_1)}e^{\Delta\lambda_{n-1}}) \\ & c_1(e^{\Delta\lambda_1} + c_2 e^{\Delta\lambda_2} + \ldots + c_{n-1}e^{\Delta\lambda_{n-1}}) \end{aligned} \tag{6.3}$$

where c_j's denote constant terms and $c_j \leq 1$, $\forall j \geq 2$.

Updating the Eigen-pairs

When a node is removed from the graph, its spectrum (i.e., the eigen-pairs) also changes. Recomputing the eigen-values for Equ. (6.3) every time a node is removed is computationally challenging. Therefore, we employ fast update schemes based on the first order matrix perturbation theory [133].

Let $\Delta\mathbf{A}$ and $(\Delta\lambda_j, \Delta\mathbf{u_j})$ denote the change in $\mathbf{A}$ and $(\lambda_j, \mathbf{u_j})$ $\forall j$, respectively, where $\Delta\mathbf{A}$ is symmetric. Suppose after the adjustment $\mathbf{A}$ becomes

$$\tilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$$

where each eigen-pair $(\tilde{\lambda}_j, \tilde{u}_j)$ is written as

$$\tilde{\lambda}_j = \lambda_j + \Delta\lambda_j \qquad \text{and} \qquad \tilde{\mathbf{u}}_\mathbf{j} = \mathbf{u_j} + \Delta\mathbf{u_j}$$

Lemma 1. *Given a perturbation $\Delta\mathbf{A}$ to a matrix $\mathbf{A}$, its eigenvalues can be updated by*

$$\Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j}. \tag{6.4}$$

Proof. We can write

$$(\mathbf{A} + \Delta\mathbf{A})(\mathbf{u_j} + \Delta\mathbf{u_j}) = (\lambda_j + \Delta\lambda_j)(\mathbf{u_j} + \Delta\mathbf{u_j})$$

[3] Robustness of the residual graph can be lower or higher; $S(G)$ decreases due to monotonicity, but denominator also shrinks to $(n-1)$.

Expanding the above, we get

$$\begin{aligned}&\mathbf{A}\mathbf{u_j} + \Delta\mathbf{A}\mathbf{u_j} + \mathbf{A}\Delta\mathbf{u_j} + \Delta\mathbf{A}\Delta\mathbf{u_j}\\ &= \lambda_j\mathbf{u_j} + \Delta\lambda_j\mathbf{u_j} + \lambda_j\Delta\mathbf{u_j} + \Delta\lambda_j\Delta\mathbf{u_j}\end{aligned} \tag{6.5}$$

By concentrating on first-order approximation, we assume that all high-order perturbation terms are negligible, including $\Delta\mathbf{A}\Delta\mathbf{u_j}$ and $\Delta\lambda_j\Delta\mathbf{u_j}$. Further, by using the fact that $\mathbf{A}\mathbf{u_j} = \lambda_j\mathbf{u_j}$ (i.e., canceling these terms) we obtain

$$\Delta\mathbf{A}\mathbf{u_j} + \mathbf{A}\Delta\mathbf{u_j} = \Delta\lambda_j\mathbf{u_j} + \lambda_j\Delta\mathbf{u_j} \tag{6.6}$$

Next we multiply both sides by $\mathbf{u_j}'$ and by symmetry of $\mathbf{A}$ and orthonormal property of its eigenvectors we get Equ. (6.4), which concludes the proof. □

Since updating eigenvalues involves the eigenvectors, which also change with node removals, we use the following to update the eigenvectors as well.

Lemma 2. *Given a perturbation $\Delta\mathbf{A}$ to a matrix $\mathbf{A}$, its eigenvectors can be updated by*

$$\Delta\mathbf{u_j} = \sum_{i=1, i\neq j}^{n} \left(\frac{\mathbf{u_i}'\Delta\mathbf{A}\mathbf{u_j}}{\lambda_j - \lambda_i}\mathbf{u_i}\right). \tag{6.7}$$

Proof. Using the orthogonality property of the eigenvectors, we can write the change $\Delta\mathbf{u_j}$ of eigenvector $\mathbf{u_j}$ as a linear combination of the original eigenvectors:

$$\Delta\mathbf{u_j} = \sum_{i=1}^{n} \alpha_{ij}\mathbf{u_i} \tag{6.8}$$

where α_{ij}'s are small constants that we aim to determine.

Using Equ. (6.8) in Equ. (6.6) we obtain

$$\Delta\mathbf{A}\mathbf{u_j} + \mathbf{A}\sum_{i=1}^{n} \alpha_{ij}\mathbf{u_i} = \Delta\lambda_j\mathbf{u_j} + \lambda_j\sum_{i=1}^{n} \alpha_{ij}\mathbf{u_i}$$

which is equivalent to

$$\Delta\mathbf{A}\mathbf{u_j} + \sum_{i=1}^{n} \lambda_i\alpha_{ij}\mathbf{u_i} = \Delta\lambda_j\mathbf{u_j} + \lambda_j\sum_{i=1}^{n} \alpha_{ij}\mathbf{u_i}$$

Multiplying both sides of the above by $\mathbf{u_k}'$, $k \neq j$, we get

$$\mathbf{u_k}'\Delta\mathbf{A}\mathbf{u_j} + \lambda_k\alpha_{kj} = \lambda_j\alpha_{kj}$$

Therefore,

$$\alpha_{kj} = \frac{\mathbf{u_k}'\Delta\mathbf{A}\mathbf{u_j}}{\lambda_j - \lambda_k} \tag{6.9}$$

for $k \neq j$. To obtain α_{jj} we use the following derivation.

$$\begin{aligned}\tilde{\mathbf{u}}_\mathbf{j}'\tilde{\mathbf{u}}_\mathbf{j} = 1 &\Rightarrow (\mathbf{u_j} + \Delta\mathbf{u_j})'(\mathbf{u_j} + \Delta\mathbf{u_j}) = 1 \\ &\Rightarrow 1 + 2\mathbf{u_j}'\Delta\mathbf{u_j} + \|\Delta\mathbf{u_j}\|^2 = 1\end{aligned}$$

After we discard the high-order term, and substitute $\Delta\mathbf{u_j}$ with Equ. (6.8) we get $1 + 2\alpha_{jj} = 1 \Rightarrow \alpha_{jj} = 0$.

We note that for a slightly better approximation, one can choose not to ignore the high-order term which is equal to $\|\Delta\mathbf{u_j}\|^2 = \sum_{i=1}^{n} \alpha_{ij}^2$. Thus, one can compute α_{jj} as

$$1 + 2\alpha_{jj} + \sum_{i=1}^{n} \alpha_{ij}^2 = 1 \Rightarrow 1 + 2\alpha_{jj} + \alpha_{jj}^2 + \sum_{i=1, i\neq j}^{n} \alpha_{ij}^2 = 1$$

$$\Rightarrow (1 + \alpha_{jj})^2 + \sum_{i=1, i\neq j}^{n} \alpha_{ij}^2 = 1 \Rightarrow \alpha_{jj} = \sqrt{1 - \sum_{i=1, i\neq j}^{n} \alpha_{ij}^2} - 1$$

All in all, using the α_{ij}'s as given by Equ. (6.9) and $\alpha_{jj} = 0$, we can see that $\Delta\mathbf{u_j}$ in Equ. (6.8) is equal to Equ. (6.7). □

Node Selection for Removal

By using *Lemma 1*, we can write the affect of perturbing $\mathbf{A}$ with the removal of a node i on the eigenvalues as

$$\Delta\lambda_j = \mathbf{u_j}'\Delta\mathbf{A}\mathbf{u_j} = -2\mathbf{u_{ij}} \sum_{v\in\mathcal{N}(i)} \mathbf{u_{vj}} \tag{6.10}$$

where $\mathbf{A}(i, v) = \mathbf{A}(v, i) = -1$, for $v \in \mathcal{N}(i)$, and 0 elsewhere. That is, the change in j^{th} eigenvalue after a node i's removal is twice the sum of eigenscores of i's neighbors times eigenscore of i, where eigenscores denote the corresponding entries in the associated j^{th} eigenvector. Thus, at each step we choose the node that maximizes the following.

$$\max_{i\in V} \; c_1 \left(e^{-2\mathbf{u_{i1}} \sum_{v\in\mathcal{N}(i)} \mathbf{u_{v1}}} + \ldots + c_{n-1} e^{-2\mathbf{u_{in-1}} \sum_{v\in\mathcal{N}(i)} \mathbf{u_{vn-1}}} \right) \tag{6.11}$$

We remark that it is infeasible to compute all the n eigenvalues of a graph with n nodes, for very large n. Thanks to the skewed spectrum of real-world graphs [134], we can rely on the observation that only the top few eigenvalues have large magnitudes. This implies that the c_j terms in Equ. (6.11) become much smaller for increasing j and can be ignored. Therefore, we use the top t eigenvalues to approximate the robustness of a graph. In the past, the skewed property of the spectrum has also been exploited to approximate triangle counts in large graphs [135]. The outline of our algorithm, called GreedyRLS, is given in Algorithm 10.

Complexity Analysis

Algorithm 10 has three main components: (a) computing top t eigenpairs (L1): $O(nt + mt + nt^2)$, (b) computing Equ. (6.11) scores for all nodes (L4): $O(mt)$ ($\sum_i d_i t = t \sum_i d_i = 2mt$), and (c) updating eigenvalues & eigenvectors when a node i is removed (L10-11): $O(d_i t)$ & $O(nt^2)$, respectively.

Performing (b) for all nodes at every iteration takes $O(tmn)$. Moreover, performing (c) iteratively for all nodes requires $\sum_{i=1}^{n} d_i t = t \sum_i d_i = 2mt$, i.e., $O(tm)$ for eigenvalues and $\sum_{i=k}^{n} it^2, O(t^2 n^2)$ for eigenvectors. Therefore, the overall complexity becomes $O(\max(tmn, t^2 n^2))$.

As we no longer would have small perturbations to the adjacency matrix over many iterations, updating the eigen-pairs at all steps would yield bad approximations. As such, we recompute the eigen-pairs at every $\frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots$ steps. Performing recomputes less frequently in early iterations is reasonable, as early nodes are likely the peripheral ones that do not affect the eigen-pairs much, for which updates would suffice. When perturbations accumulate over iterations and especially when we get closer to the solution, it becomes beneficial to recompute the eigen-pairs more frequently.

In fact, in a greedier version one can drop the eigen-pair updates (L9-11), so as to perform $O(\log n)$ recomputes, and the complexity becomes $O(\max(tm \log n, t^2 n \log n))$.

Algorithm Variants

To adapt our GreedyRLS algorithm for the kRLS-Problem, we can find one subgraph at a time, remove all its nodes from the graph, and continue until we find k subgraphs or end up with an empty graph. This way we generate pairwise disjoint robust subgraphs.

For the Seeded-RLS-Problem, we can add a condition to never remove nodes $u \in U$ that belong to the seed set.

Algorithm 10: GreedyRLS

Input : Graph $G = (V, E)$, its adj. matrix $\mathbf{A}$, integer s.
Output: Subset of nodes $S^* \subseteq V$, $|S^*| = s$.

1 Compute top t eigen-pairs $(\lambda_j, \mathbf{u_j})$ of $\mathbf{A}$, $1 \leq j \leq t$;
2 $S_n \leftarrow V$, $\bar{\lambda}(S_n) = \bar{\lambda}(G)$;
3 **for** $z = n$ *down to* $s + 1$ **do**
4 Select node $\bar{i}$ out of $\forall i \in S_z$ that maximizes Equ. (6.11) for top t eigen-pairs of $G[S_z]$, i.e.

$$f = \max_{i \in S_z} \; c_1 \left(e^{-2\mathbf{u_{i1}} \sum_{v \in \mathcal{N}(i)} \mathbf{u_{v1}}} + \ldots + c_t e^{-2\mathbf{u_{it}} \sum_{v \in \mathcal{N}(i)} \mathbf{u_{vt}}} \right)$$

where $c_1 = e^{\lambda_1}$ and $c_j = e^{(\lambda_j - \lambda_1)}$ for $2 \leq j \leq t$;
5 $S_{z-1} := S_z \backslash \{\bar{i}\}$, $\bar{\lambda}(S_{z-1}) = \log \frac{f}{z-1}$;
6 Update $\mathbf{A}$; $\mathbf{A}(:, \bar{i}) = 0$ and $\mathbf{A}(\bar{i}, :) = 0$;
7 **if** $z = \frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots$ **then**
8 Compute top t eigen-pairs $(\lambda_j, \mathbf{u_j})$ of $\mathbf{A}$, $1 \leq j \leq t$
9 **end**
10 **else**
11 Update top t eigenvalues of $\mathbf{A}$ by Equ. (6.4);
12 Update top t eigenvectors of $\mathbf{A}$ by Equ. (6.7);
13 **end**
14 **end**
15 Return $S^* \leftarrow S_{z=s}$;

GreedyRLS algorithm is particularly suitable for the RGS-Problem, where we can iterate for $z = n, \ldots, V_{\min}$[4], record $\bar{\lambda}(S_z)$ for each subgraph at each step (L5), and return the subgraph with the maximum robustness among S_z's.

6.4.2 Greedy Randomized Adaptive Search Procedure (GRASP) Approach

The top-down approach makes a greedy decision at every step to reach a solution. If the desired subgraphs are small (i.e., for small s), however, it implies many greedy decisions, especially on large graphs, where the number

[4] $V_{\min}$ denotes the minimum number of nodes a clique C with robustness at least as large as the full graph's would contain. Any subgraph of C has lower robustness (see §6.3.2) and hence would not qualify as the most robust subgraph.

of greedy steps $(n - s)$ would be excessive. Therefore, here we propose a bottom-up approach that performs local operations to build up solutions from scratch.

Our local approach is based on a well-established meta-heuristic called GRASP [107] for solving combinatorial optimization problems. A GRASP, or greedy randomized adaptive search procedure, is a multi-start or iterative process, in which each iteration consists of two phases: (i) a construction phase, in which an initial feasible solution is produced, and (ii) a local search phase, in which a better solution with higher objective value in the neighborhood of the constructed solution is sought. The best overall solution is returned as the final result.

The pseudo-code in Algorithm 11 shows the general GRASP for maximization, where $T_{\max}$ iterations are done. For maximizing our objective, we use $f : S \to \mathbb{R} \equiv \bar{\lambda}$, i.e., the robustness function as given in Equ. (6.1). We next describe the details of our two GRASP phases.

Algorithm 11: GRASP-RLS

Input : Graph $G = (V, E)$, $T_{\max}$, $f(\cdot)$, $g(\cdot)$, integer s.
Output: Subset of nodes $S^* \subseteq V$, $|S^*| = s$

$f^* = -\infty$, $S^* = \emptyset$;
for $z = 1, 2, \ldots, T_{\max}$ **do**
  $S \leftarrow$ GRASP-RLS-Construction(G, $g(\cdot)$, s);
  $S' \leftarrow$ GRASP-RLS-LocalSearch(G, S, $f(\cdot)$, s);
  if $f(S') > f^*$ **then**
    $S^* \leftarrow S$, $f^* = f(S)$
  end
end
Return S^*

Construction

In the construction phase, a feasible solution, which we call a seed subgraph, is iteratively constructed, one node at a time. At each iteration, the choice of the next node to be added is determined by ordering all candidate nodes C (i.e., those that can be added to the solution) in a restricted candidate list, called RCL, with respect to a greedy function $g : C \to \mathbb{R}$, and randomly choosing one of the candidates in the list. The size of RCL is determined by a real parameter $\beta \in [0, 1]$, which controls the amount of greediness and randomness. $\beta = 1$ corresponds to a purely greedy construction, while $\beta = 0$ produces a purely random one. Algorithm 12 describes our construction phase.

Algorithm 12: GRASP-RLS-Construction

Input : Graph $G = (V, E)$, $g(\cdot)$, integer s
Output: Subset of nodes $S \subseteq V$

$S \leftarrow \emptyset$, $C \leftarrow V$;
while $|S| < s$ **do**
  Evaluate $g(v)$ for all $v \in C$;
  $\bar{c} \leftarrow \max_{v \in C} g(v)$, $\underline{c} \leftarrow \min_{v \in C} g(v)$;
  Select $\beta \in [0, 1]$ using a strategy;
  $RCL \leftarrow \{v \in C | g(v) \geq \underline{c} + \beta(\bar{c} - \underline{c})\}$;
  Select a vertex r from RCL at random;
  $S := S \cup \{r\}$, $C \leftarrow \mathcal{N}(S) \backslash S$;
end
Return S;

Selecting $\mathbf{g}(\cdot)$: We use three different scoring functions for ordering the candidate nodes. First we aim to include locally dense nodes in the seed subgraphs, therefore we use $g(v) = \frac{t(v)}{d(v)}$, where $t(v)$ denotes the number of local triangles of v, and $d(v)$ is its degree. We approximate the local triangle counts using the technique described in [135].

Another approach is sorting the candidate nodes by their degree in the induced neighborhood subgraph of S, i.e., $g(v) = d_{G[C \cup S]}(v)$. This strategy favors high degree nodes in the first iteration of the construction.

Finally we use $g(v) = \Delta\bar{\lambda}_v$, i.e., the difference in robustness when a candidate node is added to the current subgraph. The first iteration then chooses a node at random.

Selecting $\boldsymbol{\beta}$: Setting $\beta = 1$ is purely greedy and would produce the same seed subgraph in every GRASP iteration. To incorporate randomness, while staying close to the greedy best-first selection, one can choose a fixed $1 > \beta > 0.5$, which produces high average solution values in the presence of large variance in constructed solutions [107]. We also try choosing β uniformly. Other more complex selection strategies include using an increasingly non-uniform discrete distribution (where large values are favored), and reactive GRASP [136] that guides the construction by the solutions found along the previous iterations (where β values that produced better average solutions are favored).

Local Search

A solution generated by GRASP-RLS-Construction is a preliminary one and may not necessarily have the best robustness. Thus, it is almost always beneficial to apply a local improvement procedure to each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution with a better one in the neighborhood of the current solution. It terminates when no better solution can be found. We give the pseudo-code of our local search phase in Algorithm 13.

As the RLS-Problem asks for a subgraph of size s, the local search takes as input an s-subgraph generated by construction and searches for a better solution around it by "swapping" nodes in and out. Ultimately it finds a locally optimal subgraph of size upper bounded by $s+1$. As an answer, it returns the best s-subgraph with the highest robustness found over the iterations.[5] As such, GRASP-RLS-LocalSearch is an adaptation of a general local search procedure to yield subgraphs of desired size, as in our setting.

Convergence: The local search algorithm is guaranteed to terminate, as the objective function (i.e., subgraph robustness) improves with every iteration and the robustness values are upper-bounded from above, by the robustness of the n-clique, i.e., $\bar{\lambda}(S) < \bar{\lambda}(C_n)$, for all $|S| \leq n$.

Complexity analysis

The size of subgraphs $|S|$ obtained during local search is $O(s)$. Computing their top t eigen-pairs takes $O(s^2t + st^2)$, where we use $e([S]) = O(s^2)$ as robust subgraphs are often dense. To find the best improving node (L12), all nodes in the neighborhood $\mathcal{N}(S)\backslash S$ are evaluated, with worst-case size $O(n)$. As such, each expansion costs $O(ns^2t+nst^2)$. With deletions incorporated (L3-4), the number of expansions can be arbitrarily large [137], however assuming $O(s)$ expansions are done, overall complexity becomes $O(ns^3t + ns^2t^2)$. If all $t = |S|$ eigen-pairs are computed, the complexity is quadruple in s and linear in n, which is feasible for small s. Otherwise, we exploit eigen-pair updates as in GreedyRLS to reduce computation.

Algorithm Variants

Adapting GRASP-RLS for the kRLS-Problem can be easily done by returning the best k (out of $T_{\max}$) distinct subgraphs computed during the GRASP iterations in Algorithm 11. These subgraphs are likely to overlap, although one can incorporate constraints as to the extent of overlap.

[5] Note that the locally optimal solution size may be different from s.

Algorithm 13: GRASP-RLS-LocalSearch

Input : Graph $G = (V, E)$, S, integer s.
Output: Subset of nodes $S' \subseteq V$, $|S'| = s$.

$more \leftarrow$ TRUE, $S' \leftarrow S$;
while *more* **do**
  if *there exists* $v \in S$ *such that* $\bar{\lambda}(S \backslash \{v\}) \geq \bar{\lambda}(S)$ **then**
    $S := S \backslash \{v^*\}$ where $v^* := \max_{v \in \mathcal{N}(S) \backslash S} \bar{\lambda}(S \backslash \{v\})$;
    if $|S| = s$ **then** $S' \leftarrow S$;
  end
  else *more* $\leftarrow$ FALSE ;
  add $\leftarrow$ TRUE;
  while *add and* $|S| \leq s$ **do**
    if *there is* $v \in \mathcal{N}(S) \backslash S$ *s.t.* $\bar{\lambda}(S \cup \{v\}) > \bar{\lambda}(S)$ **then**
      $S := S \cup \{v^*\}$, $v^* := \max_{v \in \mathcal{N}(S) \backslash S} \bar{\lambda}(S \cup \{v\})$;
      more $\leftarrow$ TRUE;
      if $|S| = s$ **then** $S' \leftarrow S$;
    end
    else
      add $\leftarrow$ FALSE;
    end
  end
end
Return S'

For the Seeded-RLS-Problem, we can initialize set S with the seed nodes U in construction, while, during the local search phase, we never allow a node $u \in U$ to leave S.

Finally, for the RGS-Problem, we can waive the size constraint in the expansion step of local search.

6.5 Evaluations

We evaluate our proposed methods extensively on many real-world graphs, as shown in Table 6.1. We select our graphs from various domains, including biological, email, Internet AS backbone, P2P, collaboration, and Web.

Our work is in the general lines of subgraph mining, with a new objective based on robustness. The closest to our setting is the densest subgraph mining. Therefore, we compare our results to dense subgraphs found by Charikar's

algorithm [5] (which we refer as Charikar), as well as by Tsourakakis *et al.*'s two algorithms [6] (which we refer as Greedy and LS for local search). We remark that the objectives used in those works are distinct; average degree and edge-surplus, respectively, and also different from ours. As such, we compare the robust and dense subgraphs based on three main criteria: (a) robustness $\bar{\lambda}$ as in Equ (6.1), (b) triangle density $t[S]/\binom{|S|}{3}$, and (c) edge density $e[S]/\binom{|S|}{2}$.

Table 6.2 shows results on several of our datasets. Note that the three algorithms we compare to aim to find the densest subgraph in a given graph, without a size restriction. Thus, each one obtains a subgraph of a different size. To make the robust subgraphs (RS) comparable to densest subgraphs (DS) found by each algorithm, we find subgraphs with the same size as Charikar, Greedy, and LS, respectively noted as s_{Ch}, s_{Gr}, and s_{Ls}. We report our results based on GreedyRLS and GRASP-RLS.[6]

We notice that densest subgraphs found by Greedy and LS are often substantially smaller than those found by Charikar, and also have higher edge density, which was also the conclusion of [6]. On the other hand, robust subgraphs have higher robustness than densest subgraphs, *even* at lower densities. This shows that high density does not always imply high robustness, and vice versa, illustrating the differences in the two problem settings. It signifies the emphasis of robust subgraph mining on the subgraph topology, rather than the total edge count. We also note that GRASP-RLS consistently outperforms GreedyRLS, suggesting bottom-up is a superior strategy to top-down search.

Figure 6.4 shows the relative difference in robustness of GRASP-RLS subgraphs over those by Charikar, Greedy, and LS on all of our graphs. We achieve a wide range of improvements depending on the graph, while the difference is always positive.

We remark that the above comparisons are made for subgraphs at sizes where best results are obtained for each of the three densest subgraph algorithms. Our algorithms, on the other hand, accept a subgraph size input s. Thus, we compare the algorithms at varying output sizes next. Charikar and Greedy are both top-down algorithms, in which the lowest degree node is removed at each step and the best subgraph (best average degree or edge surplus, respectively) is output among all graphs created along the way. We modify these so that we pull out the subgraphs when size s is reached during the course of the algorithms.[7] Figure 6.5 shows that our GRASP-RLS produces sub-

[6] GRASP-RLS has various versions depending on choice of $g(\cdot)$ and β (§6.4.2). Our analysis suggests best results are obtained for $g(v) = \Delta\bar{\lambda}_v$ and a uniform $\beta \in [0.8, 1)$, which we report in this section.

[7] Local search by [6] finds locally optimal subgraphs, which are not guaranteed to grow to a given size s. Thus, we omit comparison to LS subgraphs at varying sizes. Figure 6.4 shows improvements over LS subgraphs are already substantially large.

Table 6.1: Real-world graphs. δ: density, $\bar{\lambda}$: robustness

Dataset	n	m	δ	$\bar{\lambda}$
Jazz	198	2742	0.1406	34.74
Celegans N.	297	2148	0.0489	21.32
Email	1133	5451	0.0085	13.74
Oregon-A	7352	15665	0.0005	42.29
Oregon-B	10860	23409	0.0004	47.54
Oregon-C	13947	30584	0.0003	52.10
P2P-GnutellaA	6301	20777	0.0010	19.62
P2P-GnutellaB	8114	26013	0.0008	19.45
P2P-GnutellaC	8717	31525	0.0008	13.35
P2P-GnutellaD	8846	31839	0.0008	14.46
P2P-GnutellaE	10876	39994	0.0007	7.83
DBLP	317080	1049866	2.09×10^{-5}	103.18
Web-Google	875713	4322051	1.13×10^{-5}	99.36

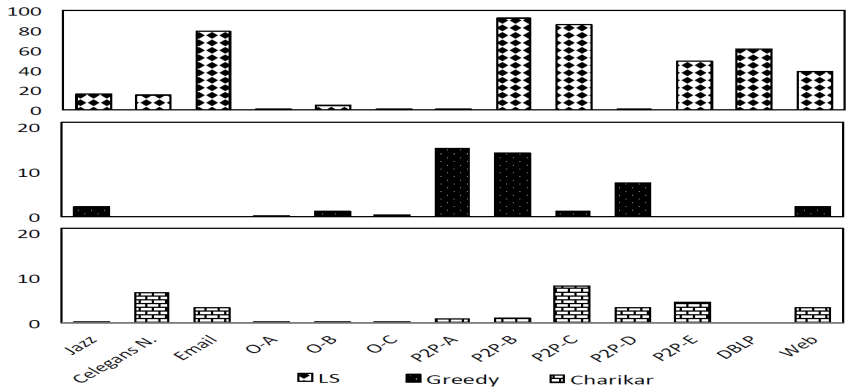


Figure 6.4: Robustness gap (%) of GRASP-RLS over (top to bottom) LS, Greedy, and Charikar on all graphs.

graphs with higher robustness than the rest of the methods at varying sizes on two example graphs. This also shows that the densest subgraph mining approaches are not suitable for our new problem.

Experiments thus far illustrate that we find subgraphs with higher robustness than densest subgraphs. These are relative results. To show that the subgraphs we find are in fact robust, we next quantify the magnitude of the robustness scores we achieve through significance tests.

Given a subgraph that GRASP-RLS finds, we bootstrap $B = 1000$ new subgraphs by rewiring its edges at random. We compute a p-value for each

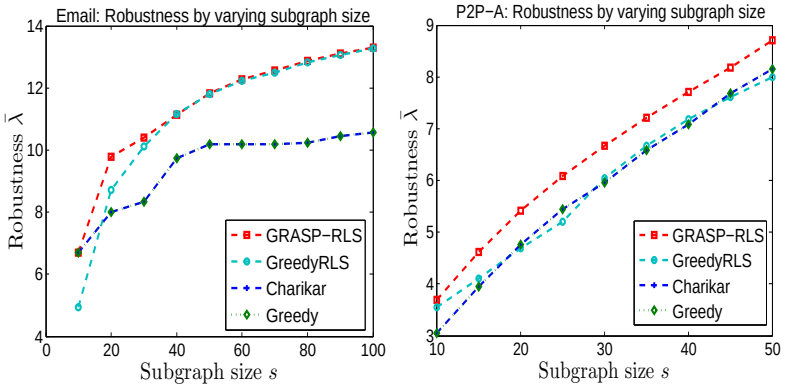


Figure 6.5: Subgraph robustness at varying sizes s.

subgraph from the number of random subgraphs created by rewiring with larger robustness than our method finds divided by B. The p-value essentially captures the probability that we would be able to obtain a subgraph with robustness greater than what we find by chance if we were to create a topology with the same number of nodes and edges at random (note that all such subgraphs would have the same edge density). Thus a low p-value implies that, among the same density topologies, the one we find is in fact robust with high probability.

Figure 6.6 shows that the subgraphs we find on almost all real graphs are significantly robust at 0.05. For cases with large p-values, it is possible to obtain higher robustness subgraphs with rewiring. For example, *P2P-E* is a graph where all the robust subgraphs (also the dense subgraphs) found contain very few or no triangles (see Table 6.2). Therefore, rewiring edges that shortcut longer cycles they contain help improve their robustness. We remark that large p-values indicate that the found subgraphs are not significantly robust, but does not imply our algorithms are unable to find robust subgraphs. That is because the rewired more robust subgraphs do not necessarily exist in the original graph G, and it is likely that G does not contain any subgraph with robustness that is statistically significant.

Next we analyze the performance of our GRASP-RLS. Recall that GRASP-RLS-Construction quickly builds a subgraph which GRASP-RLS-LocalSearch uses to improve over to obtain a better result. In Figure 6.7 we show the robustness of subgraphs obtained at construction and after local search on two example graphs for $s = 50$ and $T_{\max} = 300$. We notice that most of the GRASP-RLS iterations find a high robustness subgraph right at construction. In most other cases, local search is able to improve over construction results significantly. In fact, the most robust outcome on *Oregon-A* (Figure 6.7 left) is obtained when construction builds a subgraph with robustness around 6, which local search improves over 20.

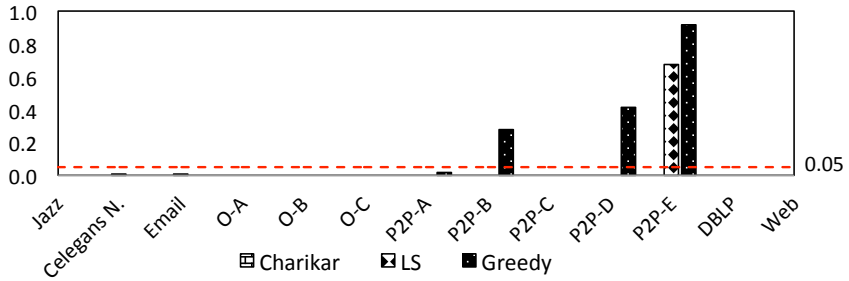


Figure 6.6: p-values of significance tests indicate that w.h.p. subgraphs we find are in fact significantly robust.

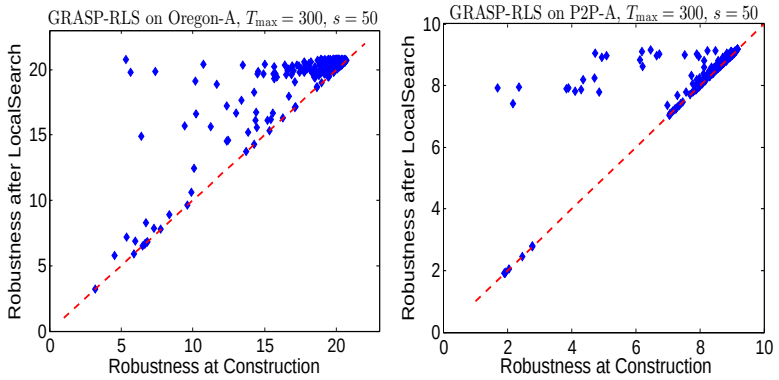


Figure 6.7: $\bar{\lambda}$ achieved at GRASP-RLS-Construction versus after GRASP-RLS-LocalSearch.

We next study the performance of GRASP-RLS w.r.t. scalability. Figure 6.8 shows that its running time grows linearly with respect to graph size on the Oregon graphs.[8]

Finally, we perform several case studies on the DBLP co-authorship network to analyze our subgraphs qualitatively. Here, we use the seeded variant of our algorithm. Christos Faloutsos is a prolific researcher with various interests. In Figure (Table) 6.3 (a), we invoke his interest in databases when we use him and Rakesh Agrawal as seeds, given that Agrawal is an expert in this field. On the other hand in (b), we invoke his interest in data mining when we use Jure Leskovec as the second seed, who is a rising star in the field. Likewise in (c) and (d) we find robust subgraphs around other selected prominent researchers in data mining and databases. In (d,e) we show how our subgraphs change with varying size. Specifically, we find a clique that the seeds Widom and Ullman belong to for s=10. The subgraph of s=15, while no longer a clique, remains stable with other researchers like Rajeev Motwani

[8]We have a total of 9 *Oregon* graphs with various sizes. We report results only on the largest 3 due to space limit (see Table 6.1).

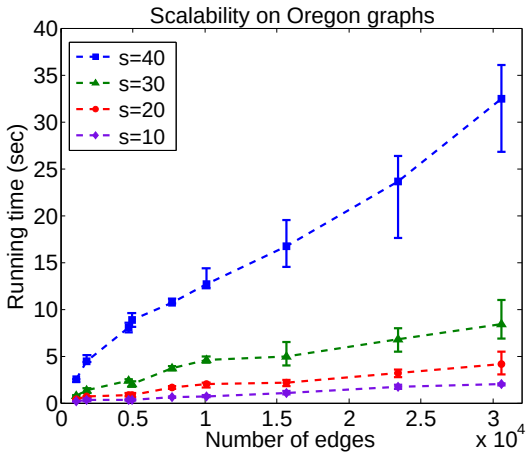


Figure 6.8: Scalability of GRASP-RLS by graph size m and subgraph size s (mean running time avg'ed over 10 independent runs, bars depict 25%-75%).

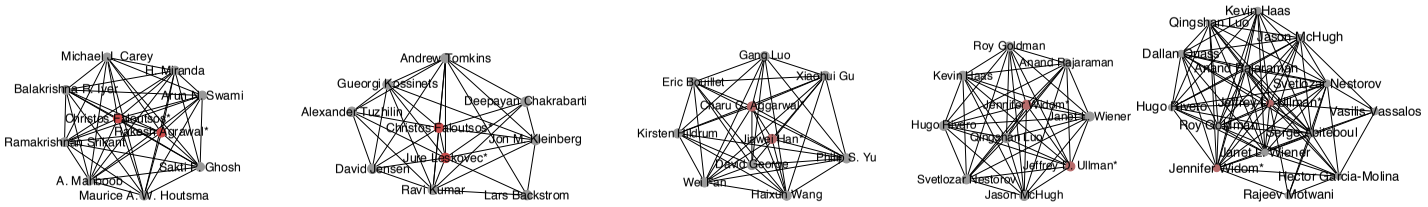
and Hector Garcia-Molina added to it.

6.6 Chapter Summary

In this chapter, we introduced the RLS-Problem of finding most robust local subgraphs of a given size in large graphs, as well as its three practical variants. While our work bears similarity to densest subgraph mining, it differs from it in its objective; robustness emphasizes subgraph topology more than edge density. We showed that our problem is **NP**-hard and that it does not exhibit semi-heredity or subgraph-monotonicity properties. We designed two heuristic algorithms based on top-down and bottom-up search strategies, and showed how we can adapt them to address the problem variants. We found that our bottom-up strategy provides consistently superior results, scales linearly with graph size, and finds subgraphs with significant robustness. Experiments on many real-world graphs further showed that our subgraphs achieve higher robustness than densest subgraphs even at lower densities, which signifies the novelty of our problem setting.

Table 6.2: Comparison of robust and densest subgraphs. Ch: Charikar [5], Gr: Greedy [6], Ls: Local search [6].

Data	Method	robustness $\lambda[S]$			triangle density $\Delta[S]$			edge density $\delta[S]$		
	(s_{Ch}, s_{Gr}, s_{Ls})	Ch	Gr	Ls	Ch	Gr	Ls	Ch	Gr	Ls
Email	DS (271, 12, 13)	13.58	8.51	4.96	**0.0009**	**1.0000**	0.2237	**0.0600**	**1.0000**	0.5897
	RS-Greedy	13.94	5.96	6.27	0.0001	0.0696	0.0606	0.0523	0.7576	0.7179
	RS-GRASP	**14.04**	**8.52**	**8.91**	0.0007	**1.0000**	**0.8671**	0.0508	**1.0000**	**0.9487**
Oreg-C	DS (87, 61, 52)	34.44	30.01	27.69	0.0868	0.1768	0.2327	**0.3892**	**0.5311**	0.5927
	RS-Greedy	34.31	24.70	21.75	0.0857	0.1022	0.1193	0.3855	0.4131	0.4367
	RS-GRASP	**34.47**	**30.14**	**28.01**	**0.0870**	**0.1775**	**0.2375**	0.3884	0.5301	**0.5943**
P2P-E	DS (386, 22, 4)	8.81	6.40	0.86	**9.77E-06**	0.0	0.0	**0.0306**	**0.4372**	0.6667
	RS-Greedy	9.10	5.22	0.86	6.83E-06	0.0	0.0	0.0267	0.3593	0.6667
	RS-GRASP	**9.22**	**6.41**	**1.29**	6.93E-06	0.0	**0.5**	0.0270	**0.4372**	**0.8333**
Web	DS (240, 105, 18)	52.15	47.62	10.20	**0.0266**	**0.2160**	0.4178	**0.2274**	**0.4759**	0.7254
	RS-Greedy	41.57	22.56	8.69	0.0027	0.0082	0.2525	0.0710	0.1225	0.6144
	RS-GRASP	**53.96**	**48.68**	**14.11**	0.0153	0.1246	**1.0000**	0.1296	0.3996	**1.0000**



(a) {C. Faloutsos, R. Agrawal} δ=1, Δ=1, $\bar{\lambda}$=6.7

(b) {C. Faloutsos, J. Leskovec} δ=0.78, Δ=0.51, $\bar{\lambda}$=5.0

(c) {J. Han, C. C. Aggarwal} δ=0.91, Δ=0.78, $\bar{\lambda}$=6.0

(d,e) {J. Widom, J. D. Ullman} δ=1, Δ=1, $\bar{\lambda}$=6.7 δ=0.8, Δ=0.58, $\bar{\lambda}$=9.0

Table 6.3: Robust DBLP subgraphs returned by our GRASP-RLS algorithm when seeded with the indicated authors.

Chapter 7

Sparse Feature Graph

In this chapter, we introduce Sparse Feature Graph with the application in redundant feature removal for high dimensional data. The redundant features existing in high dimensional datasets always affect the performance of learning and mining algorithms. How to detect and remove them is an important research topic in machine learning and data mining research. Based on the framework of sparse learning based unsupervised feature selection, Sparse Feature Graph is introduced not only to model the redundancy between two features, but also to disclose the group redundancy between two groups of features. With SFG, we can divide the whole features into different groups, and improve the intrinsic structure of data by removing detected redundant features. With accurate data structure, quality indicator vectors can be obtained to improve the learning performance of existing unsupervised feature selection algorithms such as multi-cluster feature selection (MCFS).

7.1 Chapter Introduction

For unsupervised feature selection algorithms, the structure of data is used to generate indication vectors for selecting informative features. The structure of data could be local manifold structure [138] [139], global structure [140] [141], discriminative information [142] [143] and etc. To model the structure of data, methods like Gaussian similarity graph, or k-nearest neighbor similarity graph are very popular in machine learning research. All these similarity graphs are built based on the pairwise distance like Euclidean distance ($\mathcal{L}_2$ norm) or Manhattan distance ($\mathcal{L}_1$ norm) defined between two data samples (vectors). As we can see, the pairwise distance is crucial to the quality of indication vectors, and the success of unsupervised feature selection depends on the accuracy of these indication vectors.

When the dimensional size of data becomes high, or say, for high dimensional datasets, we will meet the curse of high dimensionality issue [144]. That means the differentiating ability of pairwise distance will degraded rapidly when the dimension of data goes higher, and the nearest neighbor indexing will give inaccurate results [145] [146]. As a result, the description of data structure by using similarity graphs will be not precise and even wrong. This create an embarrassing *chicken-and-egg* problem [147] for unsupervised feature selection algorithms: "the success of feature selection depends on the quality of indication vectors which are related to the structure of data. But the purpose of feature selection is to giving more accurate data structure."

Most existing unsupervised feature selection algorithms use all *original features* [147] to build the similarity graph. As a result, the obtained data structure information will not as accurate as the intrinsic one it should be. To remedy this problem, dimensionality reduction techniques are required. For example, Principal Component Analysis (PCA) and Random Projection (RP) are popular methods in machine learning research. However, most of them will project the data matrix into another (lower dimensional) space with the constraint to approximate the original pairwise similarities. As a result, we lose the physical meaning or original features and the meaning of projected features are unknown.

In this chapter, we proposed a graph-based approach to reduce the data dimension by removing redundant features. Without lose of generality, we categorize features into three groups [148]: *relevant feature*,*irrelevant feature* and *redundant feature*. A feature $\boldsymbol{f_i}$ is relevant or irrelevant based on it's correlation with indication vectors (or target vectors named in other articles) $\boldsymbol{Y} = \{\boldsymbol{y_i}, i \in [1, k]\}$. For supervised feature selection algorithms [149] [23] [150], these indication vectors usually relate to class labels. For unsupervised scenario [151] [152], as we mentioned early, they follow the structure of data. Redundant features are features that highly correlated to other features, and have no contribution or trivial contribution to the target learning task. The formal definition of redundant feature is by [153] based on the Markov blanket given by [154].

Based on the philosophy of sparse learning based MCFS algorithm, a feature could be redundant to another single feature, or to a subset of features. In this work, we propose a graph based approach to identify these two kind of redundancy at the same time. The first step is to build a Sparse Feature Graph (SFG) at feature side based on sparse representation concept from subspace clustering theory [25]. Secondly, we review the quality of sparse representation of each single feature vector and filtered out those failed ones. In the last, we defined Local Compressible Subgraphs (LCS) to represent those local feature

groups that are very redundant. Moreover, a greedy local search algorithm is designed to discover all those LCSs. Once we have all LCSs, we pick the feature which has the highest node in-degree as the representative feature and treat all other as redundant features. With this approach, we obtain a new data matrix with reduced size and alleviate the curse of dimensional issues.

To be specific, the contribution of this chapter can be highlighted as:

- We propose sparse feature graph to model the feature redundancy existing in high dimensional datasets. The sparse feature graph inherits the philosophy of sparse learning based unsupervised feature selection framework. The sparse feature graph not only records the redundancy between two features but also show the redundancy between one feature and a subset of features.

- We propose local compressible subgraph to represent redundant feature groups. And also design a local greedy search algorithm to find all those subgraphs.

- We reduce the dimensionality of input data and alleviate the curse of dimensional issue through redundant features removal. With a more accurate data structure, the *chicken-and-egg* problem for unsupervised feature selection algorithms are remedied in certain level. One elegant part of our proposed approach is to reduce the data dimension without any pairwise distance calculation.

- Abundant experiments and analysis over twelve high dimensional datasets from three different domains are also presented in this study. The experiment results show that our method can obtain better data structure with reduced size of dimensionality, and proof the effectiveness of our proposed approach.

The rest of this chapter is organized as follows. The first section describe the math notation used in our work. The Section 2 introduces the background , motivation and preliminaries of our problem. In Section 3, we define the problem we are going to solve. In Section 4, we present our proposed sparse feature graph algorithm and discuss the sparse representation error problem. We also introduce the local compressible subgraph and related algorithm. The experiment results are reported in Section 5, and a briefly reviewing of related works is given in Section 6. Finally, we conclude our work in last Section 7.

7.2 Related Works

Remove redundant features is an important step for feature selection algorithms. Prestigious works include [153] which gives a formal definition of redundant features. Peng et al. [150] propose a greedy algorithm (named as mRMR) to select features with minimum redundancy and maximum dependency. Zhao et al. [155] develop an efficient spectral feature selection algorithm to minimize the redundancy within the selected feature subset through $\mathcal{L}_{2,1}$ norm. Recently, researchers pay attention to unsupervised feature selection with global minimized redundancy [156] [157]. Several graph based approaches are proposed in [158], [159]. The most closed research work to us is [160] which build a sparse graph at feature side and ranking features by approximation errors.

7.3 Background and Preliminaries

7.3.1 Unsupervised Feature Selection

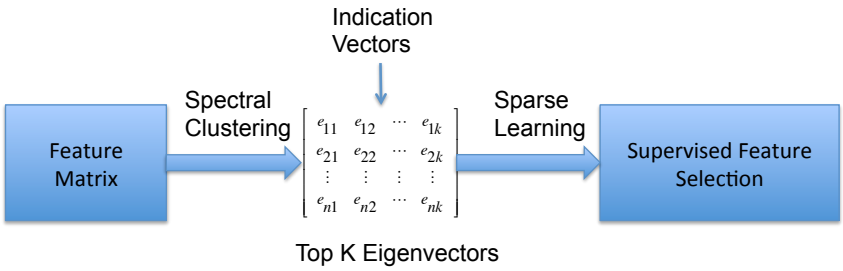


Figure 7.1: The framework of sparse learning based unsupervised feature selection.

In unsupervised feature selection framework, we don't have label information to determine the feature relevance. Instead, the data similarity or manifold structure constructed from the whole feature space are used as criteria to select features. Among all those algorithms of unsupervised feature selection, the most famous one is MCFS. The MCFS algorithm is a sparse learning based unsupervised feature selection method which can be illustrated as figure 7.1. The core idea of MCFS is to use the eigenvectors of graph Lapalcian over similarity graph as indication vectors. And then find set of features that can approximate these eigenvectors through sparse linear regression. Let us assume the input data has number K clusters that is known beforehand (or an estimated K value by the expert's domain knowledge). The top K

non-trivial eigenvectors, $\boldsymbol{Y} = [\boldsymbol{y_1}, \cdots, \boldsymbol{y_k}]$, form the spectral embedding $\boldsymbol{Y}$ of the data. Each row of $\boldsymbol{Y}$ is the new coordinate in the embedding space. To select the relevant features, MCFS solves K sparse linear regression problems between $\boldsymbol{F}$ and $\boldsymbol{Y}$ as:

$$\min_{\alpha_i} \|\boldsymbol{y}_i - \boldsymbol{F}\boldsymbol{\alpha}_i\|^2 + \beta\|\boldsymbol{\alpha}_i\|_1, \tag{7.1}$$

where $\boldsymbol{\alpha}_i$ is a n-dimensional vector and it contains the combination coefficients for different features $\boldsymbol{f}_i$ in approximating $\boldsymbol{y}_i$. Once all coefficients $\boldsymbol{\alpha}_i$ are collected, features will be ranked by the absolute value of these coefficients and top features are selected. This can be show by a weighted directed bipartite graph as following:

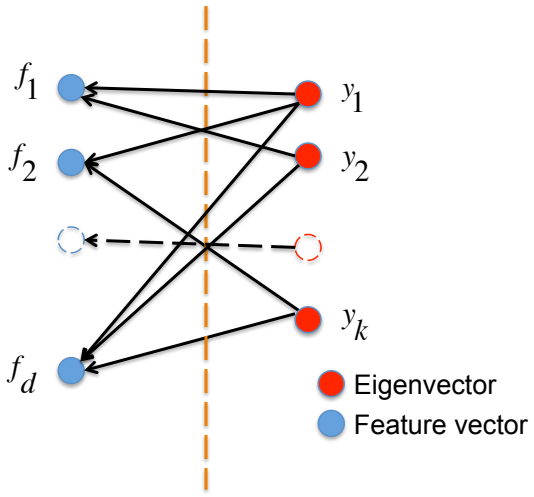


Figure 7.2: Sparse learning bipartite graph for MCFS.

7.3.2 Adaptive Structure Learning for High Dimensional Data

As we can seen, the MCFS uses whole features to model the structure of data. That means the similarity graph such as Gaussian similarity graph is built from all features. This is problematic when the dimension of data vector goes higher. To be specific, the pairwise distance between any two data vectors becomes almost the same, and as a consequence of that, the obtained structural information of data is not accuracy. This observation is the motivation of unsupervised Feature Selection with Adaptive Structure Learning (FSASL) algorithm which is proposed by Du et al. [147]. The idea of

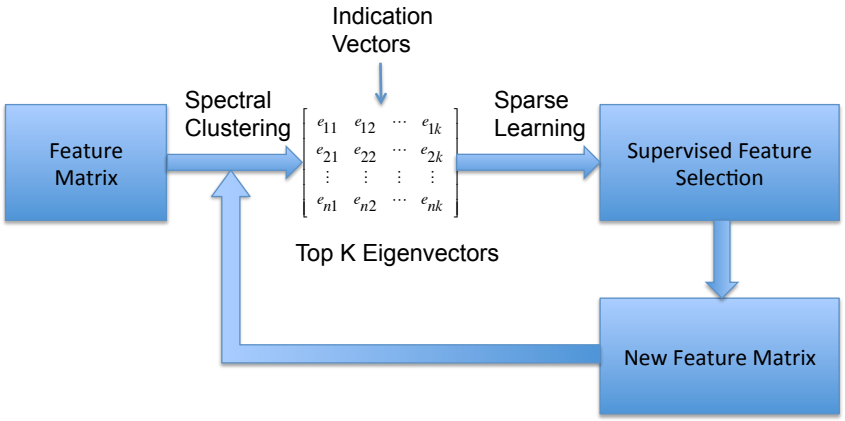


Figure 7.3: Unsupervised Feature Selection with Adaptive Structure Learning.

FSASL is to repeat MCFS iteratively with updating selected feature sets. It can be illustrated as following: FASAL is an iterative algorithms which keeps pruning irrelevant and noisy features to obtain better manifold structure while improved structural info can help to search better relevant features. FASAL shows better performance in normalized mutual information and accuracy than MCFS generally. However, it's very time consuming since it is an iterative algorithm includes many eigen-decompositions.

7.3.3 Redundant Features

For high dimensional data $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, it exists information redundancy among features since $d \ll n$. Those redundant features can not provide further performance improvement for ongoing learning task. Instead, they impair the efficiency of learning algorithm to find intrinsic data structure.

In this section, we describe our definition of feature redundancy. Unlike the feature redundancy defined bt Markov blanket [153] which is popular in existing research works, our definition of feature redundancy is based on the linear correlation between two vectors (the "vector" we used here could be a feature vector or a linear combination of several feature vectors.) To measure the redundancy between two vectors $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$, squared cosine similarity[156] is used:

$$R_{ij} = cos^2(\boldsymbol{f}_i, \boldsymbol{f}_j). \tag{7.2}$$

By the math definition of cosine similarity, it is straightforward to know that a higher value of $R_{i,j}$ means high redundancy existing between $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$. For example, feature vector $\boldsymbol{f}_i$ and its duplication $\boldsymbol{f}_i$ will have R_{ii} value equals to one. And two orthogonal feature vectors will have redundancy value equals to

zero.

7.4 Problem Statement

In this work, our goal is to detect those redundant features existing in high dimensional data and obtain a more accurate intrinsic data structure. To be specific:

Problem 5. *Given a high dimensional data represented in the form of feature matrix $\boldsymbol{X}$, how to remove those redundant features $f_{(\cdot)} \in \boldsymbol{X}^T$ for unsupervised feature selection algorithms such as MCFS?*

Technically, the MCFS algorithm does not involve redundant features. However, the performance of MCFS depends on the quality of indication vectors which are used to select features via sparse learning. And those indication vectors are highly related to the intrinsic structure of data which is described by the selected features and given distance metric. For example, the MCFS algorithm uses all features and Gaussian similarity to represent the intrinsic structure. This is the discussed 'chicken-and-egg" problem [147] between structure characterization and feature selection. The redundant and noise features will lead to an inaccurate estimation of data structure. As a result, it's very demanding to remove those redundant (and noise) features before the calculation of indication vectors.

7.5 Algorithm

In this section, we present our graph-based algorithm to detect and remove redundant features existing in high dimensional data. First, the sparse feature graph that modeling the redundancy among feature vectors will be introduced. Secondly, the sparse representation error will be discussed. In the last, the local compressible subgraph is proposed to extract redundant feature groups.

7.5.1 Sparse Feature Graph (SFG)

The most popular way to model the redundancy among feature vectors is correlation such as Pearson Correlation Coefficient (PCC). The correlation value is defined over two feature vectors, and it's a pairwise measurement. However, there also exiting redundancy between one feature vector and a set of feature vectors according to the philosophy of MCFS algorithm. In this section, we present SFG, which model the redundancy not only between two feature vectors but also one feature vector and a set of feature vectors.

The basic idea of sparse feature graph is to looking for a sparse linear representation for each feature vector while using all other feature vectors as dictionary. For each feature vector $\boldsymbol{f}_i$ in features set $\boldsymbol{F} = [\boldsymbol{f_1}, \boldsymbol{f_2}, \cdots, \boldsymbol{f_d}]$, SFG solves the following optimization problem:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^{d-1}} \|\boldsymbol{f_i} - \boldsymbol{\Phi^i \alpha_i}\|_2^2, \quad \text{s.t. } \|\boldsymbol{\alpha_i}\|_0 < L, \tag{7.3}$$

where $\boldsymbol{\Phi}^i = [\boldsymbol{f_1}, \boldsymbol{f_2}, \cdots, \boldsymbol{f_{i-1}}, \boldsymbol{f_{i+1}}, \cdots, \boldsymbol{f_d}]$ is the dictionary of f_i and each column of $\boldsymbol{\Phi}^i$ is a selected feature from data matrix $\boldsymbol{X}$. L is a constraint to limit the number of nonzero coefficients. In SFG, we set it to the number of features d. The $\boldsymbol{\alpha}_i$ is the coefficient of each atom of dictionary $\boldsymbol{\Phi}^i$. This coefficient vector not only decides the edge link to $\boldsymbol{f_i}$ but also indicates the weight of that connection. The resulted SFG is a weighted directed graph and may have multiple components.

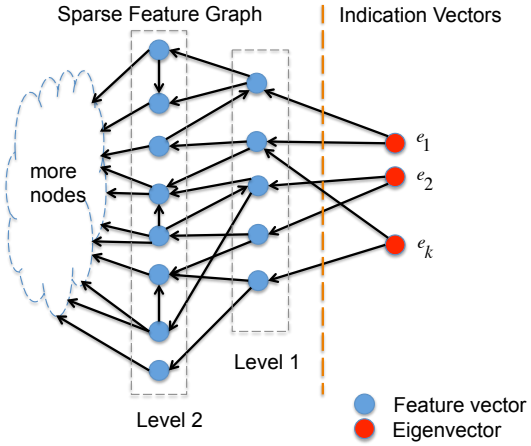


Figure 7.4: Sparse feature graph and its relation with indication vectors. Level 1 features are direct sparse representation of those calculated indication vectors. Level 2 features only have representation relationship with level 1 features but not with indication vectors.

To solve the optimization problem 7.3, we use Orthogonal Matching Pursuit (OMP) solver [66] here since the number of features in our datasets is larger than 1,000. We modify the stop criterion of OMP by checking the value change of residual instead of residual itself or the maximum number of supports. The reason is that we want the number of supports (or say, the number of edge connections) to follow the raw data property. Real world datasets are always noisy and messy. It's highly possible that several feature vectors may fail to find a correct sparse linear representation through OMP. If we set resid-

ual or maximum of supports as criteria, we can not differentiate the successful representations and the failed ones.

The OMP solver and SFG algorithm can be described as following.

Algorithm 14: Orthogonal Matching Pursuit (OMP)

Input : $\boldsymbol{\Phi} = [\boldsymbol{f_1}, \boldsymbol{f_2}, \cdots, \boldsymbol{f_{i-1}}, \boldsymbol{f_{i+1}}, \cdots, \boldsymbol{f_d}] \in \mathbb{R}^{n \times (d-1)}$, $\boldsymbol{f_i} \in \mathbb{R}^n, \epsilon$.

Output: Coefficient $\boldsymbol{\alpha}_i$.

Initialize residual difference threshold $r_0 = 1.0$, residual $\boldsymbol{q}_0 = \boldsymbol{f}_i$, support set $\Gamma_0 = \emptyset$, $k = 1$;

while $k \leq d-1$ *and* $|r_k - r_{k-1}| > \epsilon$ **do**

  Search the atom which most reduces the objective:

    $j^* = \arg\min_{j \in \Gamma^C} \left\{ \min_{\alpha} \|\boldsymbol{f}_i - \boldsymbol{\Phi}_{\Gamma \cup \{j\}} \boldsymbol{\alpha}\|_2^2 \right\}$;

  Update the active set:

    $\Gamma_k = \Gamma_{k-1} \cup \{j^*\}$;

  Update the residual (orthogonal projection):

    $\boldsymbol{q}_k = (I - \boldsymbol{\Phi}_{\Gamma_k} (\boldsymbol{\Phi}_{\Gamma_k}^T \boldsymbol{\Phi}_{\Gamma_k})^{-1} \boldsymbol{\Phi}_{\Gamma_k}^T) \boldsymbol{f}_i$;

  Update the coefficients:

    $\boldsymbol{\alpha}_{\Gamma_k} = (\boldsymbol{\Phi}_{\Gamma_k}^T \boldsymbol{\Phi}_{\Gamma_k})^{-1} \boldsymbol{\Phi}_{\Gamma_k}^T \boldsymbol{f}_i$;

  $r_k = \|\boldsymbol{q}_k\|_2^2$;

  $k \leftarrow k + 1$;

end

7.5.2 Sparse Representation Error

In our modified OMP algorithm 14, we set a new stop criterion of searching sparse representation solution for each feature vector $\boldsymbol{f}_i$. Instead of keep searching until arriving a minimization error, we stop running while the solver could not reduce the length of residual vector anymore. To be specific, the 2-norm of residual vector is monitored and the solver will stop once the change of this value small than a user specified threshold.

The reason we use this new stop criterion is that several feature vectors may not find correct sparse representation in current dataset, and the ordinary OMP solver will return a meaningless sparse representation when the maximum iteration threshold arrived. Since the goal of SFG is not to find a correct sparse representation for every feature vectors, we utilize the new stop criterion and add a filter process in our algorithm to identify those failed sparse representation.

Algorithm 15: Sparse Feature Graph

Input : Data matrix $\boldsymbol{F} = [\boldsymbol{f_1}, \boldsymbol{f_2}, \cdots, \boldsymbol{f_d}] \in \mathbb{R}^{n \times d}$;
Output: Adjacent matrix $\boldsymbol{W}$ of Graph $\boldsymbol{G} \in \mathbb{R}^{d \times d}$;

Normalize each feature vector $\boldsymbol{f_i}$ with $\|\boldsymbol{f_i}\|_2^2 = \mathbf{1}$;
for $i = 1, \cdots, d$ **do**
    Compute $\boldsymbol{\alpha}_i$ from OMP($\boldsymbol{F}_{-i}, \boldsymbol{f}_i$) using algorithm 14;
end
Set adjacent matrix $W_{ij} = \boldsymbol{\alpha_i(j)}$ if $i > j$, $W_{ij} = \boldsymbol{\alpha_i(j-1)}$, if $i < j$ and $W_{ij} = 0$ if $i == j$;

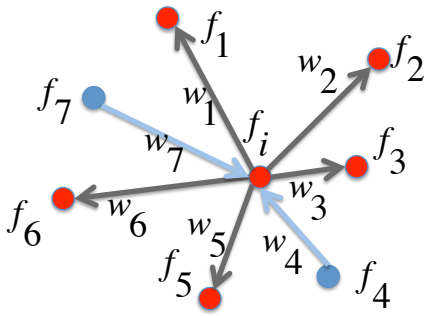


Figure 7.5: Illustration of sparse representation error. SFG is a weighted directed graph.

To identify those failed sparse representation, we check the angle between the original vector and the linear combination of its sparse representation. In the language of SFG, we check the angle between a node (a feature vector) and the weighted combination of its one-ring neighbor. Only the neighbors of out edges will be considered. This can be illustrated by following figure 7.5. As the example in Figure 7.5, node $\boldsymbol{f}_i$ has seven one-ring neighbors. But only

$bmf_1, bmf_2, \boldsymbol{f}_3, \boldsymbol{f}_5, \boldsymbol{f}_6$ are its sparse representation and $\boldsymbol{f}_4$ and $\boldsymbol{f}_7$ are not. Then the sparse representation error ζ is calculated by:

$$\begin{aligned} \boldsymbol{f}_i^* &= w_1\boldsymbol{f}_1 + w_2\boldsymbol{f}_2 + w_3\boldsymbol{f}_3 + w_5\boldsymbol{f}_5 + w_6\boldsymbol{f}_6, \\ \zeta &= \arccos(\boldsymbol{f}_i, \boldsymbol{f}_i^*). \end{aligned}$$

Once we have the SFG, we calculate the sparse representation errors for all nodes. A sparse representation is treated as fail if the angle ζ less than a user specified value. We will filter out these node which has failed representation by removing its out-edges.

7.5.3 Local Compressible Subgraph

We group high correlated features through *local compressible subgraphs*. The SFG $\boldsymbol{G}$ is a weighted directed graph. With this graph, we need to find all feature subsets that has very high redundancy. To archive this goal, we propose a local search algorithm with seed nodes to group those highly correlated features into many subgraphs which are named as *local compressible subgraphs* in this chapter. Our local search algorithm involves two steps, the first step is to sort all nodes by the in-degree. By the definition of SFG, the node with higher in-degree means it appears more frequently in other nodes' sparse representation. The second step is a local bread-first search approach which finds all nodes that has higher weight connections (in and out) to the growing subgraph. The detail subgraph searching algorithm can be described by: In Alg. 16, function

Algorithm 16: Local Compressible Subgraphs.

Input : Weighted directed graph $G = (V, E)$, edge weight threshold θ;
Output: Local compressible subgraphs C .

```
1  Tag all nodes with initial label 0;
2  Sort the nodes by its in-degree decreasingly;
3  current_label = 1;
4  for n = 1 : |V| do
5  |  if label(n) ! = 0 then
6  |  |  continue;
7  |  end
8  |  set label of node n to current_label;
9  |  BFS(n, θ, current_label);
10 |  current_label + = 1;
11 end
   /* current_label now has the maximum value of labels.      */
12 for i = 1 : current_label do
13 |  Extract subgraph c_i which all nodes have label i;
14 |  if |c_i| > 1 then
15 |  |  add c_i to C;
16 |  end
17 end
```

$label(n)$ check the current label of node n, and $BFS(n, \theta, current\_label)$ function runs a local Breadth-First search for subgraph that has edge weight large than θ.

7.5.4 Redundant Feature Removal

The last step of our algorithm is to remove the redundant features. For each local compressible subgraph we found, we pick up the node which has the highest in-degree as the representative node of that local compressible subgraph. So the number of final feature vectors equals to the number of local compressible subgraphs.

7.6 Experiments

In this section, we present experimental results to demonstrate the effectiveness of our proposed algorithm. We first evaluate the spectral clustering performance before and after applying our algorithms. Secondly, we show the performance of MCFS with or without our algorithm. In the last, the properties of generated sparse graphs and sensitivity of parameters are discussed.

7.6.1 Experiment Setup

Datasets. We select twelve real-world high dimensional datasets [161] from three different domains: Image, Text and Biomedical. The detail of each dataset is listed in Table 7.1. The datasets have sample size different from 96 to 8293 and feature size ranging from 1,024 to 18,933. Also, the datasets have class labels from 2 to 64. The purpose of this selection is to let the evaluation results be more general by applying datasets with various characteristics.

Name	#Sample	#Feature	#Class	Type
ORL	400	1024	40	Image
Yale	165	1024	15	Image
PIE10P	210	2420	10	Image
ORL10P	100	10304	10	Image
BASEHOCK	1993	4862	2	Text
RELATHE	1427	4322	2	Text
PCMAC	1943	3289	2	Text
Reuters	8293	18933	65	Text
lymphoma	96	4026	9	Biomedical
LUNG	203	3312	5	Biomedical
Carcinom	174	9182	11	Biomedical
CLL-SUB-111	111	11340	3	Biomedical

Table 7.1: High dimensional datasets.

Normalization. The features of each dataset are normalized to have unit length, which means $\|\boldsymbol{f_i}\|_2 = 1$ for all datasets.

Evaluation Metric. Our proposed algorithm is under the framework of unsupervised learning. Without loss of generality, the cluster structure of data is used for evaluation. To be specific, we measure the spectral clustering performance with Normalized Mutual Information (NMI) and Accuracy (ACC). NMI value ranges from 0.0 to 1.0, with higher value means better clustering performance. ACC is another metric to evaluate the clustering performance by measuring the fraction of its clustering result that are correct. Similar to NMI, its values range from 0 to 1 and higher value indicates better algorithm performance.

Suppose A is the clustering result and B is the known sample label vector. Let $p(a)$ and $p(b)$ denote the marginal probability mass function of A and B, and let $p(a, b)$ be the joint probability mass function of A and B. Suppose $H(A), H(B)$ and $H(A, B)$ denote the entropy of $p(a), p(b)$ and $p(a, b)$ respectively. Then the normalized mutual information NMI is defined as:

$$NMI(A, B) = \frac{H(A) + H(B) - H(A, B)}{max(H(A), H(B))} \tag{7.4}$$

Assume A is the clustering result label vector, and B is the known ground truth label vector, ACC is defined as:

$$ACC = \frac{\sum_{i=1}^{N} \delta(B(i), Map_{(A,B)}(i))}{N} \tag{7.5}$$

where N denotes the length of label vector, $\delta(a, b)$ equals to 1 if only if a and b are equal. $Map_{A,B}$ is the best mapping function that permutes A to match B.

7.6.2 Effectiveness of Redundant Features Removal

Our proposed algorithm removes many features to reduce the dimension size of all data vectors. As a consequence, the pairwise Euclidean distance is changed and the cluster structure will be affected. To measure the effectiveness of our proposed algorithm, we check the spectral clustering performance before and after redundant feature removal. If the NMI and ACC values are not changed to much and stay in the same level, the experiment results show that our proposed algorithm is correct and effective.

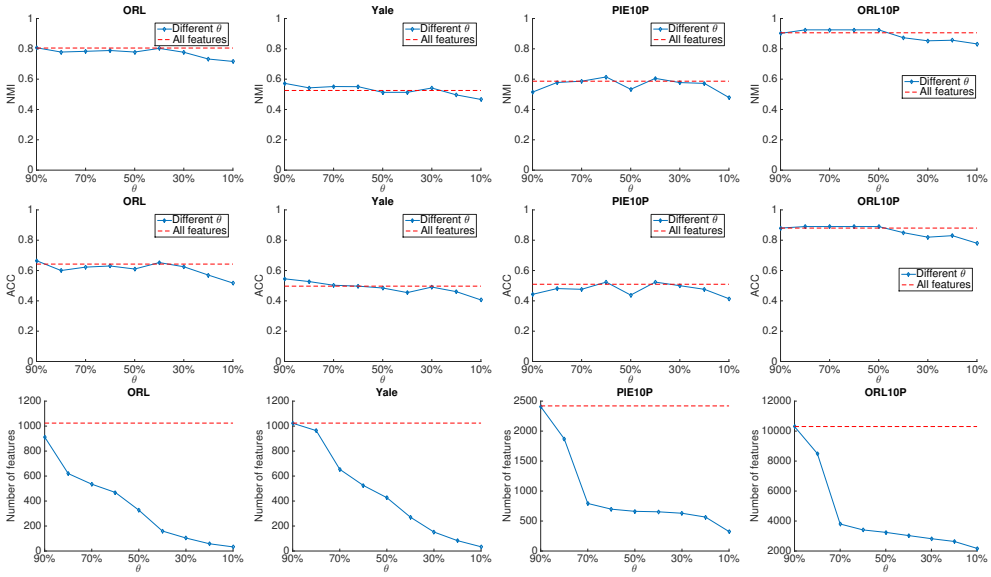


Figure 7.6: Spectral clustering performance of image datasets with different parameter θ. Top row: NMI; Middle row: ACC; Bottom row: number of features, the red dash line means the size of raw dataset.

The spectral clustering algorithm we used in our experiments is the Ng-Jordan-Weiss (NJW) algorithm [42]. The Gaussian similarity graph is applied here as the input and parameter σ is set to the mean value of pairwise Euclidean distance among all vectors.

Our proposed LCS algorithm includes a parameter θ which is the threshold of redundancy. It decides the number of redundant features implicitly, and affects the cluster structure of data consequently. In our experiment design, we test different θ values ranging from 90% to 10% with step size equal to 10%: $\theta = [0.9, 0.8, 0.7, \cdots, 0.1]$.

We present our experiment results for image datasets, text datasets, and biological datasets in Figure 7.6, Figure 7.7 and Figure 7.8 respectively. For each dataset, we show the NMI, ACC performance with different θ and comparing with original spectral clustering performance by using all features. From the experimental results, we can read that: **Even when θ is reduced to 30%, the NMI and ACC values are staying in same level as original data.** When θ equals to 30%, it means the edges of SFG that with weights (absolute value) in the highest 70% value range are removed. (It does not mean that 70% of top weights edges are removed). This observation validate the correctness of our proposed algorithm.

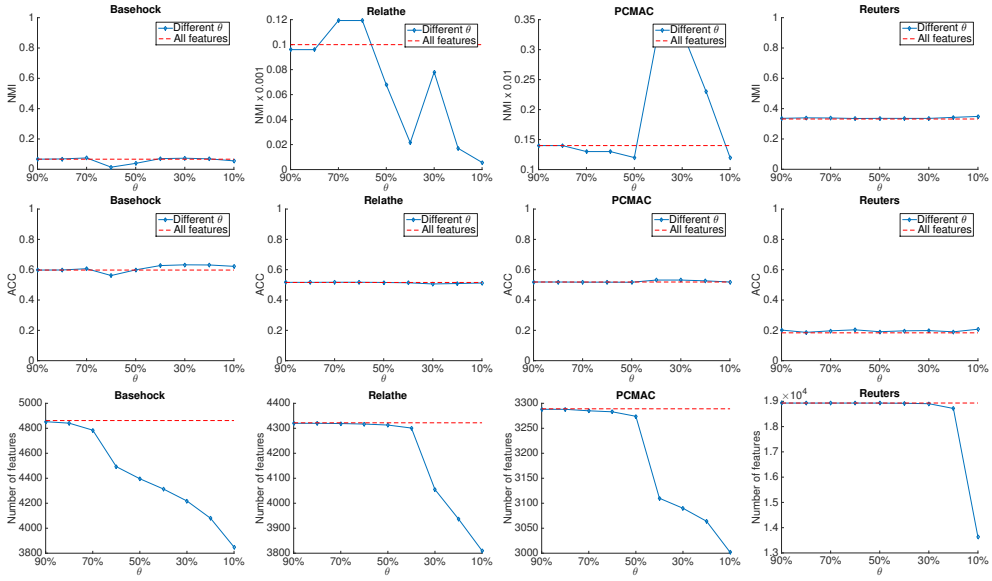


Figure 7.7: Spectral clustering performance of text datasets with different parameter θ. Top row: NMI; Middle row: ACC; Bottom row: number of features, the red dash line means the size of raw dataset.

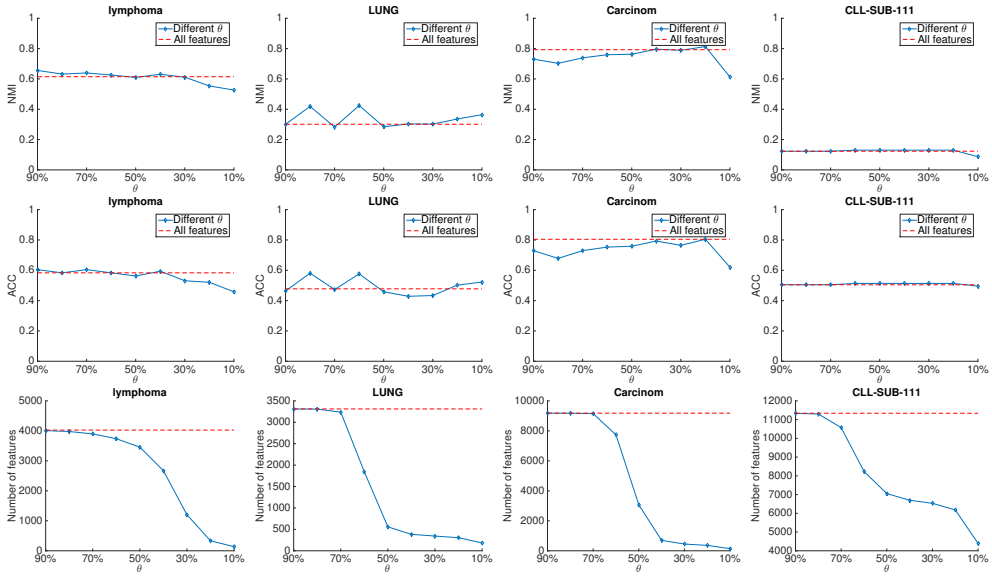


Figure 7.8: Spectral clustering performance of biomedical datasets with different parameter θ. Top row: NMI; Middle row: ACC; Bottom row: number of features, the red dash line means the size of raw dataset.

7.6.3 Performance of MCFS

Our proposed algorithm is targeting for unsupervised feature selection. And the quality of indication vectors (or the spectral clustering performance based on eigenvectors) is an important factor evaluate the effectiveness of our proposed algorithm. In this section, we evaluate the MCFS performance over the redundant feature removed data, and comparing with the raw data that without any feature removal.

The spectral clustering performance is measured for different input data from original whole feature data to processed ones by our proposed algorithm with different θ. We report the experiment results over image datasets and biological datasets in this section. For text datasets, the feature vectors of them are very sparse, and our eigen decomposition process are always failed and we only can collect partial results. For fair evaluation, we omit the experiment results of text datasets in this work. The result of MCFS performance shows from Table 7.2 to Table 7.17.

For each dataset, we set the number of selected features ranging from $[5, 10, 15, \cdots, 60]$, which has 11 different sizes in total. The parameter θ is configured from 0.9 to 0.1 with stepsize equals to 0.1.

We report the experimental results in tables (from Table 7.2 to Table 7.17). For each table, the first row means the number of features that used as input of MCFS. The first column is the number of selected features by MCFS algorithm. The baseline is in the second column, which is the testing result of MCFS algorithm with raw data. The hyphens in the tables means the number of selected features is larger than the feature size of input data, which means invalid test. To show the effectiveness of our algorithm, we also mark those NMI and ACC scores that larger or equals to baseline in bold text.

7.6.4 Sparse Representation Errors

With the design of our modified OMP solvers, there will be failed/wrong sparse representations existing in generated sparse feature graph. The meaning of these edge connections and edge weights are invalid. And they should be removed from the SFG since wrong connections will deteriorate the accuracy of feature redundancy relationship. To validate the sparse representation, we check the angle between original feature vector and the linear weighted summation resulted vector (or recover signal from sparse coding point of view) from its sparse representation. If the angle lower than a threshold, we remove all out-edges from the generated sparse feature graph. To specify the threshold, we learn it from the empirical results of our selected twelve datasets. The distribution (or histogram) result of angle values is presented in figure 7.9.

#f	1024	913	620	535	469	327	160	104	58	33
10	0.63	0.51	0.60	0.56	0.53	0.62	0.61	**0.65**	0.60	0.62
15	0.66	0.56	0.63	0.60	0.58	**0.67**	0.62	0.60	0.63	0.58
20	0.67	0.59	0.65	0.64	0.59	0.64	0.63	0.61	0.64	0.56
25	0.67	0.59	0.66	0.64	0.63	0.65	0.66	0.64	0.65	0.58
30	0.68	0.63	0.66	0.65	0.66	0.67	0.65	0.67	0.65	0.59
35	0.69	0.64	**0.70**	0.66	0.65	0.67	0.67	0.68	0.65	-
40	0.70	0.67	**0.71**	0.68	0.67	0.68	**0.70**	**0.70**	0.66	-
45	0.70	0.69	**0.70**	0.69	0.66	0.69	**0.70**	0.69	0.65	-
50	0.73	0.71	0.72	0.68	0.66	0.70	0.72	0.69	0.66	-
55	0.71	**0.74**	0.70	0.68	0.67	0.71	0.71	0.71	0.66	-
60	0.71	**0.74**	**0.71**	**0.72**	**0.71**	0.69	**0.72**	0.71	-	-

Table 7.2: NMI results of "ORL" dataset

#f	1024	913	620	535	469	327	160	104	58	33
10	0.38	0.28	0.36	0.31	0.28	**0.39**	**0.39**	**0.46**	**0.39**	**0.41**
15	0.45	0.33	0.41	0.40	0.34	0.43	0.40	0.38	0.42	0.36
20	0.47	0.34	0.43	0.43	0.35	0.43	0.41	0.39	0.43	0.32
25	0.48	0.35	0.45	0.44	0.37	0.42	0.47	0.41	0.45	0.34
30	0.47	0.40	0.42	0.42	0.43	0.47	0.43	0.45	0.42	0.35
35	0.49	0.41	0.48	0.46	0.44	0.44	0.47	0.47	0.42	-
40	0.51	0.46	**0.53**	0.48	0.46	0.45	0.48	**0.51**	0.43	-
45	0.49	0.47	**0.51**	**0.51**	0.44	0.48	**0.49**	**0.49**	0.43	-
50	0.55	0.51	0.52	0.47	0.47	0.50	0.52	0.48	0.46	-
55	0.53	**0.53**	0.51	0.46	0.45	0.48	0.50	**0.53**	0.46	-
60	0.51	**0.55**	**0.52**	**0.54**	0.51	0.47	**0.54**	0.51	-	-

Table 7.3: ACC results of "ORL" dataset.

#f	1024	1023	964	654	525	427	271	152	83	34
10	0.48	0.43	0.43	0.45	0.42	0.46	0.45	0.46	0.47	0.44
15	0.49	0.47	0.46	**0.51**	**0.49**	0.48	0.45	0.47	**0.50**	0.43
20	0.49	0.48	0.46	**0.55**	0.48	**0.51**	0.47	0.47	**0.51**	0.41
25	0.51	0.49	0.49	**0.52**	**0.52**	**0.52**	0.45	0.49	**0.54**	0.41
30	0.51	**0.51**	0.49	**0.54**	0.50	**0.51**	**0.51**	0.49	**0.50**	0.39
35	0.53	0.49	0.50	**0.54**	**0.53**	0.52	0.52	0.48	0.50	-
40	0.49	**0.50**	**0.51**	**0.53**	**0.58**	**0.55**	**0.55**	0.48	**0.51**	-
45	0.48	**0.51**	**0.51**	**0.56**	**0.59**	**0.57**	**0.52**	**0.52**	**0.49**	-
50	0.52	0.50	0.47	**0.53**	**0.59**	**0.53**	**0.53**	**0.56**	0.49	-
55	0.54	0.51	0.52	**0.55**	0.50	0.51	0.51	0.51	0.49	-
60	0.54	0.49	0.51	0.49	**0.54**	0.50	0.51	0.46	0.52	-

Table 7.4: NMI results of "Yale" dataset

#f	1024	1023	964	654	525	427	271	152	83	34
10	0.39	0.36	0.37	0.36	0.33	0.38	**0.41**	**0.40**	**0.41**	0.36
15	0.43	0.41	0.42	**0.44**	0.41	0.41	0.39	0.41	**0.46**	0.39
20	0.44	0.42	0.41	**0.48**	**0.44**	**0.44**	0.43	0.42	**0.44**	0.35
25	0.45	**0.45**	0.44	**0.46**	**0.47**	**0.45**	0.41	0.43	**0.49**	0.33
30	0.48	0.44	0.42	0.47	0.47	0.45	0.45	0.40	0.47	0.33
35	0.48	**0.48**	0.44	**0.50**	0.47	0.46	0.47	0.41	0.44	-
40	0.42	**0.44**	**0.45**	**0.50**	**0.55**	**0.48**	**0.53**	0.41	**0.44**	-
45	0.41	**0.48**	**0.46**	**0.51**	**0.53**	**0.54**	**0.49**	**0.47**	**0.42**	-
50	0.46	0.41	0.42	**0.48**	**0.56**	**0.50**	**0.46**	**0.52**	0.41	-
55	0.48	0.44	**0.48**	**0.48**	0.43	0.45	**0.49**	0.47	0.42	-
60	0.50	0.42	0.44	0.40	**0.50**	0.41	0.46	0.42	0.43	-

Table 7.5: ACC results of "Yale" dataset.

#f	2420	2409	1871	793	698	662	654	630	566	324
10	0.44	**0.48**	**0.55**	**0.53**	**0.58**	**0.56**	**0.54**	**0.61**	**0.50**	0.38
15	0.44	**0.61**	**0.57**	**0.50**	**0.58**	**0.58**	**0.55**	**0.59**	**0.53**	0.39
20	0.43	**0.56**	**0.61**	**0.59**	**0.60**	**0.56**	**0.62**	**0.59**	**0.56**	0.41
25	0.52	**0.61**	**0.61**	**0.64**	**0.61**	**0.60**	**0.58**	**0.58**	**0.54**	**0.43**
30	0.53	**0.61**	**0.62**	**0.57**	**0.62**	**0.62**	**0.60**	**0.53**	**0.63**	0.41
35	0.59	**0.60**	**0.59**	**0.60**	**0.63**	**0.61**	**0.60**	**0.62**	**0.64**	0.43
40	0.53	**0.60**	**0.58**	**0.57**	**0.66**	**0.62**	**0.59**	**0.62**	**0.69**	0.42
45	0.55	**0.61**	**0.61**	**0.62**	**0.60**	**0.64**	0.60	**0.64**	**0.65**	0.43
50	0.56	**0.63**	**0.62**	**0.68**	**0.64**	**0.62**	**0.58**	**0.63**	**0.66**	0.37
55	0.61	0.60	**0.62**	**0.69**	**0.62**	0.60	0.57	**0.65**	0.58	0.39
60	0.55	**0.64**	**0.63**	**0.64**	**0.60**	**0.63**	0.54	**0.63**	0.51	0.39

Table 7.6: NMI results of "PIE10P" dataset

#f	2420	2409	1871	793	698	662	654	630	566	324
10	0.39	**0.45**	**0.48**	**0.50**	**0.56**	**0.50**	**0.53**	**0.59**	**0.46**	0.39
15	0.39	**0.58**	**0.51**	**0.49**	**0.51**	**0.55**	**0.56**	**0.60**	**0.50**	**0.41**
20	0.36	**0.51**	**0.53**	**0.53**	**0.55**	**0.56**	**0.60**	**0.54**	**0.50**	**0.38**
25	0.45	**0.59**	**0.53**	**0.60**	**0.54**	**0.59**	**0.60**	**0.56**	**0.52**	0.40
30	0.50	**0.58**	**0.56**	**0.58**	**0.59**	**0.60**	**0.59**	**0.49**	**0.60**	0.40
35	0.48	**0.57**	**0.51**	**0.59**	**0.61**	**0.53**	**0.54**	**0.62**	**0.61**	**0.37**
40	0.42	**0.52**	**0.53**	**0.56**	**0.63**	**0.59**	**0.53**	**0.60**	**0.64**	0.38
45	0.44	**0.52**	**0.52**	**0.58**	**0.51**	**0.63**	**0.54**	**0.62**	**0.60**	**0.41**
50	0.44	**0.61**	**0.52**	**0.64**	**0.60**	**0.59**	**0.55**	**0.62**	**0.61**	0.37
55	0.46	**0.54**	**0.53**	**0.67**	**0.58**	**0.57**	**0.57**	**0.63**	**0.54**	0.37
60	0.49	**0.60**	**0.61**	**0.61**	**0.57**	**0.61**	**0.51**	**0.61**	**0.46**	0.35

Table 7.7: ACC results of "PIE10P" dataset.

#f	10304	10302	8503	3803	3408	3244	3030	2822	2638	2175
10	0.65	**0.78**	**0.77**	**0.76**	**0.77**	**0.80**	**0.74**	**0.72**	**0.75**	**0.73**
15	0.72	**0.82**	**0.79**	**0.78**	**0.81**	**0.83**	**0.79**	**0.81**	**0.75**	**0.79**
20	0.76	**0.81**	0.74	**0.78**	**0.84**	**0.83**	**0.81**	**0.76**	**0.80**	**0.78**
25	0.79	**0.84**	0.74	0.73	**0.82**	**0.86**	**0.88**	**0.83**	**0.86**	**0.81**
30	0.75	**0.77**	**0.82**	0.74	**0.88**	**0.82**	**0.83**	**0.83**	**0.86**	**0.86**
35	0.81	**0.81**	0.80	**0.83**	**0.85**	**0.83**	0.80	**0.82**	**0.85**	**0.85**
40	0.83	**0.88**	**0.84**	**0.84**	**0.90**	**0.86**	0.81	**0.93**	**0.84**	**0.87**
45	0.84	0.93	0.83	**0.85**	**0.91**	**0.86**	0.83	**0.88**	**0.84**	**0.86**
50	0.78	**0.88**	**0.88**	**0.87**	**0.89**	**0.86**	**0.82**	**0.90**	**0.84**	**0.83**
55	0.84	**0.89**	**0.86**	**0.89**	**0.91**	**0.89**	**0.88**	**0.86**	**0.84**	**0.86**
60	0.85	**0.88**	**0.86**	0.84	**0.85**	**0.91**	**0.85**	**0.88**	**0.86**	**0.85**

Table 7.8: NMI results of "ORL10P" dataset

#f	10304	10302	8503	3803	3408	3244	3030	2822	2638	2175
10	0.66	**0.74**	**0.81**	**0.75**	**0.75**	**0.69**	**0.72**	**0.70**	**0.69**	**0.67**
15	0.69	**0.85**	**0.76**	**0.78**	**0.78**	**0.86**	**0.80**	**0.75**	**0.73**	**0.75**
20	0.77	**0.84**	0.74	0.76	**0.80**	**0.80**	**0.78**	0.69	0.75	0.74
25	0.71	**0.79**	0.68	**0.74**	**0.78**	**0.86**	**0.84**	**0.82**	**0.82**	**0.74**
30	0.71	**0.71**	**0.77**	0.68	**0.86**	**0.77**	**0.81**	**0.77**	**0.82**	**0.81**
35	0.74	**0.74**	**0.74**	**0.76**	**0.81**	**0.77**	0.73	**0.76**	**0.82**	**0.78**
40	0.80	**0.85**	0.74	0.77	**0.87**	**0.80**	0.75	**0.89**	**0.80**	**0.83**
45	0.82	**0.89**	0.73	0.81	**0.88**	0.78	0.77	**0.86**	0.80	0.79
50	0.73	**0.80**	**0.80**	**0.74**	**0.86**	**0.79**	**0.74**	**0.88**	**0.81**	**0.77**
55	0.79	**0.85**	**0.82**	**0.86**	**0.89**	**0.87**	**0.80**	**0.82**	**0.81**	**0.79**
60	0.82	**0.84**	0.77	0.75	**0.82**	**0.89**	0.77	**0.84**	**0.82**	**0.82**

Table 7.9: ACC results of "ORL10P" dataset.

#f	4026	4009	3978	3899	3737	3456	2671	1203	334	136
10	0.51	**0.59**	**0.58**	**0.52**	0.50	0.50	**0.51**	0.50	0.50	0.49
15	0.55	**0.60**	**0.62**	**0.56**	**0.58**	**0.58**	**0.58**	**0.56**	0.47	0.52
20	0.60	**0.61**	**0.60**	0.57	**0.62**	**0.62**	**0.64**	0.58	0.58	**0.60**
25	0.63	0.59	**0.64**	0.60	**0.63**	0.58	**0.66**	0.57	0.56	0.53
30	0.59	**0.61**	**0.62**	**0.60**	**0.62**	**0.64**	**0.65**	**0.60**	**0.60**	**0.59**
35	0.61	**0.66**	**0.62**	0.60	**0.65**	**0.62**	**0.61**	**0.62**	0.56	0.53
40	0.64	0.60	**0.66**	0.63	0.61	0.63	**0.66**	0.61	0.58	0.55
45	0.58	**0.63**	**0.62**	**0.62**	**0.58**	**0.61**	**0.63**	**0.64**	**0.60**	0.57
50	0.65	0.60	0.61	0.61	0.56	0.63	0.61	0.63	0.58	0.54
55	0.63	0.60	0.61	0.62	0.60	0.60	**0.63**	0.60	0.58	0.58
60	0.60	**0.60**	**0.63**	**0.61**	**0.63**	0.59	**0.65**	0.59	0.57	0.57

Table 7.10: NMI results of "Lymphoma" dataset

#f	4026	4009	3978	3899	3737	3456	2671	1203	334	136
10	0.50	**0.57**	**0.56**	**0.53**	0.49	**0.51**	**0.51**	0.48	**0.50**	**0.50**
15	0.53	**0.62**	**0.59**	**0.58**	**0.56**	**0.59**	**0.58**	**0.55**	0.50	**0.53**
20	0.59	0.56	0.55	0.56	0.56	**0.59**	**0.59**	0.54	0.55	**0.59**
25	0.60	0.57	**0.62**	0.56	**0.62**	0.58	**0.64**	0.56	0.52	0.50
30	0.56	**0.60**	**0.58**	**0.58**	**0.59**	**0.61**	**0.65**	**0.59**	**0.57**	0.55
35	0.55	**0.62**	**0.59**	**0.58**	**0.61**	**0.60**	**0.57**	**0.59**	**0.55**	0.53
40	0.66	0.57	0.61	0.61	0.61	0.59	0.60	0.58	0.59	0.54
45	0.54	**0.60**	**0.60**	**0.58**	**0.55**	**0.60**	**0.62**	**0.59**	**0.56**	**0.54**
50	0.65	0.62	0.58	0.64	0.52	0.59	0.56	0.59	0.53	0.53
55	0.57	**0.60**	**0.65**	**0.60**	0.54	**0.57**	**0.65**	**0.59**	0.54	**0.59**
60	0.56	**0.58**	**0.64**	**0.58**	**0.61**	**0.57**	**0.67**	**0.56**	0.53	**0.57**

Table 7.11: ACC results of "Lymphoma" dataset.

#f	3312	3311	3309	3236	1844	559	384	344	305	183
10	0.42	**0.42**	**0.43**	**0.49**	**0.52**	**0.53**	**0.43**	**0.46**	**0.43**	0.25
15	0.54	**0.54**	0.53	0.51	0.51	0.51	0.45	0.52	0.38	0.21
20	0.51	**0.51**	**0.52**	**0.53**	0.41	0.49	0.36	**0.52**	0.38	0.20
25	0.51	**0.51**	**0.53**	0.48	0.42	**0.52**	0.40	0.48	0.35	0.26
30	0.47	**0.48**	**0.52**	**0.49**	0.41	0.37	**0.49**	**0.48**	0.41	0.24
35	0.46	0.38	**0.46**	**0.48**	0.39	**0.52**	**0.49**	0.38	0.35	0.27
40	0.49	**0.49**	**0.50**	0.46	0.43	0.40	0.38	0.35	0.40	0.29
45	0.36	**0.42**	0.33	**0.47**	**0.40**	0.33	**0.38**	0.35	0.35	0.31
50	0.45	**0.45**	**0.47**	**0.49**	**0.52**	0.32	0.40	0.36	0.35	0.31
55	0.44	**0.44**	**0.44**	**0.49**	**0.51**	0.33	**0.49**	0.31	0.30	0.31
60	0.47	0.46	0.45	**0.51**	**0.49**	0.33	0.39	0.32	0.31	0.35

Table 7.12: NMI results of "LUNG" dataset

#f	3312	3311	3309	3236	1844	559	384	344	305	183
10	0.71	**0.72**	**0.73**	**0.77**	**0.77**	**0.75**	0.68	0.65	0.66	0.56
15	0.81	**0.81**	0.79	0.72	0.73	0.72	0.67	0.65	0.58	0.48
20	0.71	**0.73**	**0.74**	**0.72**	0.69	0.69	0.61	0.60	0.58	0.39
25	0.71	**0.71**	**0.74**	0.67	0.69	0.68	0.59	0.61	0.56	0.49
30	0.66	**0.66**	**0.67**	**0.71**	**0.68**	0.56	0.59	0.59	0.61	0.43
35	0.64	0.60	0.63	**0.68**	**0.66**	0.60	0.58	0.56	0.53	0.49
40	0.65	**0.65**	**0.66**	**0.65**	0.64	0.57	0.54	0.54	0.56	0.46
45	0.60	**0.63**	0.57	**0.65**	**0.61**	0.52	0.54	0.52	0.52	0.49
50	0.65	**0.65**	0.63	**0.65**	**0.65**	0.48	0.57	0.53	0.53	0.52
55	0.61	**0.61**	0.59	**0.65**	**0.62**	0.48	0.59	0.48	0.49	0.49
60	0.64	0.63	0.63	**0.64**	0.62	0.51	0.55	0.49	0.48	0.51

Table 7.13: ACC results of "LUNG" dataset.

#f	9182	9180	9179	9150	7736	3072	697	449	360	144
10	0.70	**0.70**	**0.70**	0.69	0.67	0.64	0.66	0.65	0.66	0.47
15	0.71	0.70	**0.73**	**0.73**	**0.74**	0.66	0.67	0.70	0.66	0.52
20	0.77	**0.78**	**0.77**	0.72	0.75	0.72	0.73	0.71	0.73	0.54
25	0.74	**0.77**	**0.77**	**0.75**	**0.74**	0.71	**0.79**	**0.75**	**0.74**	0.53
30	0.69	**0.71**	**0.72**	**0.70**	**0.74**	**0.75**	**0.77**	**0.79**	**0.73**	0.54
35	0.77	0.76	0.76	0.76	0.74	**0.77**	**0.78**	**0.78**	**0.78**	0.60
40	0.75	0.74	**0.76**	**0.77**	0.74	**0.79**	**0.76**	**0.78**	**0.75**	0.59
45	0.77	0.76	0.74	**0.78**	0.74	**0.82**	**0.78**	**0.80**	**0.79**	0.57
50	0.79	0.76	0.75	0.75	**0.79**	0.76	**0.79**	**0.84**	**0.83**	0.58
55	0.75	**0.76**	**0.76**	0.74	**0.75**	**0.79**	**0.79**	**0.83**	**0.83**	0.59
60	0.74	0.72	**0.76**	0.73	**0.76**	**0.82**	**0.84**	**0.82**	**0.78**	0.62

Table 7.14: NMI results of "Carcinom" dataset

#f	9182	9180	9179	9150	7736	3072	697	449	360	144
10	0.63	**0.66**	0.62	0.61	**0.67**	0.60	0.60	0.59	**0.64**	0.48
15	0.67	0.57	**0.70**	0.66	**0.68**	0.63	0.57	**0.67**	0.64	0.53
20	0.70	0.68	**0.74**	0.66	**0.71**	**0.71**	0.64	**0.73**	**0.74**	0.56
25	0.70	**0.72**	**0.75**	0.69	**0.75**	0.64	**0.75**	**0.72**	**0.76**	0.51
30	0.61	**0.64**	**0.70**	**0.69**	**0.67**	**0.71**	**0.74**	**0.76**	**0.71**	0.52
35	0.76	0.74	0.74	0.74	0.70	0.75	0.70	**0.76**	**0.77**	0.57
40	0.72	**0.72**	**0.73**	**0.75**	0.69	**0.76**	0.66	**0.78**	0.71	0.56
45	0.75	0.74	0.70	**0.75**	0.74	**0.79**	0.72	**0.79**	**0.76**	0.55
50	0.74	**0.74**	0.70	0.72	**0.74**	0.66	**0.74**	**0.83**	**0.79**	0.56
55	0.73	**0.74**	**0.74**	0.72	0.71	0.72	0.72	**0.82**	**0.80**	0.56
60	0.70	0.61	**0.71**	0.66	**0.72**	**0.75**	**0.82**	**0.80**	**0.77**	0.55

Table 7.15: ACC results of "Carcinom" dataset.

#f	11340	11335	11301	10573	8238	7053	6697	6533	6180	4396
10	0.16	**0.16**	0.15	**0.26**	**0.18**	**0.22**	**0.20**	**0.20**	**0.20**	**0.21**
15	0.14	**0.14**	**0.15**	**0.26**	**0.18**	**0.28**	0.09	**0.24**	0.07	0.06
20	0.16	**0.16**	0.15	0.08	0.14	**0.21**	0.04	**0.31**	**0.16**	0.11
25	0.14	**0.14**	**0.15**	0.09	0.08	**0.22**	**0.23**	0.10	0.09	0.11
30	0.13	**0.13**	**0.13**	0.08	0.07	**0.18**	0.03	**0.14**	0.10	0.11
35	0.17	**0.17**	0.13	0.03	0.07	0.12	0.10	0.01	0.08	0.10
40	0.14	**0.14**	**0.14**	0.07	0.08	0.13	0.12	0.05	**0.14**	0.09
45	0.09	**0.09**	**0.18**	0.08	**0.11**	**0.10**	**0.13**	0.07	**0.12**	**0.09**
50	0.15	0.14	**0.15**	0.08	0.11	0.11	0.12	0.12	0.13	0.09
55	0.15	**0.15**	0.14	**0.21**	0.08	0.13	0.13	0.12	0.13	0.07
60	0.10	**0.10**	**0.14**	**0.15**	0.08	**0.10**	**0.12**	**0.12**	**0.14**	0.07

Table 7.16: NMI results of "CLL-SUB-111" dataset

#f	11340	11335	11301	10573	8238	7053	6697	6533	6180	4396
10	0.51	**0.51**	0.50	**0.54**	**0.59**	**0.57**	**0.58**	**0.55**	**0.51**	0.50
15	0.51	**0.51**	0.50	**0.57**	**0.55**	**0.62**	0.47	**0.59**	0.45	0.43
20	0.50	**0.50**	0.48	0.46	**0.50**	**0.54**	0.40	**0.59**	**0.54**	**0.50**
25	0.48	**0.48**	**0.51**	0.44	0.46	**0.54**	**0.57**	**0.50**	0.46	**0.50**
30	0.49	**0.49**	**0.49**	0.44	0.44	**0.53**	0.42	**0.51**	0.48	0.48
35	0.51	**0.51**	0.49	0.42	0.44	0.49	0.49	0.41	0.44	0.48
40	0.51	**0.51**	0.50	0.43	0.45	0.50	0.49	0.43	0.48	0.47
45	0.46	0.45	**0.52**	0.44	**0.46**	**0.47**	**0.51**	0.45	**0.47**	**0.47**
50	0.51	0.50	**0.51**	0.45	0.46	0.50	0.49	0.49	0.49	0.48
55	0.49	**0.49**	**0.50**	**0.54**	0.46	**0.50**	**0.50**	**0.49**	**0.49**	0.45
60	0.49	**0.49**	**0.50**	**0.53**	0.43	0.48	**0.49**	**0.49**	**0.50**	0.44

Table 7.17: ACC results of "CLL-SUB-111" dataset.

7.7 Chapter Summary

In this chapter, we propose sparse feature graph to model both one-to-one feature redundancy and one-to-many features redundancy. By separate whole

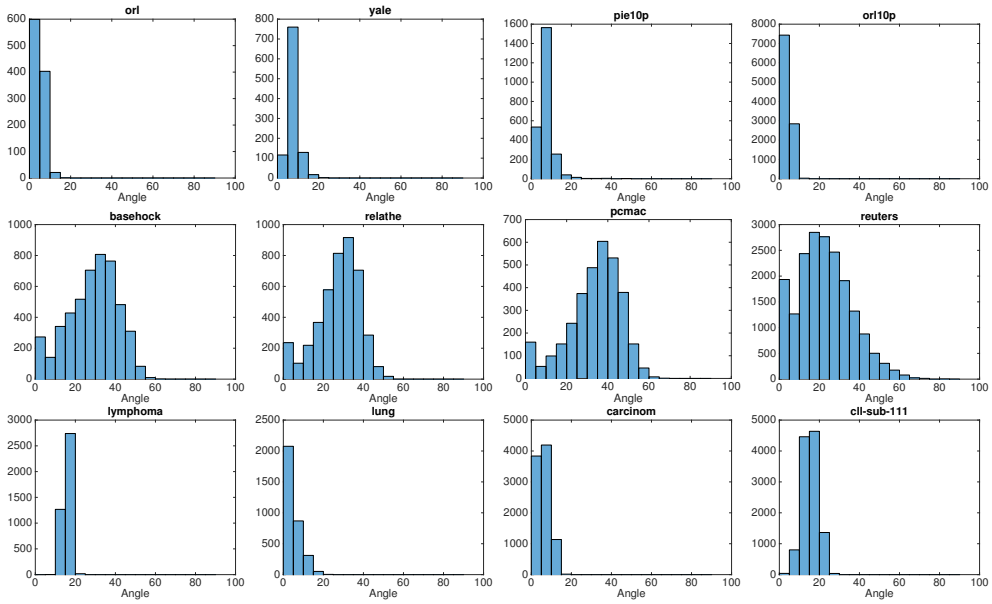


Figure 7.9: The distribution of angle between original feature vector and its sparse representation.

features into different redundancy feature group through local compressible subgraphs, we reduce the dimensionality of data by only select one representative feature from each group. One advantage of our algorithm is that it does not need to calculate the pairwise distance which is always not accurate for high dimensional datasets. The experiment results shows that our algorithm is an effective way to obtain accurate data structure information which is demanding for unsupervised feature selection algorithms.

Chapter 8

Capturing Properties of Names with Distributed Representations

In this chapter, we introduce the technique of *distributed name embeddings*, which represents names in a vector space such that the distance between name components reflects the degree of cultural similarity between them. We propose approaches to constructing such name embeddings using large volume of Email contact lists that record the human communication patterns and socializing preferences. We evaluate the cultural coherence of such embeddings, and demonstrate that they strongly capture gender and ethnicity information encoded in names. Finally, we propose two applications of the name embeddings, including a fake-contact generation process for security login challenges, and a large-scale look-alike names construction algorithm. Both applications generation names that respect cultural coherence and name popularity.

8.1 Chapter Introduction

Names are important. The names that people carry with them are arguably the strongest single facet of their identity. Names convey cues to people's gender, ethnicity, and family history. Hyphenated last names suggest possible marital relationships. Names even encode information about age, as social trends alter the popularity of given names.

In this chapter, we propose *distributed name embeddings* as a way to capture the cultural coherence properties of human names, such as gender and ethnicity. Our distributed name embeddings are trained on a real-world Email contact lists dataset which contains millions of "who-contact-who" records.

Male names	1th NN	2nd NN	3rd NN	4th NN	5th NN	Female names	1th NN	2nd NN	3rd NN	4th NN	5th NN
Andy	Andrew	Ben	Chris	Brian	Steve	Adrienne	Allison	Aimee	Amber	Debra	Amy
Dario	Pablo	Santiago	Federico	Hernan	Diego	Aisha	Aliyah	Nadiyah	Khadijah	Akil	Aliya
Elijah	Isaiah	Joshua	Jeremiah	Bryant	Brandon	Brianna	Brittany	Briana	Samantha	Jessica	Christina
Felipe	Rodrigo	Rafael	Eduardo	Fernando	Ricardo	Candy	Connie	Becky	Angie	Cindy	Christy
Heath	Brent	Chad	Brad	Brett	Clint	Chan	Wong	Poon	Ho	Wai	Yip
Hilton	Xooma	Eccie	Erau	Plexus	Gapbuster	Cheyenne	Destiny	Madison	Brittany	Taylor	Kayla
Isaac	Samuel	Israel	Eli	Esther	Benjamin	Dominque	Renarda	Lakenya	Lakia	Lashawna	Shatara
Jamal	Jameel	Kareem	Anmar	Khalifa	Nadiyah	Ebonie	Lakeshia	Tomeka	Ebony	Latasha	Shelonda
Lamar	Terrell	Derrick	Eboni	Tyree	Willie	Florida	Fairfield	Integrity	Beacon	Southside	Missouri
Mohammad	Shahed	Mohmmad	Ahmad	Rifaat	Farishta	Gabriella	Daniella	Vanessa	Marilisa	Isabella	Elisa
Moshe	Yisroel	Avraham	Gitty	Rivky	Zahava	Giovanna	Giovanni	Elisa	Paola	Giuliana	Mariangela
Rocco	Vito	Salvatore	Vincenza	Pasquale	Nunzio	Han	Jin	Yong	Sung	Huan	Teng
Salvatore	Pasquale	Nunzio	Gennaro	Vito	Tommaso	Kazuko	Keisuke	Junko	Yumi	Yuka	Tomoko
Thanh	Minh	Thuy	Thao	Ngoc	Khanh	Keren	Ranit	Galit	Haim	Zeev	Rochel

Table 8.1: The five nearest neighbors (NN) of representative male and female names in embedding space, showing how they preserve associations among Asian (Chinese, Korean, Japanese, Vietnamese), British, European (Spanish, Italian), Middle Eastern (Arabic, Hebrew), North American (African-American, Native American, Contemporary), and Corporate/Entity.

Each contact list encodes a particular individual's communicating customs and social interaction patterns.

Our key insight is that people tend to communicate more with people of similar cultural background and gender. Therefore if we embed names in the vector space so that the distance between names parts reflects the co-occurrence frequency, this embedding should capture aspects of culture and gender. Inspired by recent research advances in distributed word embeddings [162], we demonstrate the utility of name embeddings as convenient features to encode social/cultural information for classification tasks and other applications.

Table 8.1 illustrates the power of our distributed name embeddings, by presenting the five nearest-neighbors to a representative collection of male and female first names. These neighbors overwhelmingly preseve the gender and ethnicity of their center, capturing these properties without any labeled training data.

The major contributions of our work are:

- *Gender, ethnicity, and frequency preservation through name embeddings* – Through computational experiments involving ground truth data from the U.S. Census and Social Security Administration, we show that our name embeddings preserve such properties as gender and racial demographics for popular names and industrial sector for corporate contacts. Even more surprisingly, our embeddings preserve frequency of occurrence, a property that to the best of our knowledge has never been previously recognized in the distributed word embedding community.

- *Ethnic/gender homophily in email correspondence patterns* – Through large-scale analysis of contact lists, we establish that there is greater

than expected concentration of names of the same gender and race for all major groupings under study. We also establish that longer contact lists contain smaller concentrations of men, suggesting that women have larger correspondence circles than men.

- *Applications of name embeddings* – That names serve as people's primary societal identifier gives them power. Privacy requirements often make it undesirable or even illegal to publish people's names without their express permission. Yet there are often technical contexts where we need names which can be shared to represent things: to serve as placeholders in databases, demonstrations, and scientific studies.

  We employ name embeddings two different anonymization tasks in privacy and security applications: *replacement names* and *de novo name generation*. To preserve privacy, it is often desired to generate a replacement name for a given person. However when replacing names to anonymize (say) a medical study, dissonance is created when female names are replaced by male ones, and the names of elderly patients aliased by newly coined names. Generating names at random from component first/last name parts will not respect gender, ethnicity, or temporal biases: consider the implausibility of names like "Wei Hernandez" or "Roberto Chen". Name embeddings enable us to capture these cultural properties to generate meaningful replacements. We propose a new technique of representing the semantics of first/last names through *distributed name embeddings*. By training on millions of email contact lists, our embeddings establish cultural locality among first names, last names, and the linkages between them, as illustrated by examples in Table 8.1. Through nearest neighbor analysis in embeddings space, we can construct replacement aliases for any given name which preserve this cultural locality.

The outline of this chapter is as follows. Section 8.3 presents our approach to constructing name embeddings, including an evaluation of different approaches. Section 8.4 establishes that name embeddings preserve information concerning gender, ethnicity, and even frequency. In Section 8.6, we propose two applications: (1) a security login challenge application with our look-alike names, and (2) an efficient de novo look-alike names generation algorithm. Section 8.2 reviews related work. We conclude in Section 8.7 with discussions on remaining challenges.

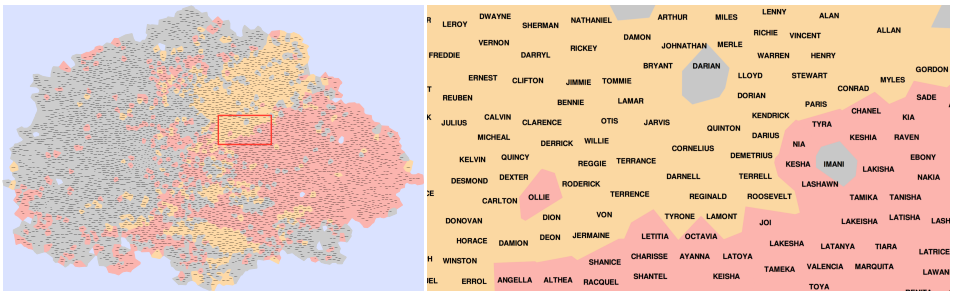


Figure 8.1: Visualization of the name embedding for the most frequent 5,000 first names from email contact data, showings a 2D projection view of name embedding (left). The pink color represents male names while orange denotes female names. Gray names have unknown gender. The right figure presents a close view along the male-female border, centered around African-American names.

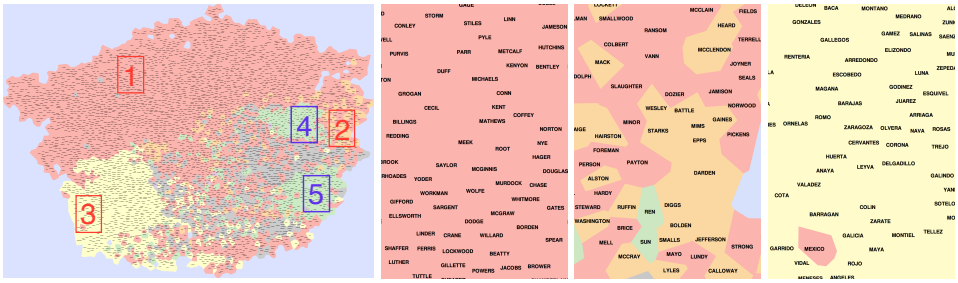


Figure 8.2: Visualization of the name embedding for the top 5000 last names, showings a 2D projection view of the embedding (left). Insets (left to right) highlight British 1, African-American 2 and Hispanic 3 names.

8.2 Related Work

Word and Graph Embeddings. Neural word embedding techniques, exemplified by the popular word2vec [163, 164], are now known to be effective in capturing syntactic and semantic relationships. Levy and Goldberg [165] found that the skipgram based word2vec embedding can be considered a matrix factorization technique, with the matrix to be factored containing the word-word point-wise mutual information. With this broad understanding of word2vec in mind, the technique is applicable to tasks beyond those of traditional natural language processing. It can be employed whenever there is a large amount of data consist of entities and their co-occurrence patterns. Our work on name-embedding is such an example. Another example is DeepWalk [166], a novel approach for learning latent representations of vertices in

a graph by constructing "sentences" via random walk on the graph.

8.3 Building Name Embeddings

8.3.1 Methodology

In our approach, each name part (first or last name) is embedded into high dimensional space as a high dimensional vector using word2vec [167]. Our hypothesis is that **people have a tendency to contact people of the same ethnicity and gender**. Consequently, when using the contact lists of millions of users as a text corpus, the resulting embedding of names would capture this tendency by placing names of the same gender and ethnicity close-by in the high-dimensional space.

8.3.2 Data Sources and Preparation

Datasets employed in our work are:

- *Contact Lists.* This set of data, here after referred to as the *contact lists*, is a proprietary sample of recent and/or frequent contacts of 2 million distinct email users of a major Internet company. To preserve privacy, the data does not contain the owners of the contact lists.
- *Census 1990.* The Census 1990 dataset [168] is a public dataset from US Census website. It records the frequently occurring surnames from US Census 1990. This dataset contains 4,725 popular female names and 1,219 popular male names.
- *Census 2000.* The Census 2000 dataset [169] is another public dataset from US Census website. It contains the frequently occurring 151,672 surnames from US Census 2000. Associated with each name is a distribution over six categories of races. The races are: White, Black, Asian/Pacific Islander (API), American Indian/Alaskan Native (AIAN), Two or more races (2PRACE), and Hispanics. In this study we refer to the races and ethnicity interchangeably.

Data Preparation. The contact lists include substantial noise in the name fields [170]. To improve the quality and integrity of the *contact list* data, we apply the following data cleaning processes to the original data following the guidance of US Census 2000 demographic report [171]: (1) Remove non-English characters; (2) Remove known special appellations, such as "Dr", "Mr", "MD", "JR", "I", "II" and "III"; (3) Remove middle names. First

name is the first part of a full name, and last name is the last part of it. For example, for name "Margarita M. Alvarez", only "Margarita" and "Alvarez" will be kept. After data cleaning and removing lists containing no names, 92% of the lists remains.

8.3.3 Word2vec Embeddings

The word2vec software [172] is an efficient tool to learn the distributed representation of words for large text corpus. It comes with two models: the Continuous Bag-of-Words model (CBOW) and the Skip-Gram (SG) model. The CBOW model predicts the current word based on the context while the Skip-Gram model does the inverse and maximizes classification of a word based on another word in the same context [163].

We start our analysis by using the cleaned *contact lists* and the word2vec software [172]. Each contact list is treated as a sentence, and together they form a text corpus. Unless otherwise stated, all results in the study are based on the CBOW model with the default word2vec parameter settings (see Section 8.3.4 for comparison of different models). The output of word2vec is a dense matrix of dimension $517,539 \times 100$, with each unique name represented as a row of the matrix.

Embedding Visualization. To understand the landscape of the name embeddings, we visualize the names as a 2D map. We used the stochastic neighborhood embedding [173] to reduce the original 100-dimensional embedding to 2D. We assign each name to a cluster using gender/ethnicity ground truth, and created the maps using `gvmap` [174].

Figure 8.1 (left) illustrates the landscape of first names. This visualization establishes that the embedding places names of the same gender close-by. Using Census data, we color male names orange, female names pink, and names with unknown gender gray. Overall names of the same gender form mostly contiguous regions, indicating that the embedding correctly capture gender information by placing names of the same gender close-by. Figure 8.1 (right) is an inset showing a region along the male/female border. We can see that "Ollie", which is considered a predominantly female name [175] per Census data (2:1 ratio of female/male instances), is placed in the male region, close to the male/female border.

we found that "Ollie" is more often a male name, and used as a nickname for "Oliver" or "Olivia". Hence our embedding is correct in placing it near the border. The embedding also correctly placed "Imani" and "Darian", two names not labelled by the Census data, near the border, but in the female/male regions, respectively. Per [175], "Imani" is a African name of Arabic origin, and can be both female and male, mainly female; "Darian" can also be female

and male, but mainly male, and is a variant of "Daren" and "Darien", among others.

Fig. 8.2 (left) presents a map of the top 5000 last-names. We color a name according to the dominant racial classification from the Census data. The top 5000 names contain four races: White (pink), African-American (orange), Hispanic (yellow), and Asian (green). Names without a dominant race are colored gray. The three cutouts in Fig. 8.2 highlight the homogeneity of regions by cultural group. The embedding clearly places White, Hispanic and Asian in large contiguous regions. African-American names are more dispersed. Interestingly, there are two distinct Asian regions in the map. Fig. 8.3 presents insets for these two regions, revealing that one cluster consists of Chinese names and the other Indian names. Overall, Fig. 8.1 and Fig. 8.2 show that our name embeddings capture gender and ethnicity information well.

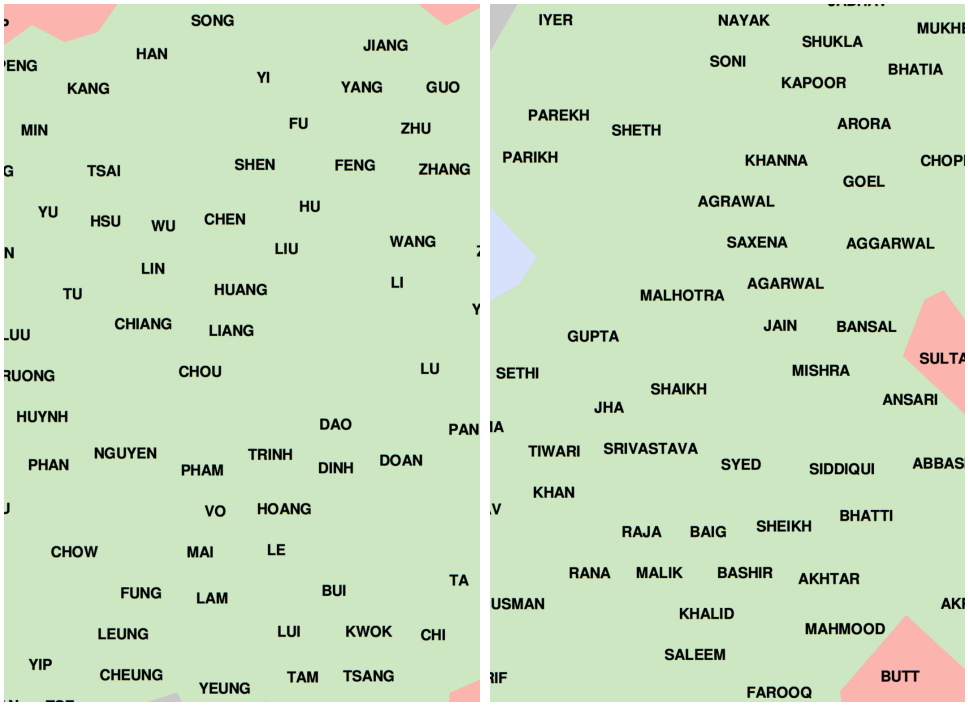


Figure 8.3: The two distinct Asian clusters. Left: Chinese/South Asian names (4 in Fig. 8.2). Right: Indian names (5 Fig. 8.2).

8.3.4 Evaluation of Different Word2vec Embeddings

The embedding from word2vec is influenced by two factors: (1) the input text, and (2) the word2vec parameter settings. To understand how these two factors

Variation	Popularity	Gender		Ethnicity (NN(1))			
		NN(1)	NN(10)	$P(W\|W)$	$P(B\|B)$	$P(A\|A)$	$P(H\|H)$
CBOW joint	0.6434(0.0007)	0.9092	**0.9360**	0.9362	**0.5939**	**0.7626**	**0.7543**
SG joint	0.6747(0.0002)	0.8844	0.9274	**0.9461**	0.4561	0.7208	**0.7543**
CBOW sep	0.6675(0.0003)	**0.9162**	0.9350	0.9299	0.4437	0.7167	0.6710
SG sep	**0.5776(0.0001)**	0.8844	0.9205	0.9217	0.3451	0.6797	0.6971

Table 8.2: Evaluation of different embedding variants. The bold text means the best value of each column.

influence the embedding, we evaluate the following variants of the word2vec embeddings:

- Set the word2vec model to be CBOW or SG.
- Generating joint embeddings of first names and last names using the contact lists as they are ("CBOW joint" or "SG joint").
- Generating embedding for first names and last names separately by including only first/last names in the contact lists ("CBOW sep" or "SG sep").

Metrics. To evaluate the quality of the embeddings with regard to look-alike names, we propose three metrics to measure gender, ethnicity and popularity similarities between real and look-alike names.

- *Gender Similarity.* The gender similarity is measured by precision at k, defined as the percentage of the k-nearest neighbors of a name having the same gender as the name itself.

- *Ethnicity Similarity.* The ethnicity similarity is calculated by precision at 1. For example, the precision for White names is defined as $P(W|W) = P(\text{1st NN is White}|\text{original name is White})$.

- *Popularity Similarity.* The popularity similarity is computed by the *Jensen-Shannon Divergence (JSD)* between name frequency distribution in real name population and name frequency distribution in nearest neighbors population (which describes how frequent a name appears in other name's nearest neighbor). To be specific, we sample 10k names randomly from the name list, with sampling probability proportional to real name frequency distribution. Then we record ten nearest neighbors (NN) for each of them and build the nearest neighbors population. We repeat this process 20 times and report the mean and deviation of JSD.

Results. We present the evaluation results in Table 8.2. As we expected, the joint variants generally perform best. However the differences between

the embedding variants are seen as relatively minor. In addition, the CBOW model generally outperformed the SG model for the majority of the nearest neighbor tests. Given these observations, in this study by default we use "CBOW joint" Note that while $P(B|B)$ (35%-59%) is generally much lower than $P(W|W)$ (92%-94%), considering that a randomly picked name from the contact list has a probability of 74% of being White but only a probability of 3% of being Black, $P(B|B)$ is actually significantly above the probability of a random name being black.

8.4 Properties of Name Embeddings

Earlier, in Fig. 8.1 and Fig. 8.2, we have provided visual evidence that the embedding is coherent, in the sense that it places names of similar gender and ethnicity close-by. In Section 8.3.4 we have also seen aggregate numerical evidence of this coherence. In this section we evaluate the coherence of the name embedding quantitatively and in more detail.

8.4.1 Gender Coherence and Analysis

We first examine the gender coherence of a subset of first names and their ten nearest neighbors. This subset of first names is the intersection between *contact lists* and *Census 1990*. It contains 1,146 unique male first names and 4,009 unique female first names. All names in the subset are ranked by their popularity as measured in the *Census 1990* data. Table 8.3 shows the gender coherence results, measured by precision at k, as a function of the population of the names, and k, the number of nearest neighbors. For example, the cell at $\{\leq 20\%, 2\}$ of Table 8.3 (left) reads 97. It means that for the top 20% most popular names, 97% of their nearest 2-neighbors have the same gender as them.

To save space, we only report the first two significant digits of each precision (e.g., 0.9742 becomes 97). In addition we color the cells of the tables based on the values. Within each table, we use warm colors for high values and cold color for low values. This gives us heat-maps through which it is easier to see the trend of how the precision varies with popularity of the first name, and the number of neighbors.

From Table 8.3, we observe that our proposed name embedding scheme shows strong gender coherence, especially for popular names. As we can see from the tables, the percentage of neighbors that have same gender as the original first name is very high for the top 30% most popular names comparing to a randomly assigned name (50%). On the other hand, the percentage

Top %	1	2	3	4	5	6	7	8	9	10
≤ 10%	100	99	98	98	98	98	98	98	98	98
≤ 20%	99	97	96	96	95	95	95	95	95	95
≤ 30%	96	95	94	93	93	92	93	92	92	91
≤ 40%	93	93	91	90	90	89	89	89	88	88
≤ 50%	89	89	86	85	85	84	84	84	83	83
≤ 60%	86	86	84	83	82	82	82	81	81	80
≤ 70%	82	81	79	79	78	77	77	76	76	76
≤ 80%	79	78	76	75	74	74	74	73	73	72
≤ 90%	76	75	73	73	72	71	71	71	70	70
All	73	72	70	69	69	68	68	67	67	67

Top %	1	2	3	4	5	6	7	8	9	10
≤ 10%	97	97	97	96	95	95	95	95	95	95
≤ 20%	91	91	91	90	89	89	89	89	88	88
≤ 30%	85	84	84	84	83	83	82	82	82	81
≤ 40%	80	79	79	78	78	77	77	77	76	76
≤ 50%	75	74	74	73	73	72	72	72	71	71
≤ 60%	69	69	68	68	67	67	66	66	66	66
≤ 70%	66	65	66	65	64	64	63	63	63	63
≤ 80%	62	61	61	61	60	60	59	59	59	59
≤ 90%	59	59	59	58	58	57	57	57	57	57
All	57	56	56	56	55	55	55	54	54	54

Table 8.3: Gender coherence of the name embedding for males (left) and females (right), as measured by the percentage of k-neighbors being male or female.

decreases when unpopular names are included, and also decreasing as the number of neighbors increases.

8.4.2 Ethnicity Coherence and Analysis

We evaluate the ethnicity coherence by examining the ethnicity of a last name and its ten nearest neighbors. The evaluation is based on the intersected last names between *Census 2000* and the *contact list*. The coherence values are computed by the percentage of nearest neighbors that have same ethnicity as a query name itself. To better understand the coherence trend, we use the same strategy as with gender coherence analysis, and examine the precision as a function of the popularity of the names, and the size of nearest neighbors. The results are presented in Table 8.4. In general, the top neighbors of a popular name tend to have a high probability of being in the same ethnicity group. The coherence for an ethnic group correlates positively with the popularity of the group in the *contact lists*. The coherence for AIAN and 2PRACE are poor, because they only account for 0.1% and 0.05% of the last names in the contact lists. Thus there is too little data to get the embedding correctly.

8.4.3 Name Popularity Analysis

We define two types of frequencies. The *real name frequency* is the frequency of names in the *Contact list*. The *replacement usage frequency* is the frequency of a name in the replacement name population. To measure the popularity preserving of word embedding, we calculate the *real name frequency* of a name

(R), the average *real name frequency* of its replacement names (ten nearest neighbors) (A), and its *replacement usage frequency* (U).

Two measurements, Pearson's correlation coefficient (PCC) and Spearman's rank correlation coefficient (SCC), are used to measure how well the popularity is preserved. The results are shown in Table 8.5. For example, R vs U means the correlation between the *real name frequency* and the *replacement usage frequency*. Overall we can see that the correlation between the *real name frequency* R and the *replacement usage frequency* U is higher than that for the *real name frequency* R and its neighbors' *real name frequency* A. This indicates that a popular name is very likely to appear in among the nearest neighbors of other names, even though its nearest neighbors are not necessarily popular names. A visualization of the relationship between the frequency of real names and replacements are given in the Appndix.

8.5 Cultural Coherence Mining

The visualizations of Figures 8.1 and 8.2, and quantitative analysis in the previous section have confirmed that our name embedding is able to capture both gender and ethnicity coherence information. Since the embedding is generated in a completely ***unsupervised*** manner by applying word2vec to millions email contact lists, it is surprising that the embedding can capture gender and ethnicity so well. Our hypothesis is that in aggregate, ***users exhibit a preference to communicate with people of the same ethnicity and gender***. In this section we attempt to verify this hypothesis.

8.5.1 Coherence in Gender Distribution

One important aspect of a name is its associated gender. To identify a name's gender, the first name is always preferred than last name in demographic studies [176]. Here, we follow this popular rule and use the first name to identify the gender of a given full name. The gender of a name could be male, female or unknown. The "unknown" names could be human names with no known ground truth gender, or non-human names, for example, "Microsoft" or "Amazon". To avoid the bias of any specific machine learning classifiers, we rely on dictionary look-up method to identify the gender of a name using the *Census 1990* data.

To start with, we look at the gender distribution of the *contact lists* as a function of the length of the contact list. Fig. 8.4 shows the average percentage of males as a function of the length of the contact lists (red dot curve), as well as as a function of the length of gender-identifiable names in the lists(blue

dot curve). It is seen that the longer the list, the less percentage of males it contains. We conjecture that the female users tend to have more contacts than male users. Second, we want to know the difference between the observed

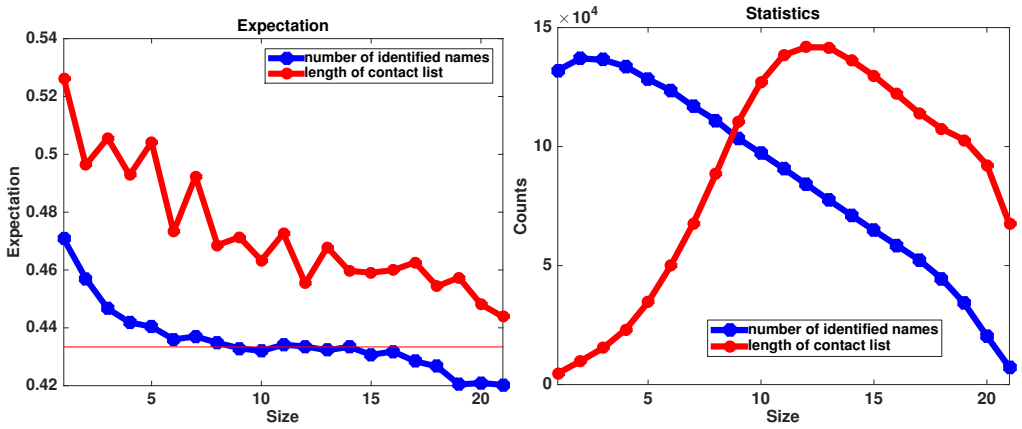


Figure 8.4: Left: the expectation of male name percentage as a function of the size of identified names (blue) and contact list lengths (red). Right: count of contact lists as a function of the size of gender identified names (blue) and contact list lengths (red).

contact lists and a randomly generated contact list. Our hypothesis is that users' contact lists exhibit a bias towards either male domination, or female domination. To test this hypothesis, we look at the frequency distribution of percentage of males in our mailing list, and compare with the null hypothesis. In Fig. 8.5 (left), we divide contact lists by a threshold based on the minimum number T of identifiable genders in the list. E.g., $T = 5$ means those contact lists with at least five gender-identifiable names. The distributions of the ratio of identifiable males in the contact lists with $T = 5$ and 10 are seen as the two lower curves. Clearly, the majority of the contact list has around 50% males. However, looking at these distribution along would not tell us whether the distributions have any bias. For this purpose, we need to compare them with the null hypothesis.

We generate the null distribution by assigning the gender of a name randomly following the gender distribution of the *contact list*. As a result, for a contact list with the number of identified names s equals to i and a probability of male of p_m, the probability that this list has j males is the binomial:

$$p(m = j|s = i) = C_i^j p_m^j (1 - p_m)^{i-j}.$$

Since the number of identified names varies for different contact lists, the

probability of having a ratio of $x \in [0, 1]$ male in the contact lists is:

$$p(x) = \frac{\sum_{i=1}^{21} \sum_{k=i*x \ \ is \ \ integer} p(s=i)p(m=k|s=i)}{\sum_{i=1}^{21} \sum_{j=1}^{i} p(s=i)p(m=j|s=i)}. \tag{8.1}$$

Here $p(s = i)$ is the percentage of contact lists having exactly i gender identifiable names. Fig. 8.5 (left) shows that the distributions based on the null hypothesis (the two higher curves) are spikier, with around 30% of the contact lists having 50% of males, compared with the observed 15%. Fig. 8.5 (right) shows the deviation of the observed distribution from the null hypothesis. It shows a clear bimodal pattern, confirming our hypothesis that contact lists on average exhibit a bias towards either male domination, or female domination, especially the latter.

To further verify the gender bias in observed contact lists, we model the observed number of males in all contact lists as a Binomial mixture model. Basically we assume that number of males in a contact list that we observe is generated by one of two separate Binomial distributions with different parameters, one representing female users and the other representing male users. We run Expectation-Maximization algorithm to find the best set of model parameters that explains the observed data most accurately. Here we only consider contact lists with more than 5 identifiable genders. After the EM algorithm converges, we generate synthetic data from the model and plot it alongside with the observed data in Figure 8.5. We observe that model fits the observed data quite well. Also the parameters of the fitted model suggest a strong gender-bias in contact lists, such that the probability of a contact in a male user's contact lists being male is 0.61, whereas the probability of a contact in a female user's contact lists being male is 0.27. The results also suggest that 47% of the observed contact lists belong to male users and 53% belongs to female users.

8.5.2 Coherence in Ethnicity Distribution

Another important aspect of a name is its ethnicity. While first names often reveal the gender, last names give a strong signal about ethnicity. In this subsection, we study the ethnicity distribution of the *contact lists*. We use *Census 2000* data set as ground truth and perform a similar look-up classification as we did for the gender analysis.

Just like the case for gender, we conjecture that users' contact lists on average exhibit a bias towards one ethnicity. To test this hypothesis, we look at the frequency distribution of percentage of a particular ethnicity in our mailing

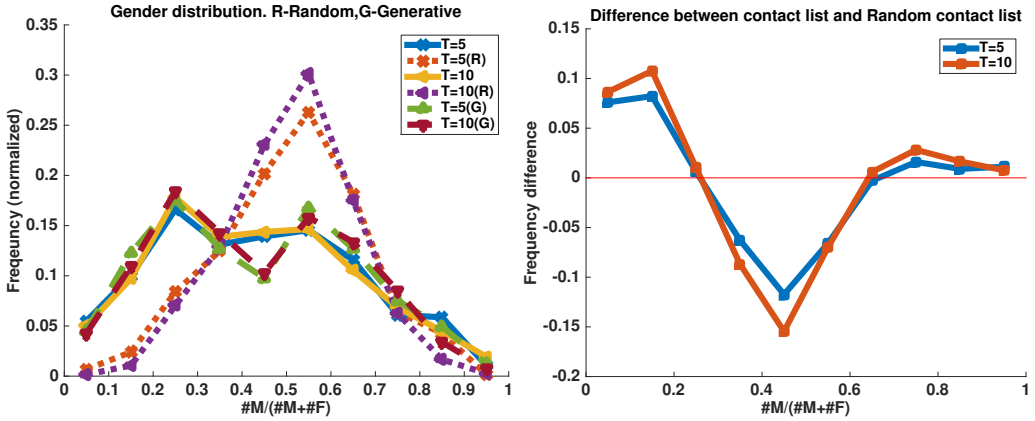


Figure 8.5: Left: the distribution of user's gender in contact lists data. Distributions with legend "R" are from binomial distribution with probability 0.5. Distributions with legend "G" are from binomial mixture model with parameters inferred using EM algorithm. Other distributions are from observation in the *contact lists*. Right: deviation from the null hypothesis.

list, and compare with the null hypothesis. The null hypothesis is constructed just like in the case for genders. Take the Hispanic ethnicity as an example. They constitute 14.75% of the names in the *contact lists*, so we set $p_m = 0.1475$ in (8.1). This allow us to plot the distribution for the null hypothesis, and compare with the observed Hispanic distribution. Fig. 8.6 shows the deviation of the observed distribution from the null hypothesis for "Black", "API" and "Hispanics" ethnic groups. It confirms that the *contact lists* have a tendency of containing lists that have higher than expected percentage of one of these ethnic groups, even though the bias is not quite as pronounced as in the case of genders.

8.6 Applications

Beyond the obvious applications of name embeddings as features for training classifiers and other models, we have employed name embeddings in two different security/privacy applications of generating realistic names respecting cultural constraints.

8.6.1 Replacement Name Generation

In *replacement name generation*, for a given a particular name (f, l) we seek to construct a look-alike name (f', l') with similar properties and veracity.

This task comes from a computer security application at a large Internet

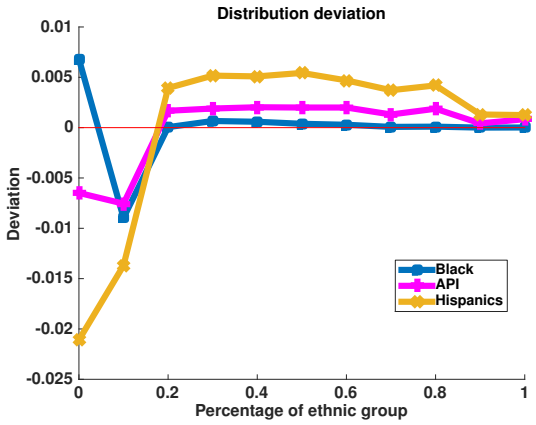


Figure 8.6: Deviation between observed distribution of percentage of names in ethnic groups "Black", "API" and "Hispanics", and the distribution from the null hypothesis, showing a bimodal pattern.

company. How might an email user who lost their password be able to convince the account provider of their identity as part of account recovery process? We reasoned that the genuine account holder should be able to distinguish the actual email contacts they have corresponded with from a background of imitation names. But this is only effective when the background names are culturally indistinguishable from the contacts, a property which did not hold under naive random name generation methods, as shown in Figure 8.7.

We propose to generated names preserve ethnic and cultural properties of the real contacts that they replace (middle), by finding replacement names among the nearest neighbors of the real contacts. For example "Amanda" is close to "Amy", and "Hsu" is close to "Chiang". With this scheme, the guessing task for attacker remains hard, because the imitation names look very similar to the real contacts.

8.6.2 De Novo Name Generation

A related class of applications concerns generating large sets of plausible names without starting templates, to serve as demonstration identities in information processing systems.

A synthetic name generation algorithm should have the following properties:

- *Scale* – The algorithm should be able to generate an arbitrarily large number of names, without high levels of repetition.
- *Respect population-level frequency-of-use statistics* – First name and last name tokens should be generated as per names in the target population.

Real Contacts	Proposed Challenge	Naive Challenge
Angela Chiang	Amanda Hsu	John Sander
Paresh Singh	Nirav Sharma	Steve Pignootti
Charles Wan	*Charles Wan*	*Charles Wan*
Yuda Lin	Joko Yu	Jeff Guibeaux
Lin Wong	Hua Li	Sam Khilkevich
Tony Kuang	David Feng	Mary Lopez
Hua Yim	Jie Fung	Ron Clemens

Figure 8.7: A security challenge question: "pick someone you contacted among the followings". Left: the contact list of a hypothetical user *wendy_wong@*. Middle: a replacement list generated using the technique proposed in this study (retaining one real contact *Charles Wan*). Right: a naively generated random replacement list. It is very easy to pick out the only Asian name "Charles Wan" from Naive Challenge.

- *Culturally-appropriate first/last-name linkage* – As we have seen, name token usage is not independent, but conditionally linked.
- *Privacy preservation* – No linkage between real and synthetic identities is permitted.

We propose the following approach. We construct a batch of m names simultaneously, where $m = 100$ is an appropriate value. We randomly sample m first and last name components as per the population distribution, here generated according to the U.S. Census distribution. We use the embedding-similarity between name components to weigh a complete $m \times m$ bipartite graph. By computing a maximum weight bipartite matching, we get m synthetic names with linkage informed by the geometry of the name embedding. The detail algorithm of de novo names generation can be described as following:

Table 8.6 presents a comparison of the first 25 synthetic men and women names produced by our methods versus `http://listofrandomnames.com`. We conducted a study by searching for each of the full names in Google and checking how many results are returned. Our rationale is that a plausible name should appear more often on the web than an implausible one. In the table, we marked in bold names that has at least 100 matches in Google search. In addition we use red color to show names that have no matchs at all. Clearly our name generator performs much better, with 47 bold names vs

Algorithm 17: De novo names generation algorithm.

Input: A set of full names: $\boldsymbol{S} = \{(f_1, l_1), (f_2, l_2), \cdots, (f_n, l_n)\} = (\boldsymbol{F}, \boldsymbol{L})$; Name embedding vectors; Number of required look-like names: K; Name batch size: m; Name popularity threshold: T.

Output: Number K look-alike full names: $\boldsymbol{U}$.

Calculate the frequency of unique first names in $\boldsymbol{F}$ and unique last names in $\boldsymbol{L}$.

Build the cumulative distribution function(CDF) of top $\boldsymbol{T}$ first names and last names.

for $i \leftarrow 1$ ***to*** $\frac{K}{m}$ **do**

  Randomly select m first names from $\boldsymbol{F}$ following the CDF of first names.

  Randomly select m last names from $\boldsymbol{L}$ following the CDF of last names.

  Build a bipartite graph of first names vs. last names, with edge weight the cosine similarity per word2vec.

  Find the best matches through Maximum weighted bipartite matching algorithm.

  Add the resulted best matching into $\boldsymbol{U}$.

end

return $\boldsymbol{U}$.

18 for http://listofrandomnames.com.

8.7 Chapter Summary

In this chapter, we propose a new technique for generating look-alike names through distributed name embeddings. By training on millions of email contact lists, our embeddings establish gender and cultural locality among names. The embeddings make possible construction of replacement aliases for any given name that preserve gender and cultural identity. Through large-scale analysis of contact lists, we establish that there is a greater than expected concentration of names of the same gender and race for all major groupings under study. Using the techniques developed in this study, we have constructed a collection of synthetic names, which will be released as an open resource upon the publication of this manuscript.

Top %	1	2	3	4	5	6	7	8	9	10
≤10%	96	96	96	96	96	96	96	96	96	96
≤20%	96	96	96	96	96	96	96	96	96	96
≤30%	96	96	96	96	96	96	96	96	96	96
≤40%	96	96	96	96	96	96	96	96	96	96
≤50%	95	95	95	95	95	95	95	95	95	95
≤60%	95	95	95	95	95	95	95	95	95	95
≤70%	95	95	95	95	95	95	95	95	95	95
≤80%	94	94	94	94	94	94	94	94	94	94
≤90%	94	94	94	94	94	94	94	94	94	94
All	94	94	94	94	94	94	94	94	94	94

White

Top %	1	2	3	4	5	6	7	8	9	10
≤10%	62	59	57	55	54	53	52	52	51	50
≤20%	65	63	60	59	58	58	57	57	56	56
≤30%	63	62	61	60	59	59	58	58	58	57
≤40%	63	62	61	60	59	59	59	59	59	58
≤50%	62	61	60	60	59	59	58	58	58	58
≤60%	61	61	60	60	59	59	58	58	58	58
≤70%	61	61	60	59	59	59	58	58	58	58
≤80%	60	60	59	59	58	58	58	58	57	57
≤90%	60	59	59	59	58	58	58	57	57	57
All	59	59	59	59	58	58	58	57	57	57

Black

Top %	1	2	3	4	5	6	7	8	9	10
≤10%	90	91	91	91	91	91	91	91	91	91
≤20%	89	89	89	89	89	89	89	89	89	89
≤30%	86	85	86	86	86	86	86	86	86	86
≤40%	83	83	84	84	84	84	84	84	84	84
≤50%	81	81	81	81	82	82	82	82	81	81
≤60%	80	80	80	80	80	80	80	80	80	80
≤70%	79	79	79	79	79	79	79	79	79	79
≤80%	78	78	78	78	78	78	78	78	78	78
≤90%	77	77	77	77	77	77	77	77	77	77
All	76	76	76	76	76	76	76	76	76	76

API

Top %	1	2	3	4	5	6	7	8	9	10
≤10%	19	19	22	22	22	21	20	20	19	18
≤20%	17	19	20	19	18	17	16	16	15	14
≤30%	15	18	18	17	16	15	14	14	13	13
≤40%	13	15	15	14	14	13	12	12	11	10
≤50%	12	13	13	11	11	11	10	10	9	9
≤60%	11	12	11	10	10	10	9	9	8	8
≤70%	10	11	10	10	9	9	8	8	7	7
≤80%	9	10	9	9	8	8	7	7	6	6
≤90%	9	9	9	8	8	7	7	7	6	6
All	8	9	8	8	7	7	6	6	6	6

AIAN

Top %	1	2	3	4	5	6	7	8	9	10
≤10%	54	46	44	44	45	46	47	47	45	45
≤20%	54	50	49	49	49	49	51	51	50	50
≤30%	56	53	52	51	51	51	52	52	50	50
≤40%	54	51	52	52	52	52	52	52	50	51
≤50%	58	52	54	55	54	54	53	54	52	53
≤60%	58	51	50	53	52	51	51	51	50	51
≤70%	57	52	53	54	53	52	52	52	51	52
≤80%	56	51	51	52	51	50	51	51	51	51
≤90%	53	49	50	50	50	49	49	50	49	49
All	51	48	49	49	48	47	48	48	48	48

2PRACE

Top %	1	2	3	4	5	6	7	8	9	10
≤10%	97	97	97	96	96	96	96	96	95	95
≤20%	95	94	94	94	94	94	93	93	93	93
≤30%	91	91	91	90	91	90	90	90	90	90
≤40%	89	89	88	88	88	88	88	88	88	88
≤50%	86	86	86	86	86	86	86	86	85	85
≤60%	83	83	83	83	83	83	83	82	82	82
≤70%	81	80	81	81	81	81	81	80	80	80
≤80%	79	79	79	79	79	79	79	79	79	79
≤90%	77	77	77	77	77	77	77	77	77	77
All	75	76	76	76	76	76	76	76	75	75

Hispanic

Table 8.4: Percentage of k-nearest ($k = 1, 2, \ldots, 10$) neighbors of a name that has the same ethnicity as itself, when restricting the name in the top p-percent ($p = 10, 20, \ldots, 90, All$) of names. API: Asian/Pacific Islander. AIAN: American Indian/Alaska Native. 2PRace: two or more races.

	PCC		SCC	
	R vs A	R vs U	R vs A	R vs U
First names	0.5813	0.7795	0.5170	0.5402
Last names	0.2260	0.4454	0.3444	0.3916

Table 8.5: Correlation of real names and replacement names frequencies.

listofrandomnames.com		Embedding-based de novo generation	
Male	Female	Male	Female
Keith Albro	Sibyl Bjork	Reginald Bouldin	**Ethel Agnew**
Sonny Bordner	**Amie Corrao**	**Max Bowling**	**Mabel Beaudoin**
Stanley Brummond	**Joselyn Custard**	**Dale Depriest**	Jolanda Boring
Reuben Carlucci	Marvella Deese	**Richard Diefenderfer**	**Lori Butz**
Darrell Chatmon	Holly Delman	**Michael Doutt**	**Diana Chao**
Jeffry Egnor	Kayleigh Derr	**Randall Drain**	**Cynthia Clay**
Russel Foye	Eugenia Fahnestock	**Anthony Hattabaugh**	Karin Combes
Hank Fries	Clemmie Formica	**Henry Humbert**	**Krista Emmons**
Patrick Gazaway	Gigi Fredericksen	**Jeremy Jacobsen**	**Rebecca Gagnon**
Roy Gilman	Marylyn Gersten	**Jeffrey Jimenez**	**Betty Grant**
Federico Gulley	**Elisabeth Harkness**	**Brian Kerns**	**Ruth Griffin**
Adalberto Hakes	**Almeda Ivy**	**Ronald King**	**Nancy Lantz**
Sylvester Kammer	Dot Klingbeil	**Elton Kolling**	**Joann Larsen**
Tanner Lundblad	Shay Krom	**Robert Kuhls**	**Deborah Lovell**
Jarod Man	Tessie Kush	**Fred Lawyer**	**Carla Mccourt**
Lee Mcclintock	Providencia Laughter	**Raymond Middleton**	**Caroline Mclaney**
Elvin Mcwhirt	Merlyn Lovings	**Andres Morales**	Denise Murders
Harry Nino	Milda Marcos	**John Morales**	**Mary Navarro**
Preston Pickle	Sierra Olivieri	**Alvin Morrison**	**Margarita Reyes**
Edgar Ramer	Pennie Pasquale	**Patrick Mulvey**	**Brenda Rock**
Rafael Rasheed	**Mallory Peralta**	**Victor Rahn**	**Selina Rubin**
Earnest Robert	Manda Stetz	**Nick Shick**	**Opal Sinkfield**
Ryan Seiber	Lissette Torrey	**Howard Siegel**	**Denise Stephens**
Kraig Tullos	Zelda Vanderburg	**Daniel Spady**	**Doretha Thurmond**
Howard Welk	Hee Weast	**Patricia Vargas**	Serina Webb

Table 8.6: Comparison of our de novo generated synthetic names and random names from website http://listofrandomnames.com. Bold names are mentioned over 100 times on the web, while red colored names appear in zero documents.

Chapter 9

Conclusion and Ongoing Works

This chapter summarizes our finished works and the future research directions. Our works focus on seeking sparse graph representation for real-world data. The ideal goal is to find a parameter-free method which can model the structure of data accurately and succinctly. Our algorithms are demonstrated to be efficient and scalable. They will have many potential applications in machine learning and data mining research area.

9.1 Contribution Summary

Our research works can be divided into three parts: (1) sparse graph representation, (2) graph structure analysis, and (3) applications. For the first part, we present three novel sparse graph representation algorithms. The proposed methods have competitive performance over original $\mathcal{L}_1$ graph and with lower construction cost. For the second part, we discuss the importance of dense subgraph when analyzing the structure of graph. This analysis is the key to understand the information presented in the data. For the last part, we successfully apply our research works to the application of unsupervised feature selection. Our proposed algorithms have great potential in semi-unsupervised learning research and graph data mining research. To be specific, our contributions include:

- We present an efficient locality preserving sparse graph construction algorithm to improve the spectral clustering performance.
- We demonstrate a structure preserving algorithm to generate sparse graph by using diffusion distance to capture the manifold structure of data, and compare the performance with Euclidean distance as metric.

- We introduce a greedy algorithm based on ranked dictionary to solve the scalable issue of constructing $\mathcal{L}_1$ graph.
- We propose a graph-based algorithm to solve the multi-source data integration problem in computational biology research. Our proposed method use the idea of semi-supervised learning to predict the labels of unlabeled data samples.
- We develop the robustness local subgraph to differentiate subgraphs with different sizes topologies. A greedy approach and heuristic local search algorithm are proposed to find all those robustness local subgraphs.
- We propose sparse feature graph to remove the redundant features for high dimensional data and reduce the dimensionality without calculating the pairwise distance between samples. This is very useful for high dimensional data as the quality of nearest neighbors becomes low when the size of dimensionality goes high.

The three novel sparse graph representation methods can be summarized as following:

Name	Distance metric	$\mathcal{L}_1$ solver	Summary
LOP-$\mathcal{L}_1$	Euclidean	Non-greedy	good for #samples $\leq$ 1000
SA-$\mathcal{L}_1$	Diffusion	Non-greedy	good for #samples $\leq$ 1000
Greedy-$\mathcal{L}_1$	Euclidean/Diffusion	Greedy	good for #samples $\geq$ 1000

Table 9.1: Summary of different sparse graph representation methods.

We also summarize the spectral clustering performance difference of them in the Table 9.3. For the results of LOP-$\mathcal{L}_1$ graph and SA-$\mathcal{L}_1$ graph in this table, we remove the non-negative constraint of sparse representation coefficient. From the results, we see that the Greedy-$\mathcal{L}_1$ with Euclidean distance has general better performance.

9.2 On-going Works

There are still many immediate and valuable research topics that can be included into our current works. The following are several potential subjects that can be directly extended from the works we have done so far.

Data	$\mathcal{L}_1$	LOP-$\mathcal{L}_1$			SA-$\mathcal{L}_1$			Greedy-$\mathcal{L}_1$ (Euclidean)			Greedy-$\mathcal{L}_1$ (Diffusion)		
		K=1M	K=2M	K=3M	K=1M	K=2M	K=3M	K=1M	K=2M	K=3M	K=1M	K=2M	K=3M
Iris	0.3615	0.4989	0.5719	0.5923	0.4287	**0.6103**	0.5827	0.395	0.4623	0.407	0.5106	0.4626	0.464
Bt	0.3615	0.4549	0.4205	0.4692	0.4002	0.5035	0.4411	**0.5473**	0.4517	0.5024	0.4197	0.4073	0.3839
Wine	0.3607	0.6442	0.7915	0.6354	0.7137	0.6276	0.6333	0.8943	**0.9072**	0.8566	0.6925	0.4291	0.6093
Glass	0.2287	0.2822	0.3414	0.3201	0.2205	0.2871	0.2714	0.2569	**0.3688**	0.3039	0.2991	0.3056	0.2918
Soybean	0.651	0.6447	0.656	0.6139	0.613	0.6791	0.6208	**0.6919**	0.6833	0.6775	0.5788	0.5493	0.5432
Vehicle	0.1743	0.169	0.097	0.1906	0.0913	0.1442	0.2132	0.1512	**0.2121**	0.2067	0.1438	0.1035	0.1244
Image	0.426	0.6178	**0.7064**	0.6918	0.5872	0.6914	0.6877	0.5821	0.6673	0.6649	0.4866	0.4483	0.3155

Table 9.2: NMI performance comparison. Bold values mean the best performance.

Data	$\mathcal{L}_1$	LOP-$\mathcal{L}_1$			SA-$\mathcal{L}_1$			Greedy-$\mathcal{L}_1$ (Euclidean)			Greedy-$\mathcal{L}_1$ (Diffusion)		
		K=1M	K=2M	K=3M	K=1M	K=2M	K=3M	K=1M	K=2M	K=3M	K=1M	K=2M	K=3M
Iris	0.68	0.8	0.6733	0.6867	0.7133	0.8067	0.68	0.6933	**0.7200**	0.68	**0.7200**	0.6533	0.64
BT	0.4245	0.4906	0.5	0.5283	0.4811	0.5377	0.5755	**0.6698**	0.4811	0.5943	0.4528	0.4906	0.4717
Wine	0.4888	0.8596	0.9438	0.8708	0.8876	0.8315	0.8708	**0.9719**	**0.9719**	0.9551	0.8989	0.7865	0.8596
Glass	0.3645	0.4252	0.4673	0.4299	0.3318	0.4533	0.4252	0.4579	0.4533	0.4346	0.4626	0.4813	**0.5187**
Soybean	0.5244	0.5212	0.4919	0.4137	0.4365	**0.5277**	0.4691	0.5244	0.4853	0.5016	0.4430	0.3746	0.4876
Vehicle	0.4019	**0.4799**	0.409	0.4433	0.4019	0.4137	0.4752	0.4539	0.4243	0.409	0.3664	0.3522	0.3605
Image	0.5176	0.621	0.7024	**0.7819**	0.6862	0.6838	0.6833	0.6348	0.7181	0.7043	0.5190	0.5524	0.3505

Table 9.3: ACC performance comparison. Bold values mean the best performance.

9.2.1 Subspace Learning with Sparse Graph

Similar to the graph construction process in Locally Linear Embedding (LLE), the $\mathcal{L}_1$ graph characterizes the neighborhood reconstruction relationship. In LLE, the graph is constructed by reconstructing each data sample with its k nearest neighbors of the samples within the ϵ-ball based on the ℓ^2 norm. LLE and its linear extension, called neighborhood preserving embedding (NPE) [177], both rely on the global graph parameter (k or ϵ). Following the idea of NPE algorithm, $\mathcal{L}_1$ graph can be used to develop a subspace learning algorithm as follows.

The general purpose of subspace learning is to search for a transformation matrix $P \in \mathbb{R}^{m\times d}$(usually $d \ll m$) for transforming the original high-dimensional data sample into another low-dimensional one. $\mathcal{L}_1$ graph uncovers the underlying sparse reconstruction relationship of each data sample, and it is desirable to preserve these reconstruction relationships in the dimensionality reduced feature space. Note that in the dimension reduced feature space, the reconstruction capability is measure by ℓ_2 norm instead of ℓ_1 norm for computational efficiency. Then the pursuit of transformation matrix can be formulated as the following optimization problem:

$$\min_{P^T XX^T P=I} \sum_{i=1}^{N} \|P^T x_i - \sum_{j=1}^{N} W_{ij} P_{x_j}^T\|^2, \tag{9.1}$$

where W_{ij} is determined by the constructed $\mathcal{L}_1$ graph. This optimization problem can be solved with generalized eigenvalue decomposition approach as:

$$XMX^T p_{m+1-j} = \lambda_j XX^T p_{m+1-j}, \tag{9.2}$$

where $M = (I-W)^T(I-W)$, and p_{m+1-j} is the eigenvector corresponding to the jth largest eigenvalue λ_j as well as the $(m+1-j)$th column vector of the matrix P. The derived transformation matrix is then used for dimensionality reduction as:

$$y_i = P^T x_i \tag{9.3}$$

where y_i is the corresponding low-dimensional representation of the sample x_i and finally the classification process is performed in this low-dimensional feature space with reduced computational cost.

9.2.2 Semi-supervised Learning with Sparse Graph

$\mathcal{L}_1$ graph is proved to be robustness with data noises and empirically has the potential to convey more discriminative information compared with conventional graphs based on k-nearest neighbor or ϵ-ball method [22]. These properties make $\mathcal{L}_1$ graph a good candidate for propagating the label information from labeled data to unlabeled data. Semi-supervised learning recently has attracted much attention, and is widely used for both regression and classification purposes. The main idea of semi-supervised learning is to utilize unlabeled data to improve the classification and generalization capability on the testing data. Commonly, the unlabeled data is used as an extra regularization term to the objective function which is from traditional supervised learning algorithms.

The unlabeled data are used to enlarge the vertex number of the $\mathcal{L}_1$ graph, and further enhance the robustness of the graph. Finally, the $\mathcal{L}_1$ graph based on both labeled and unlabeled data is used to develop semi-supervised learning algorithm. Here, we take marginal Fisher analysis(MFA) [178] as an example for the supervised part in semi-supervised learning. Similar to the philosophy [179], the objective for $\mathcal{L}_1$ graph based semi-supervised learning is defined as:

$$\min_P \frac{\gamma S_c(P) + (1-\gamma)\sum_{i=1}^{N} \|P^T x_i - \sum_{j=1}^{N} W_{ij} P^T x_j\|^2}{S_p(P)}, \tag{9.4}$$

where $\gamma \in (0,1)$ is a threshold for balancing the supervised term and $\mathcal{L}_1$ graph regularization term, and the supervised part is defined as:

$$S_c(P) = \sum_i \sum_{j \in N_{k_1}^+(i)} \|P^T x_i - P^T x_j\|^2, \tag{9.5}$$

$$S_p(P) = \sum_i \sum_{(i,j) \in P_{k_2}(l_i)} \|P^T x_i - P^T x_j\|^2, \tag{9.6}$$

where S_c indicates the intraclass compactness, which is represented as the sum of distances between each point and its neighbors of the same class and $N_{k_1}^+(i)$ is the index set of the K_1 nearest neighbors of the sample x_i in the same class, S_p indicates the separability of different classes, which is characterized as the sum of distances between the marginal points and their neighboring points of different classes and $P_{k_2}(l)$ is a set of data pairs that are the k_2 nearest pairs among the set $(i,j), l_i = l, l_j \neq l$, and W is the weight matrix of the $\mathcal{L}_1$ graph. The optimal solution can be obtained via the generalized eigenvalue

decomposition method, and the derived projection matrix P is then used for dimensional reduction and consequent data classification.

9.2.3 Diffusion-based Learning

The algorithm introduced in last section 9.2.2 uses sparse graph as a penalty graph to learn a projection matrix P. With this projection matrix, both the labeled data and unlabeled data can be projected into a new space where intraclass data samples are close to each other and interclass data samples are far away. However, the most famous semi-supervised learning algorithms are based on label diffusion (propagation) over the constructed graph [180] [181] [182]. In this section, we introduce several popular and important graph diffusion algorithms and tools.

Zhou's Diffusions. Zhou et al. [180] propose the following diffusion method:

$$F = (I - \alpha S)^{-1} Y, \tag{9.7}$$

where I is the unit matrix, S is the graph Laplacian, and Y is the label matrix which $Y_{i,j} = 1$ if node i has label j and $Y_{i,j} = 0$ otherwise. F is the resulted labeled matrix.

Joachim's Diffusion. Joachims [181] proposes a new diffusion equation as:

$$F = (D_Y + S)^{-1} Y, \tag{9.8}$$

where D_Y is the a diagonal matrix with the row-sums of Y on the diagonal. S is the graph Laplacian.

ZGL's diffusion. Zhu et al. [182] proposed the following diffusion way to predict the label of j-th class.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} y^T S y \\ \text{subject to} & y_i = \begin{cases} 1 & \text{if node } i \text{ labeled in class } j \\ 0 & \text{if node } i \text{ labeled in another class} \\ \text{free} & \text{otherwise} \end{cases} \end{array} \tag{9.9}$$

Heat Diffusion. Heat diffusion is a diffusion process originate from physical science. It describes the way how heat flows from one place to other places. The definition of heat diffusion is based on the heat equation:

$$\frac{\partial H_t}{\partial t} = -\Delta_{\mathcal{M}} H_t, \tag{9.10}$$

where $\Delta_{\mathcal{M}}$ is the Laplace-Beltrami operator over Riemannian manifold$\mathcal{M}$, and $H_t = e^{-tS}$ is the heat kernel. $S = D^{-1}A$ where A is the adjacent matrix of graph which describe (approximate) the geometry of manifold $\mathcal{M}$. The above definitions of heat equation and heat kernel over manifold are same with definition over a general graph [183].

The heat kernel can be re-formulated as:

$$H_t(i,j) = \sum_{p=1} e^{-\lambda_p t}\phi_p(i)\phi_p(j), \tag{9.11}$$

where λ_p is the p-th eigenvalue and ϕ_p is the p-th eigenvector of Laplacian S. $H_t(i,j)$ calculates the amount of heat being transferred from i to j in time t given a unit heat source at i in the very beginning.

Heat Kernel Signature. Sun et al. [184] propose the heat kernel signature (HKS) with the following definition:

$$H_t(i) = H_t(i,i) = \sum_{p=1} e^{-\lambda_p t}\phi_p(i)^2. \tag{9.12}$$

The physical meaning of HKS is the amount of heat i keeps within itself at time t. The heat diffusion process states that heat tends to diffuse slower at point with sparse neighborhood and faster at point with denser neighbor hood. Therefore, HKS can intuitively depict the local density of each point (or graph node).

9.3 Future Research Directions

In long term, we expect to extend our sparse graph representation framework to have the characteristic of "scalability", and let it can handle very large-sized data in real-world applications. Three potential directions will be streaming algorithms [185], stretch [186] and hashing algorithms [187]. Improving the graph quality from the theory perspective is also an challenging research in data mining and machine learning research [188]. The sparse graph generated by our algorithms can be seen as the skeleton of data's structure, which means we still have space to improve the graph connectivity. The quality of them can be improved through edge connection manipulations following the requirements of specified mining or learning task as discussed in [7]. Around this, dense subgraph mining technique is the key to let us know "where" should we start the operation.

Bibliography

[1] Guangyao Zhou, Zhiwu Lu, and Yuxin Peng. L1-graph construction using structured sparsity. *Neurocomputing*, 120:441–452, 2013.

[2] Yuqiang Fang, Ruili Wang, Bin Dai, and Xindong Wu. Graph-based learning via auto-grouped sparse regularization and kernelized extension. *IEEE Transactions on Knowledge and Data Engineering,*, 27(1):142–154, 2015.

[3] Yingzhen Yang, Zhangyang Wang, Jianchao Yang, Jiangping Wang, Shiyu Chang, and Thomas S Huang. Data clustering by laplacian regularized l1-graph. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 3148–3149, 2014.

[4] Shuchu Han, Hao Huang, Hong Qin, and Dantong Yu. Locality-preserving l1-graph and its application in clustering. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 813–818. ACM, 2015.

[5] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.

[6] Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *KDD*, pages 104–112. ACM, 2013.

[7] David F Gleich and Michael W Mahoney. Using local spectral methods to robustify graph-based learning algorithms. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 359–368. ACM, 2015.

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

[9] Yu-Feng Li, Shao-Bo Wang, and Zhi-Hua Zhou. Graph quality judgement: A large margin expedition. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1725–1731, 2016. URL http://www.ijcai.org/Abstract/16/247.

[10] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[11] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[12] Paramveer S Dhillon, Partha Pratim Talukdar, and Koby Crammer. Inference driven metric learning (idml) for graph construction. 2010.

[13] Fei Wang and Changshui Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20 (1):55–67, 2008.

[14] Jun Wang, Tony Jebara, and Shih-Fu Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th international conference on Machine learning*, pages 1144–1151. ACM, 2008.

[15] Samuel I Daitch, Jonathan A Kelner, and Daniel A Spielman. Fitting a graph to vector data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 201–208. ACM, 2009.

[16] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

[17] Jean Cardinal, Sébastien Collette, and Stefan Langerman. Empty region graphs. *Computational geometry*, 42(3):183–195, 2009.

[18] Jerzy W Jaromczyk and Godfried T Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1517, 1992.

[19] K Ruben Gabriel and Robert R Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969.

[20] International Business Machines Corporation. Research Division, DG Kirkpatrick, and JD Radke. *A framework for computational morphology*. 1984.

[21] Prosenjit Bose, Sébastien Collette, Stefan Langerman, Anil Maheshwari, Pat Morin, and Michiel Smid. Sigma-local graphs. *Journal of discrete algorithms*, 8(1):15–23, 2010.

[22] Bin Cheng, Jianchao Yang, Shuicheng Yan, Yun Fu, and Thomas S Huang. Learning with l1-graph for image analysis. *Image Processing, IEEE Transactions on*, 19(4):858–866, 2010.

[23] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[24] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.

[25] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

[26] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.

[27] David L Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006.

[28] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[29] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

[30] Peng Zhao and Bin Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

[31] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[32] Yuqiang Fang, Ruili Wang, Bin Dai, and Xindong Wu. Graph-based learning via auto-grouped sparse regularization and kernelized extension. 2013.

[33] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[34] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[35] G Davis. Greedy adaptive approximation. *Journal of Constructive Approximation*, 13(1):57–98, 1997.

[36] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[37] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *IEEE journal of selected topics in signal processing*, 1(4):606–617, 2007.

[38] Michael R Osborne, Brett Presnell, and Berwin A Turlach. A new approach to variable selection in least squares problems. *IMA journal of numerical analysis*, 20(3):389–403, 2000.

[39] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4): 1168–1200, 2005.

[40] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[41] Junfeng Yang and Yin Zhang. Alternating direction algorithms for l1-problems in compressive sensing. *SIAM journal on scientific computing*, 33(1):250–278, 2011.

[42] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pages 849–856, 2001.

[43] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[44] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[45] Andrew V Goldberg. *Finding a maximum density subgraph*. University of California Berkeley, CA, 1984.

[46] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 121(1):15–26, 2002.

[47] Liansheng Zhuang, Haoyuan Gao, Zhouchen Lin, Yi Ma, Xin Zhang, and Nenghai Yu. Non-negative low rank and sparse graph for semi-supervised learning. In *CVPR*, pages 2328–2335. IEEE, 2012.

[48] P. Hall, B. U. Park BU, and R. J. Samworth. Choice of neighbor order in nearest-neighbor classification. *Annals of Statistics*, 36(5):2135–2152, 2008.

[49] Carlos D Correa and Peter Lindstrom. Locally-scaled spectral clustering using empty region graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1330–1338. ACM, 2012.

[50] H. Huang, H. Qin, S. Yoo, and D. Yu. Local anomaly descriptor: a robust unsupervised algorithm for anomaly detection based on diffusion space. *ACM CIKM*, pages 405–414, 2012.

[51] Shenghua Gao, IW-H. Tsang, and Liang-Tien Chia. Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. *PAMI*, 35 (1):92–104, 2013.

[52] Shuicheng Yan and Huan Wang. Semi-supervised learning by sparse representation. In *Society for Industrial and Applied Mathematics. Proceedings of the SIAM International Conference on Data Mining*, page 792. Society for Industrial and Applied Mathematics, 2009.

[53] Bin Dai, Xindong Wu, et al. Graph-based learning via auto-grouped sparse regularization and kernelized extension. *IEEE Transactions on Knowledge and Data Engineering*, page 1, 2014.

[54] Kwangmoo Koh, Seung-Jean Kim, and Stephen P Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8(8):1519–1555, 2007.

[55] A Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD cup and workshop*, pages 5–8. ACM, 2007.

[56] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2011.

[57] Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. *Advances in Neural Information Processing Systems*, 16:169–176, 2004.

[58] Bin Xu, Jiajun Bu, Chun Chen, Deng Cai, Xiaofei He, Wei Liu, and Jiebo Luo. Efficient manifold ranking for image retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 525–534. ACM, 2011.

[59] Yingzhen Yang, Zhangyang Wang, Jianchao Yang, Jiawei Han, and Thomas Huang. Regularized l1-graph for data clustering. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[60] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[61] Allen Y Yang, Shankar S Sastry, Arvind Ganesh, and Yi Ma. Fast 1-minimization algorithms and an application in robust face recognition: A review. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1849–1852. IEEE, 2010.

[62] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[63] MR Brito, EL Chavez, AJ Quiroz, and JE Yukich. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35(1):33–42, 1997.

[64] Michael Donoser and Horst Bischof. Diffusion processes for retrieval revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1320–1327. IEEE, 2013.

[65] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007.

[66] Chong You, D Robinson, and René Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, 2016.

[67] Alfred M Bruckstein, Michael Elad, and Michael Zibulevsky. On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. *IEEE Transactions on Information Theory,*, 54(11):4813–4820, 2008.

[68] Tsung-Han Lin and HT Kung. Stable and efficient representation learning with nonnegativity constraints. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1323–1331, 2014.

[69] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

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

[71] Shuchu Han, Hao Huang, Hong Qin, and Dantong Yu. Locality-preserving l1-graph and its application in clustering. 2014.

[72] Raj Rao Nadakuditi and Mark EJ Newman. Graph spectra and the detectability of community structure in networks. *Physical review letters*, 108(18):188701, 2012.

[73] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.

[74] Jonatan Taminau, David Steenhoff, Alain Coletta, Stijn Meganck, Cosmin Lazar, Virginie de Schaetzen, Robin Duque, Colin Molter, Hugues Bersini, Ann Nowé, et al. insilicodb: an r/bioconductor package for accessing human affymetrix expert-curated datasets from geo. *Bioinformatics*, 27(22):3204–3205, 2011.

[75] Helen Parkinson, Ugis Sarkans, Nikolay Kolesnikov, Niran Abeygunawardena, Tony Burdett, Miroslaw Dylag, Ibrahim Emam, Anna Farne, Emma Hastings, Ele Holloway, et al. Arrayexpress updatean archive of microarray and high-throughput sequencing-based functional genomics experiments. *Nucleic acids research*, 39(suppl 1):D1002–D1004, 2011.

[76] Daniel R Rhodes and Arul M Chinnaiyan. Integrative analysis of the cancer transcriptome. *Nature genetics*, 37:S31–S37, 2005.

[77] Shuangge Ma. Integrative analysis of cancer genomic data. 2009.

[78] Cosmin Lazar, Stijn Meganck, Jonatan Taminau, David Steenhoff, Alain Coletta, Colin Molter, David Y Weiss-Solís, Robin Duque, Hugues Bersini, and Ann Nowé. Batch effect removal methods for microarray gene expression data integration: a survey. *Briefings in bioinformatics*, 14(4):469–490, 2013.

[79] Andreas Scherer. *Batch effects and noise in microarray experiments: sources and solutions*, volume 868. John Wiley & Sons, 2009.

[80] J Luo, M Schumacher, A Scherer, D Sanoudou, D Megherbi, T Davison, T Shi, W Tong, L Shi, H Hong, et al. A comparison of batch effect removal methods for enhancement of prediction performance using maqc-ii microarray gene expression data. *The pharmacogenomics journal*, 10 (4):278–291, 2010.

[81] Cosmin Lazar, Jonatan Taminau, Stijn Meganck, David Steenhoff, Alain Coletta, David Y Weiss Solis, Colin Molter, Robin Duque, Hugues Bersini, and Ann Nowé. Geneshift: A nonparametric approach for integrating microarray gene expression data based on the inner product as a distance measure between the distributions of genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 10 (2):383–392, 2013.

[82] Earl Hubbell, Wei-Min Liu, and Rui Mei. Robust estimators for expression analysis. *Bioinformatics*, 18(12):1585–1592, 2002.

[83] Rafael A Irizarry, Bridget Hobbs, Francois Collin, Yasmin D Beazer-Barclay, Kristen J Antonellis, Uwe Scherf, and Terence P Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2003.

[84] Matthew N McCall, Benjamin M Bolstad, and Rafael A Irizarry. Frozen robust multiarray analysis (frma). *Biostatistics*, 11(2):242–253, 2010.

[85] Chao Chen, Kay Grennan, Judith Badner, Dandan Zhang, Elliot Gershon, Li Jin, and Chunyu Liu. Removing batch effects in analysis of expression microarray data: an evaluation of six batch adjustment methods. *PloS one*, 6(2):e17238, 2011.

[86] Andrew H Sims, Graeme J Smethurst, Yvonne Hey, Michal J Okoniewski, Stuart D Pepper, Anthony Howell, Crispin J Miller, and Robert B Clarke. The removal of multiplicative, systematic bias allows integration of breast cancer gene expression datasets–improving meta-analysis and prediction of prognosis. *BMC medical genomics*, 1(1):42, 2008.

[87] Cheng Li and Wing Hung Wong. Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. *Proceedings of the National Academy of Sciences*, 98(1):31–36, 2001.

[88] Ki-Yeol Kim, Se Hyun Kim, Dong Hyuk Ki, Jaeheon Jeong, Ha Jin Jeong, Hei-Cheul Jeung, Hyun Cheol Chung, and Sun Young Rha. An attempt for combining microarray data sets by adjusting gene expressions. *Cancer Research and Treatment*, 39(2):74–81, 2007.

[89] W Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007.

[90] Andrey A Shabalin, Håkon Tjelmeland, Cheng Fan, Charles M Perou, and Andrew B Nobel. Merging two gene-expression studies via cross-platform normalization. *Bioinformatics*, 24(9):1154–1160, 2008.

[91] Monica Benito, Joel Parker, Quan Du, Junyuan Wu, Dong Xiang, Charles M Perou, and James Stephen Marron. Adjustment of systematic microarray data biases. *Bioinformatics*, 20(1):105–114, 2004.

[92] Orly Alter, Patrick O Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, 2000.

[93] Jeffrey T Leek and John D Storey. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS genetics*, 3(9): e161, 2007.

[94] Johann A Gagnon-Bartsch and Terence P Speed. Using control genes to correct for unwanted variation in microarray data. *Biostatistics*, 13(3): 539–552, 2012.

[95] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, pages 1065–1076, 1962.

[96] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. Springer, 2010.

[97] Björn H Junker and Falk Schreiber. *Analysis of biological networks*, volume 2. John Wiley & Sons, 2008.

[98] Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21(suppl 1):i213–i221, 2005.

[99] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, et al. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research*, 31(9):2443–2450, 2003.

[100] Charalampos E Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. 2013.

[101] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[102] Yi-Kuei Lin. Reliability evaluation for an information network with node failure under cost constraint. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 37(2):180–188, 2007.

[103] Ernesto Estrada, Naomichi Hatano, and Michele Benzi. The physics of communicability in complex networks. *CoRR*, abs/1109.2950, 2011.

[104] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2):203–221, 2000.

[105] A. V. Goldberg. Finding a maximum density subgraph. Technical Report CSD-84-171, UC Berkeley, 1984.

[106] Jeffrey Pattillo, Alexander Veremyev, Sergiy Butenko, and Vladimir Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1-2):244–257, 2013.

[107] Thomas A. Feo and Mauricio G.C. Resende. Greedy randomized adaptive search procedures. *J. of Optimization*, 6:109–133, 1995.

[108] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

[109] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network Robustness and Fragility: Percolation on Random Graphs. *Phys. Rev. Let.*, 85(25):5468–5471, 2000.

[110] Hau Chan, Leman Akoglu, and Hanghang Tong. Make It or Break It: Manipulating robustness in large networks. In *SDM*, 2014.

[111] Reuven Cohen, Keren Erez, Daniel B. Avraham, and Shlomo Havlin. Breakdown of the Internet under Intentional Attack. *Physical Review Letters*, 86(16):3682–3685, April 2001. doi: 10.1103/physrevlett.86.3682.

[112] E. Estrada. Network robustness to targeted attacks: The interplay of expansibility and degree distribution. *The Euro. Phys. J. B*, 52(4):563–574, 2006.

[113] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Phy. R. E*, 65(5), 2002.

[114] Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, pages 245–254, 2012.

[115] H. Frank and I. Frisch. Analysis and Design of Survivable Networks. *IEEE Trans. on Comm. Tech.*, 18(5), 1970.

[116] Benjamin Shargel, Hiroki Sayama, Irving R Epstein, and Yaneer Bar-Yam. Optimization of robustness and connectivity in complex networks. *Phys Rev Lett*, 90(6):068701, 2003.

[117] G. Paul, T. Tanizawa, S. Havlin, and H. Stanley. Optimization of robustness of complex networks. *The Eur. Phys. J. B*, 38(2):187–191, 2004.

[118] Walid K. Ghamry and Khaled M. F. Elsayed. Network design methods for mitigation of intentional attacks in scale-free networks. *Telecom. Systems*, 49(3):313–327, 2012.

[119] Stojan Trajanovski, Fernando A. Kuipers, and Piet Van Mieghem. Finding critical regions in a network. In *INFOCOM*, 2013.

[120] Yilin Shen, Nam P. Nguyen, Ying Xuan, and My T. Thai. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Trans. Netw.*, 21(3), 2013.

[121] Vito Latora and Massimo Marchiori. Vulnerability and protection of infrastructure networks. *Phys. Rev. E*, 71:015103, 2005.

[122] Marco Di Summa, Andrea Grosso, and Marco Locatelli. Branch and cut algorithms for detecting critical nodes in undirected graphs. *Comp. Opt. and Appl.*, 53(3):649–680, 2012.

[123] Takeshi Fujimura and Hiroyoshi Miwa. Critical links detection to maintain small diameter against link failures. In *INCoS*, pages 339–343. IEEE, 2010.

[124] Reid Andersen and Sebastian M. Cioaba. Spectral densest subgraph and independence number of a graph. *J. UCS*, 13(11):1501–1513, 2007.

[125] M. Garey and D. Johnson. *Computers and Intractability - A guide to the Theory of NP-Completeness*. Freeman, 1979.

[126] Johan Hastad. Clique is hard to approximate within $n^{(1-\epsilon)}$. In *FOCS*, pages 627–636. IEEE Computer Society, 1996.

[127] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Disc. Appl. Math.*, 121(1-3):15–26, 2002.

[128] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *KDD*, pages 228–238, 2005.

[129] Victor E. Lee, Ning Ruan, Ruoming Jin, and Charu C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Springer, 2010.

[130] Wendy Ellens and Robert E. Kooij. Graph measures and network robustness. *CoRR*, abs/1311.5064, 2013.

[131] Jun Wu, Barahona Mauricio, Yue-Jin Tan, and Hong-Zhong Deng. Natural connectivity of complex networks. *Chinese Physics Letters*, 27(7): 78902, 2010. doi: 10.1088/0256-307X/27/7/078902.

[132] Ernesto Estrada. Characterization of the folding degree of proteins. *Bioinformatics*, 18(5):697–704, 2002.

[133] G. W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[134] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, 1999.

[135] Charalampos E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *ICDM*, pages 608–617. IEEE Computer Society, 2008.

[136] Marcelo Prais and Celso C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS*, 12(3):164–176, 2000.

[137] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. *J. Comp. Sys. Sci.*, 37(1), 1988.

[138] Xiaofei He, Ming Ji, Chiyuan Zhang, and Hujun Bao. A variance minimization criterion to feature selection using laplacian regularization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33 (10):2013–2025, 2011.

[139] Chenping Hou, Feiping Nie, Xuelong Li, Dongyun Yi, and Yi Wu. Joint embedding learning and sparse regression: A framework for unsupervised feature selection. *Cybernetics, IEEE Transactions on*, 44(6):793–804, 2014.

[140] Xinwang Liu, Lei Wang, Jian Zhang, Jianping Yin, and Huan Liu. Global and local structure preservation for feature selection. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(6):1083–1095, 2014.

[141] Zheng Zhao, Lei Wang, Huan Liu, and Jieping Ye. On similarity preserving feature selection. *Knowledge and Data Engineering, IEEE Transactions on*, 25(3):619–632, 2013.

[142] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. l2, 1-norm regularized discriminative feature selection for unsupervised learning. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1589. Citeseer, 2011.

[143] Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, and Hanqing Lu. Unsupervised feature selection using nonnegative spectral analysis. In *AAAI*, 2012.

[144] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[145] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[146] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434. Springer, 2001.

[147] Liang Du and Yi-Dong Shen. Unsupervised feature selection with adaptive structure learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 209–218. ACM, 2015.

[148] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[149] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

[150] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.

[151] Jennifer G Dy and Carla E Brodley. Feature selection for unsupervised learning. *The Journal of Machine Learning Research*, 5:845–889, 2004.

[152] Deng Cai, Chiyuan Zhang, and Xiaofei He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 333–342. ACM, 2010.

[153] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research*, 5:1205–1224, 2004.

[154] D KOLLER. Toward optimal feature selection. In *Proc. 13th International Conference on Machine Learning*, pages 284–292. Morgan Kaufmann, 1996.

[155] Zheng Zhao, Lei Wang, Huan Liu, et al. Efficient spectral feature selection with minimum redundancy. In *AAAI*, 2010.

[156] Xuerui Wang, Andrew McCallum, and Xing Wei. Feature selection with integrated relevance and redundancy optimization. In *Data Mining, 2015. ICDM 2015. Fifteenth IEEE International Conference on*, pages 697–702. IEEE, 2015.

[157] De Wang, Feiping Nie, and Heng Huang. Feature selection via global redundancy minimization. *Knowledge and Data Engineering, IEEE Transactions on*, 27(10):2743–2755, 2015.

[158] Julien Mairal and Bin Yu. Supervised feature selection in graphs with path coding penalties and network flows. *The Journal of Machine Learning Research*, 14(1):2449–2485, 2013.

[159] Qinbao Song, Jingjie Ni, and Guangtao Wang. A fast clustering-based feature subset selection algorithm for high-dimensional data. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):1–14, 2013.

[160] Mingxia Liu, Dan Sun, and Daoqiang Zhang. Sparsity score: A new filter feature selection method based on graph. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 959–962. IEEE, 2012.

[161] J. Li, K. Cheng, S. Wang, F. Morstatter, R. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. 2016. URL http://featureselection.asu.edu/.

[162] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[163] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, 2013.

[164] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12, 2014.

[165] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185, 2014.

[166] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[167] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[168] https://www.census.gov/topics/population/genealogy/data/1990_census/1990_census_namefiles.html, .

[169] https://www.census.gov/topics/population/genealogy/data/2000_surnames.html, .

[170] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80. ACM, 2005.

[171] David L Word, Charles D Coleman, Robert Nunziata, and Robert Kominski. Demographic aspects of surnames from census 2000. *Unpublished manuscript, Retrieved from http://citeseerx. ist. psu. edu/viewdoc/download*, 2008.

[172] https://code.google.com/archive/p/word2vec/.

[173] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[174] Yifan Hu, Emden Gansner, and Stephen Kobourov. Visualizing graphs and clusters as maps. *IEEE Computer Graphics and Applications*, 30: 54–66, 2010.

[175] http://www.behindthename.com.

[176] Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and J Niels Rosenquist. Understanding the demographics of twitter users. *ICWSM*, 11:5th, 2011.

[177] Xiaofei He, Deng Cai, Shuicheng Yan, and Hong-Jiang Zhang. Neighborhood preserving embedding. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1208–1213. IEEE, 2005.

[178] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: a general framework for dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):40–51, 2007.

[179] Deng Cai, Xiaofei He, and Jiawei Han. Semi-supervised discriminant analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE, 2007.

[180] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[181] Thorsten Joachims et al. Transductive learning via spectral graph partitioning. In *ICML*, volume 3, pages 290–297, 2003.

[182] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

[183] Fan RK Chung. *Spectral graph theory*, volume 92. Amer Mathematical Society, 1997.

[184] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[185] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.

[186] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.

[187] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[188] Yining Wang, Yu-Xiang Wang, and Aarti Singh. Graph connectivity in noisy sparse subspace clustering. *CoRR abs/1504.01046*, 2016.