2021

HIGH PERFORMANCE SPECTRAL METHODS FOR GRAPH-BASED MACHINE LEARNING

Yongyu Wang
Michigan Technological University, yongyuw@mtu.edu

HIGH PERFORMANCE SPECTRAL METHODS FOR GRAPH-BASED MACHINE LEARNING

By

Yongyu Wang

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computer Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2021

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computer Engineering.

Department of Electrical And Computer Engineering

Dissertation Co-advisor:	*Dr. Chee-Wooi Ten*
Dissertation Co-advisor:	*Dr. Zhuo Feng*
Committee Member:	*Dr. Zhenlin Wang*
Committee Member:	*Dr. Laura E. Brown*
Department Chair:	*Dr. Glen E. Archer*

Contents

List of Figures

List of Tables

Preface

The material described in Chapter 4 has been submitted for review and publication.

Abstract

Graphs play a critical role in machine learning and data mining fields. The success of graph-based machine learning algorithms highly depends on the quality of the underlying graphs. Desired graphs should have two characteristics: 1) they should be able to well-capture the underlying structures of the data sets. 2) they should be sparse enough so that the downstream algorithms can be performed efficiently on them.

This dissertation first studies the application of a two-phase spectrum-preserving spectral sparsification method that enables to construct very sparse sparsifiers with guaranteed preservation of original graph spectra for spectral clustering. Experiments show that the computational challenge due to the eigen-decomposition procedure in spectral clustering can be fundamentally addressed.

We then propose a highly-scalable spectral graph learning approach GRASPEL (Graph Spectral Learning at Scale). GRASPEL can learn high-quality graphs from high dimensional input data. Compared with prior state-of-the-art graph learning and construction methods [26, 27, 38] , GRASPEL leads to substantially improved algorithm performance.

Chapter 1

Introduction

1.1 Background

Graph-based methods play an important role in many data mining and machine learning tasks, such as data representation and analysis, dimensionality reduction, and clustering. A key step of graph-based methods requires representing relationship between data points by graphs: it is a common practice to represent each data point as a node, and use weighted edges to represent relations between nodes. The constructed graph can be used to represent the underlying structure (manifold) of the data set.

1.2 Challenges

The graph is at the heart of graph-based methods. It has significantly effect on the performance of the methods [25, 32, 34]. However, how to construct a meaningful graph from the data set remains a challenging problem. Desired graph construction algorithms should allow good capturing and understanding of the global structure (manifold) of the data set while producing sufficiently sparse graphs that can be easily stored and efficiently manipulated in the downstream algorithms. Additionally, the graph construction process should be efficient.

1.3 Clustering

Clustering is one of the most fundamental tasks in machine learning and data mining fields. It aims to assign data samples in a data set into different clusters in such a way that samples in the same cluster are more similar compared to those in different clusters, as shown in Figure 1.1. It has been widely used in many contexts such as content analysis, information retrieval, web analytics, image segmentation, and computational biology [7, 58].

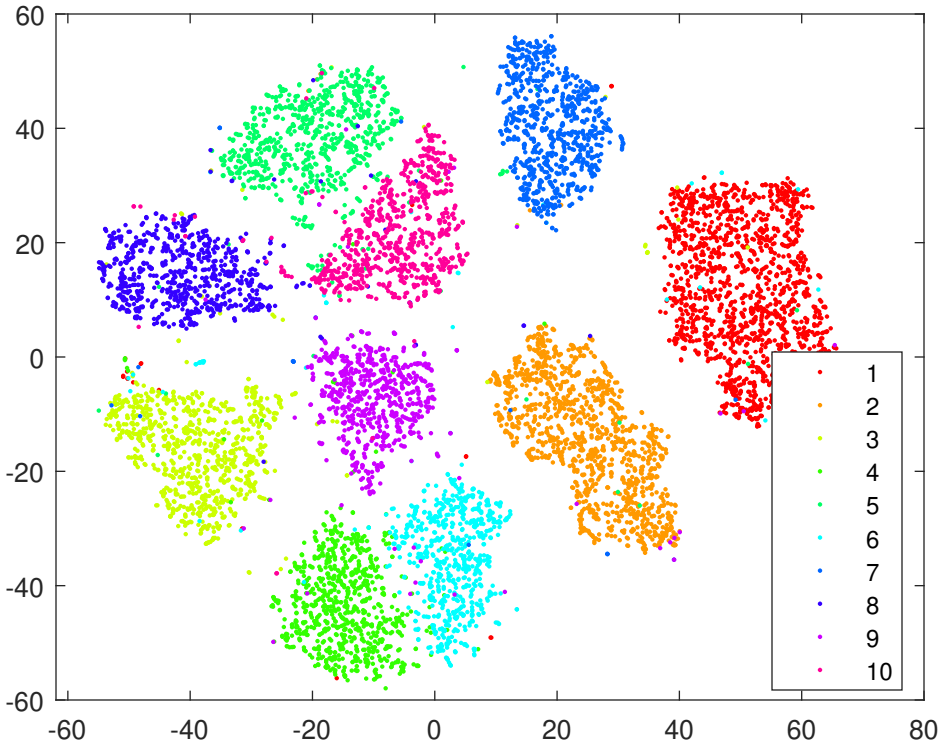


Figure 1.1: Visualization of clusters in handwritten digits USPS data set.

1.3.1 k-means

k-means is the most widely used clustering method [8]. The goal of k-means is to find a cluster membership assignment that minimizes the following objective function:

$$F = \sum_{i=1}^{n} \sum_{j=1}^{k} \eta_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|_2, \tag{1.1}$$

where:

$$\eta_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \in C_j \\ 0 & \text{otherwise ,} \end{cases} \tag{1.2}$$

and $\boldsymbol{\mu}_j$ is the centroid of the cluster C_j.

The algorithm is shown in Algorithm 1.

Algorithm 1 k-means

Input: Data samples $x_1, ..., x_n \in \mathbb{R}^d$, number of clusters k.
Output: Clusters C_1,...,C_k.

1: Randomly choose k samples as the initial centroids ;
2: **while** convergence criterion not met **do**
3: Form k clusters by assigning each sample to the nearest centroid;
4: Update centroids by calculating the mean of each cluster ;
5: **end while**

1.3.2 Graph Clustering

Due to the effectiveness of graph structure for representing and exploiting intrinsic data characteristics, graph clustering has gained great attention in recent years. It is a task to discover clusters in a graph, where each cluster is well-connected internally and with sparse connections between clusters, as shown in Figure 1.2

Graph clustering has superior performance compared to traditional clustering methods [42]. However, to perform graph clustering on a data set, a graph is required to be learned (constructed) from the data set.

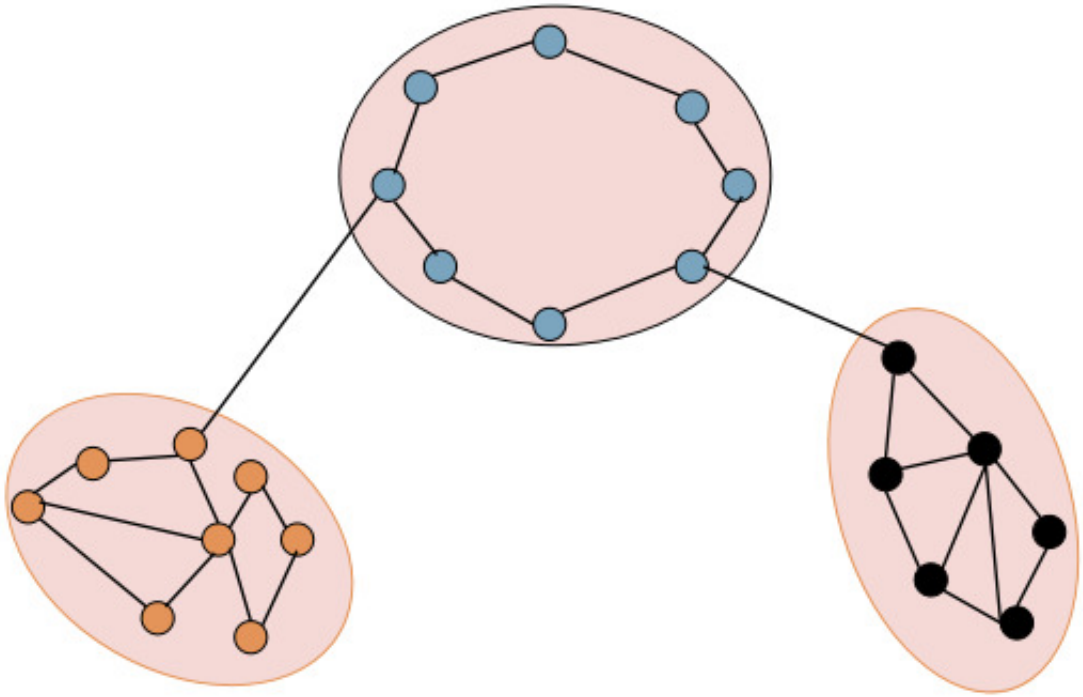


Figure 1.2: Graph clustering.

1.3.3 Evaluation Metrics

We evaluate the clustering quality by comparing the labels of each sample obtained by performing clustering algorithms with the ground-truth labels provided by the data set. Two metrics are commonly used: the clustering accuracy (ACC) metric, and the normalized mutual information (NMI) metric.

1.3.3.1 Clustering Accuracy (ACC)

The clustering accuracy (ACC) [9, 10, 31] is defined as follows:

$$ACC = \frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n}, \tag{1.3}$$

where n is the number of data samples in the data set, y_i is the ground-truth label provided by the data set, and c_i is the label generated by the clustering algorithm. $\delta(x, y)$ is a delta function defined as: $\delta(x, y)$=1 for $x = y$, and $\delta(x, y)$=0, otherwise. $map(\bullet)$ is a permutation function that maps each cluster index c_i to a ground truth label, which can be realized using the Hungarian algorithm [37]. A higher value of ACC indicates a better clustering accuracy.

1.3.3.2 Normalized Mutual Information (NMI)

For two random variables P and Q, normalized mutual information is defined as [49]:

$$NMI = \frac{I(P, Q)}{\sqrt{H(P)H(Q)}}, \tag{1.4}$$

where $I(P, Q)$ denotes the mutual information between P and Q, while $H(P)$ and $H(Q)$ are entropies of P and Q. In practice, the NMI metric can be calculated as

follows [49]:

$$NMI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} n_{i,j}\log(\frac{n \cdot n_{i,j}}{n_i \cdot n_j})}{\sqrt{(\sum_{i=1}^{k} n_i \log\frac{n_i}{n})(\sum_{j=1}^{k} n_j \log\frac{n_j}{n})}}, \tag{1.5}$$

where n is the number of data samples in the data set, k is the number of clusters, n_i is the number of data samples in cluster C_i according to the clustering result generated by the algorithm, n_j is the number of data samples in cluster C_j according to the ground-truth labels provided by the data set, and $n_{i,j}$ is the number of data samples in cluster C_i according to the clustering result as well as in class C_j according to the ground-truth labels. The NMI value is in the range of [0, 1], while a higher NMI value indicates a better matching between the algorithm generated result and the ground-truth result.

1.4 Contributions

The **first contribution** of this dissertation is a study of the use of a spectrum-preserving spectral graph sparsification method [16] for spectral graph clustering. We use it for sparsifying large nearest neighbor graphs that reduce the cost of eigen-decomposition of the Laplacian matrices. The experiments conducted in this work demonstrate the effectiveness of the spectrally-critical off-tree edges used in spectral

sparsification tool on spectral clustering, as they can improve the clustering performance. A new scheme for spectral stability checking is proposed. It can be used as the termination criterion in spectral graph sparsification and spectral graph learning methods.

The **second contribution** of this dissertation is GRASPEL (Graph Spectral Learning at Scale), a spectral method for graph learning from data. We show the clear connection between our method and the GSP-based Laplacian estimation methods. The proposed method includes a novel critical edge identification method and a new edge sampling method based on spectral embedding distortion. We show through extensive experiments that our approach can learn high-quality graphs without sacrificing the sparsity: the learned graphs can be immediately leveraged to significantly improve the efficiency and accuracy of spectral clustering tasks.

Chapter 2

Related Work

In this chapter, we review existing graph construction (learning) methods, basic spectral graph theory concepts, and spectral clustering algorithms. In section 2.1, we review the traditional and state-of-the-art graph construction methods. In section 2.2, we review the graph Laplacian matrix. In section 2.3, we review the graph sparsification technique. Then, we review the derivation of spectral clustering and provide some observations in section 2.4.

2.1 Graph Construction Methods

Graph construction is an important step in graph clustering and many other machine learning tasks such as graph-based semi-supervised learning. In this section, a brief review of several popular graph construction methods is provided.

2.1.1 k-Nearest Neighbor Graph

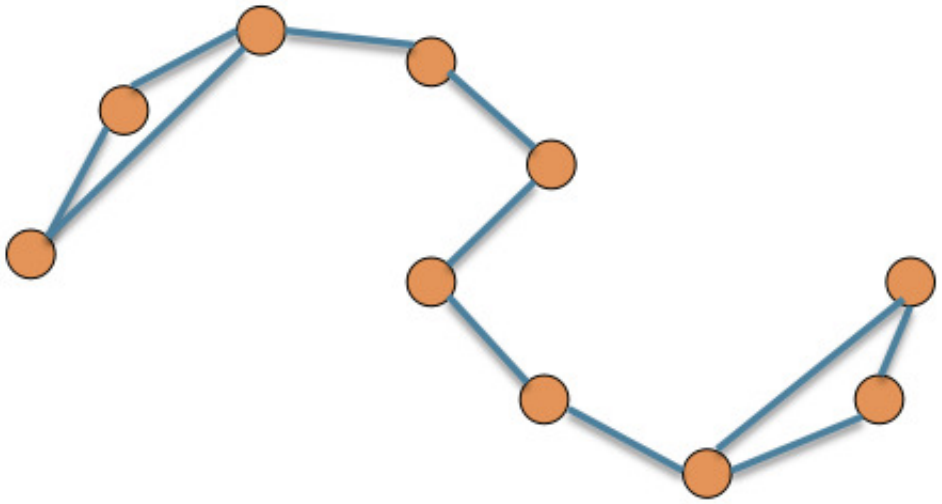


Figure 2.1: The 2-NN graph

The k-nearest neighbor (k-NN) graph is commonly used in practice. As shown in Figure 2.1, each node is connected to its k nearest neighbors. The algorithm is shown in Algorithm 2.

The k-NN graph can capture the local manifold structure. It also has good robustness

Algorithm 2 k-NN graph construction

Input: Data samples $x_1, ..., x_n \in \mathbb{R}^d$, neighborhood size k.
Output: Graph G.

1: **for** each data sample x_i **do**
2: Compute distances between x_i and all the other data samples;
3: Sort the computed distances;
4: Connect x_i with its k nearest data samples;
5: **end for**

to outliers [39]. However, the k-NN graph has the following drawbacks: 1) The optimal k value is usually problem-dependent and can be very difficult to find. 2) A k-NN graph has the tendency to include noisy edges [57], which are the edges representing false neighborhood relationships . 3) The fixed-size neighborhood limits the capability of the k-NN graph in representing the manifold structure [38].

2.1.2 ϵ-Neighborhood Graph

In ϵ-neighborhood graph, each node is connected to all the neighbors within a range of distance ϵ. But the proper ϵ value can be very difficult to find or does not exist when different clusters have different radii. Compared to the k-NN graph, ϵ-neighborhood graph is much less often used.

2.1.3 Consensus Method

The consensus of a pair of node is measured by the times that they appear together in other nodes' neighborhoods. The consensus method aims to construct the graph via the "true" neighborhood [38]. The consensus information is extracted from a provided k-NN graph for pruning noisy edges and the edges with a consensus value less than a threshold are dropped. However, it relies on the consensus information only. Some useful structural information can be discarded. The algorithm is shown in Algorithm 3.

Algorithm 3 Consensus Method

Input: Data samples $x_1, ..., x_n \in \mathbb{R}^d$, neighborhood size k, threshold τ.
Output: Graph G.

```
1: Construct a k-NN graph;
2: Extract the consensus matrix C from the k-NN graph;
3: for Each edge(p, q) in the k-NN graph do
4:    if C(p,q)<τ then
5:       Remove edge(p, q) from the graph
6:    end if
7: end for
```

2.1.4 Graph Learning Methods

Several recent graph learning methods leverage emerging graph signal processing (GSP) techniques for estimating sparse graph Laplacians, which show very promising results [13, 14, 15, 26]. For example, [15] addresses the graph learning problem by

restricting the precision matrix to be a graph Laplacian and maximizing a posterior estimation of Gaussian Markov Random Field (GMRF), while an $l1$-regularization term is used to promote graph sparsity; [40] provides error analysis for inferring sparse graphs from smooth signals; [27] leverages an approximate nearest-neighbor (ANN) technique to reduce the number of variables for optimization. However, the standard Laplacian estimation methods for graph learning require at least $O(n^2)$ time in each iteration, restricting their application to large-scale data sets.

The GSP-based methods infer the graph by adopting the criterion of signal smoothness [13]. Recent methods enforce the sparsity and smoothness of signals by formulating the task as the following optimization problem :

$$\min_{W \in \mathcal{W}} : Tr(WZ) + f(W), \tag{2.1}$$

where $\mathcal{W}$ denotes the set of valid adjacency matrices and Z denotes the pairwise distance matrix. It is obvious that the first term enforces the smoothness of the observed signals on the learned graph, while the second term is for imposing sparsity of the learned graph. In recent years, different forms of $f(W)$ have been proposed. For example, the following form has been introduced in [26]:

$$f(W) = -\alpha \mathbf{1}^T \log(W\mathbf{1}) + \frac{1}{2}\beta \|W\|_F^2, \tag{2.2}$$

where the first term is used to prevent the corresponding adjacency matrix W from obtaining the trivial solution $W = 0$, and the second term is for controlling the sparsity of the learned graph.

2.1.5 Summary

Typically graph construction is based on either neighborhood connection, or on graph learning through solving optimization problems [13, 26]. A few representative methods for graph construction is reviewed in this section.

2.2 Graph Laplacian Matrix

Consider a graph $G = (V, E, w)$, where V and E denote the vertex set and edge set, respectively, while w denotes the weight (similarity) function that assigns positive weights to all edges.

The adjacency matrix of graph G is defined as:

$$A_G(p,q) = \begin{cases} w(p,q) & \text{if } (p,q) \in E \\ 0 & \text{otherwise .} \end{cases} \tag{2.3}$$

The degree of a vertex i in graph G is defined as:

$$deg(i) = \sum_{(j \neq i) \in V} w(i,j). \tag{2.4}$$

The degree matrix of graph G is defined as:

$$D_G(p,q) = \begin{cases} deg(p) & \text{if } (p = q) \\ 0 & \text{otherwise .} \end{cases} \tag{2.5}$$

The unnormalized Laplacian matrix of graph G is defined as:

$$L_G = D_G - A_G. \tag{2.6}$$

The elements of L_G are given by:

$$L_G(p,q) = \begin{cases} -w(p,q) & \text{if } (p,q) \in E \\ \sum\limits_{(p,t)\in E} w(p,t) & \text{if } (p=q) \\ 0 & \text{otherwise .} \end{cases} \tag{2.7}$$

The graph Laplacian matrix plays a central role in spectral graph theory. The eigenvalues and eigenvectors of the matrix are useful for studying the properties and structure of a graph [12].

2.3 Spectral Graph Sparsification

A σ-spectrally similar sparisifer P of the original graph G should be able to hold the following condition for all real vectors $\mathbf{x} \in \mathbb{R}^V$ by using the the same set of vertices of G, but much fewer edges:

$$\frac{\mathbf{x}^\top \mathbf{L}_P \mathbf{x}}{\sigma} \leq \mathbf{x}^\top \mathbf{L}_G \mathbf{x} \leq \sigma \mathbf{x}^\top \mathbf{L}_P \mathbf{x}. \tag{2.8}$$

The relative condition number is defined as:

$$\kappa(\mathbf{L}_G, \mathbf{L}_P) = \frac{\lambda_{max}(\mathbf{L}_P^+ \mathbf{L}_G)}{\lambda_{min}(\mathbf{L}_P^+ \mathbf{L}_G)}, \tag{2.9}$$

where $\mathbf{L}_P^+$ denotes the Moore-Penrose pseudoinverse of $\mathbf{L}_P$.

It can be shown that $\kappa(\mathbf{L}_G, \mathbf{L}_P) = \sigma^2$ [16]. The goal of spectral graph sparsification is to find a sparsifier P that leads to a small relative condition number.

2.4 Spectral Clustering

Spectral clustering uses the spectrum of the graph Laplacian matrix to perform spectral embedding and divide data samples into different clusters. The methods proposed in this dissertation can significantly improve the performance of spectral clustering.

2.4.1 Background

Consider a problem of partitioning the nodes of graph G into two disjoint sets, C_1, C_2. The cut of the partition is defined as:

$$Cut(C_1, C_2) = \sum_{p \in C_1, q \in C_2} w(p, q). \tag{2.10}$$

If the goal of clustering is set to find a partition that minimizes the cut, it has the tendency of cutting small sets of isolated nodes in the graph, as shown in Figure 2.2.

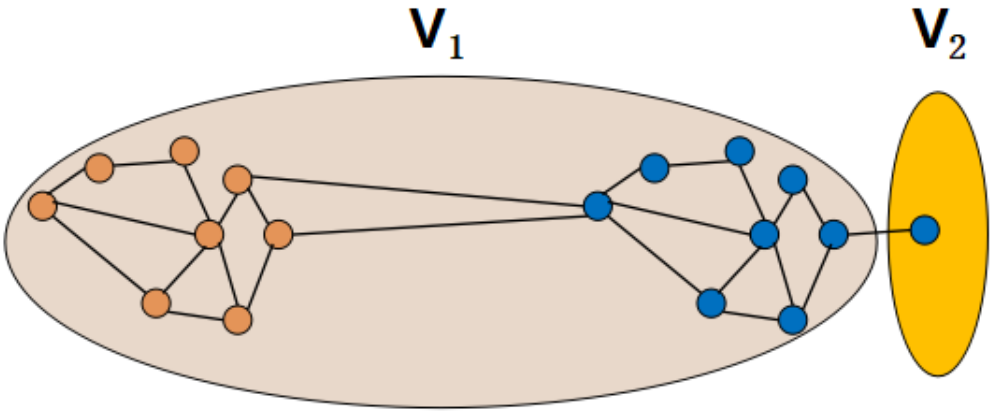


Figure 2.2: Clustering result without the constraint on cluster size.

To avoid this problem, sizes of clusters should be "reasonably large" [52]. To this end, RatioCut [23] and normalized cut Ncut [43] have been proposed.

Consider a more general problem of partitioning the nodes of graph G into k clusters C_1, C_2,...,C_k, the RatioCut is defined as:

$$RatioCut(C_1, ..., C_k) = \sum_{i=1}^{k} \frac{cut(C_i, \overline{C_i})}{|C_i|}, \tag{2.11}$$

where $|C_i|$ denotes the number of samples in cluster C_i.

And the normalized cut Ncut is defined as:

$$Ncut(C_1, ..., C_k) = \sum_{i=1}^{k} \frac{cut(C_i, \overline{C_i})}{vol(C_i)}, \tag{2.12}$$

where

$$vol(C_i) = \sum_{p \in C_i, q \in V} w(p, q). \tag{2.13}$$

Then, the objective of graph clustering can be set to find a cluster membership that minimizes the RatioCut or the Ncut:

$$\min_{C_1,...,C_k} \; RatioCut(C_1, ..., C_k)$$
$$s.t. \quad \begin{cases} C_1 \cup ... \cup C_k = V; \\ C_1 \cap ... \cap C_k = \emptyset. \end{cases} \tag{2.14}$$

$$\min_{C_1,...,C_k} \; Ncut(C_1, ..., C_k)$$
$$s.t. \quad \begin{cases} C_1 \cup ... \cup C_k = V; \\ C_1 \cap ... \cap C_k = \emptyset. \end{cases} \tag{2.15}$$

However, both optimization problems are NP-complete[53].

2.4.2 Approximation Algorithms

By taking advantage of the spectral graph theory, approximate solutions of (2.14) and (2.15) can be found.

Consider a graph $G = (V, E, w)$, where $|V| = n$. The number of clusters is k. A matrix $\boldsymbol{M} \in \mathbb{R}^{n \times k}$ is defined as:

$$\mathbf{M}(i,j) = \begin{cases} \frac{\mathbf{1}}{\sqrt{|C_j|}} & \text{if } V(i) \in C_j \\ 0 & \text{otherwise .} \end{cases} \tag{2.16}$$

It has been shown that [52]:

$$RatioCut(C_1, ..., C_k) = \mathbf{Tr}(\mathbf{M}^\top \mathbf{L}_G \mathbf{M}). \tag{2.17}$$

Thus we can solve (2.14) by solving the trace-minimization problem. By relaxing the entries of the matrix M to take arbitrary real values, the Rayleigh-Ritz theorem shows that the optimal solution is given by choosing M as the matrix which contains the eigenvectors corresponding to the k smallest eigenvalues of $\mathbf{L}_G$ as columns.

Due to the relaxation, each row in the matrix M no longer has only one non-zero entry, so we cannot obtain cluster membership of each node from M directly. To solve this problem, k-means algorithm is used to partition the n rows of M into k clusters. The cluster membership of the i-th row is assigned to the i-th node. This derivation leads to the unnormalized spectral clustering algorithm (as shown in Algorithm 4).

Algorithm 4 Unnormalized Spectral Clustering

Input: Data samples $x_1, ..., x_n \in \mathbb{R}^d$, number of clusters k.
Output: Clusters C_1,...,C_k.

1: Construct a graph G from the input data;
2: Compute the adjacency matrix A_G and diagonal matrix D_G of graph G ;
3: Obtain the unnormalized Laplacian matrix L_G=D_G-A_G;
4: Compute the eigenvectors u_1,...,u_k that correspond to the bottom k nonzero eigenvalues of L_G;
5: Construct $U \in \mathbb{R}^{n \times k}$, with the k eigenvectors stored as column vectors;
6: Perform k-means algorithm to partition the rows of U into k clusters and return the result.

Similarly, based on the normalized cut Ncut, a normalized spectral clustering can be derived (as shown in Algorithm 5).

Algorithm 5 Normalized Spectral Clustering (Ncut)

Input: Data samples $x_1, ..., x_n \in \mathbb{R}^d$, number of clusters k.
Output: Clusters C_1,...,C_k.

1: Construct a graph G from the input data;
2: Compute the adjacency matrix A_G and diagonal matrix D_G of graph G ;
3: Obtain the unnormalized Laplacian matrix L_G=D_G-A_G;
4: Compute the normalized Laplacian matrix $L_{Grw} = D_G^{-1} L_G$;
5: Compute the eigenvectors u_1,...,u_k that correspond to the bottom k nonzero eigenvalues of L_{Grw};
6: Construct $U \in \mathbb{R}^{n \times k}$, with the k eigenvectors stored as column vectors;
7: Perform k-means algorithm to partition the rows of U into k clusters and return the result.

The Laplacian matrix used in Algorithm 5 is denoted as L_{Grw} because it has close

relation to random walk.

Using the eigenvectors in a slight different way, another normalized spectral clustering algorithm has been proposed [35]. It is shown in Algorithm 6.

Algorithm 6 Normalized Spectral Clustering Proposed by Ng, Jordan, and Weiss

Input: Data samples $x_1, ..., x_n \in \mathbb{R}^d$, number of clusters k.
Output: Clusters C_1,...,C_k.

1: Construct a graph G from the input data;
2: Compute the adjacency matrix A_G and diagonal matrix D_G of graph G ;
3: Obtain the unnormalized Laplacian matrix L_G=D_G-A_G;
4: Compute the normalized Laplacian matrix $L_{Gsym} = D_G^{-1/2} L_G D_G^{-1/2}$;
5: Compute the eigenvectors u_1,...,u_k that correspond to the bottom k nonzero eigenvalues of L_{Gsym};
6: Construct $U \in \mathbb{R}^{n \times k}$, with the k eigenvectors stored as column vectors;
7: Normalize the rows in U to norm **1**
8: Perform k-means algorithm to partition the rows of U into k clusters and return the result.

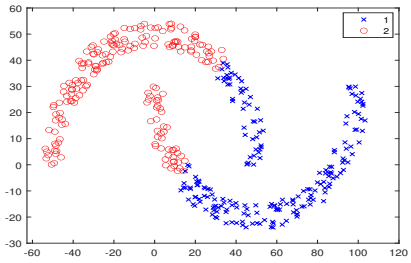
The Laplacian matrix used in Algorithm 6 is denoted as L_{Gsym} because it is a symmetric matrix.

2.4.3 Observations

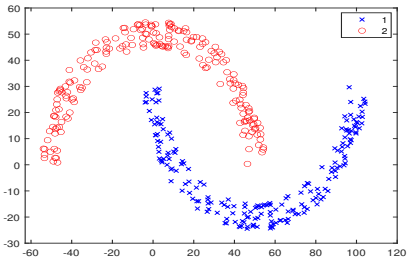
Spectral clustering has a strong ability in detecting non-convex and linearly non-separable patterns [35]. It is often more effective than traditional clustering algorithms such as k-means. This section aims to illustrate the important role that the spectrum plays in helping spectral clustering algorithm achieve its goal of detecting clusters from the data set. We also illustrate the computational complexity of the

spectral clustering approaches.

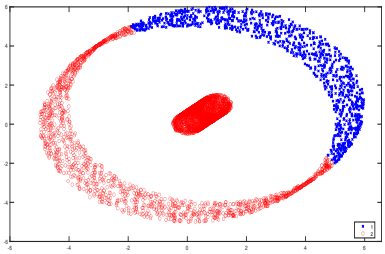
k-means often fails to handle complicated geometric shapes, especially the non-convex shapes. For example, the two moons and two circles are well-known synthetic data sets. In the two moons data set, there are two slightly entangled non-convex shapes, where each cluster corresponds to a moon. In the two circles data set, the samples are arranged in two concentric circles, where each cluster corresponds to a circle. k-means gives incorrect clustering results for both data sets while spectral clustering performs an ideal clustering, as shown in Figure 2.3.



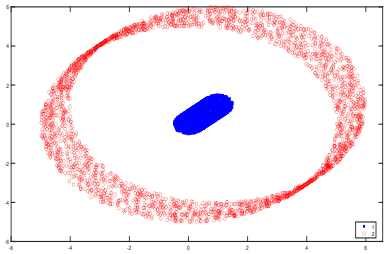
(a) k-means of the two moons data set



(b) Spectral clustering of the two moons data set



(c) k-means of the two circles data set



(d) Spectral clustering of the two circles data set

Figure 2.3: Performance of k-means and spectral clustering

k-means provides incorrect clustering results because it only cares about the distance.

It aims to minimize the total distances between each data point and its corresponding cluster centroid. In contrast, spectral clustering cares about the connectivity. Eigenvalues of graph Laplacian are referred to as the "algebraic connectivity" of the graph [12]. The connectivity information is embedded in the spectrum of the graph Laplacian matrix. Therefore, when manipulating the graph used in the spectral clustering algorithm, it is critical to preserve its spectrum.

Although spectral methods have many advantages, such as easy implementation, good clustering quality and rigorous theoretical foundations [28, 30], the high computational cost due to the involved eigen-decomposition procedure can immediately hinder their applications in emerging big data (graph) analytics tasks [9]. The eigen-decomposition procedure has a time complexity of $O(n^3)$, where n is the number of data points.

In recent years, the software package ARPACK is widely used for solving large-scale problems [9]. The overall runtime cost of ARPACK solver is proportional to $O(z^3) + (O(nz) + O(nw)) \times O(z - k)$, and the memory cost is $O(nw) + O(nz)$, where n is the number of data points, z is the arnoldi length, w is the number of nearest neighbors, and k is the number of desired eigenvalues. A sparse graph can dramatically reduce the time and memory cost due to the small w. In chapter 3, we reduce the compuational cost by reducing the number of edges in a given graph while in chapter 4, we achieve the goal by proposing a new, bottom-up graph construction

method.

Chapter 3

Towards Scalable and High-Quality Spectral Clustering via Spectrum-Preserving Sparsification

3.1 Motivation

In recent years, to address the computational bottleneck of spectral clustering, substantial effort has been devoted. Recent research efforts aim to reduce the computational cost through various kinds of approximation methods. However, none of these approximations can reliably preserve the spectrum of the original graph Laplacian

matrix, and thus may lead to degraded spectral clustering performance. For example, spectral clustering of the large data set **Covtype** that has $581,012$ instances using the existing approximation methods lead to dramatically degraded clustering quality [10].

In this chapter, section 3.2 reviews several representative methods designed for accelerating spectral clustering method. Section 3.3 reviews a spectrum-preserving sparsification method. In section 3.4, we propose a novel termination criterion based on spectral stability checking [55] for applying the sparsification tool [16] into spectral clustering. In section 3.6, we experimentally demonstrate the effectiveness of the spectrally critical off-tree edges for spectral clustering. Then, we present more experimental results based on the complete framework.

3.2 Methods for Accelerating Spectral Clustering

3.2.1 Fast Approximate Spectral Clustering

A general framework for fast approximate spectral clustering has been proposed [56]. It firstly performs k-means to get a small number of clustering centroids. Then, the

standard spectral clustering algorithm is performed on the reduced data set composed of the centroids. The data point in the original data set is assigned to the cluster as its nearest centroid. It is shown in Algorithm 7.

Algorithm 7 k-means based fast spectral clustering (KASP)

Input: A data set D with n data points $x_1, ..., x_n \in \mathbb{R}^d$, number of clusters k, number of centroids r.
Output: Clusters C_1,...,C_k.

1: Perform k-means to partition D into r clusters ;
2: Calculate the cluster means $c_1, ..., c_r$ as the r centroids;
3: Perform standard spectral clustering on the r representative points to partition them into k clusters;
4: Recover the cluster membership of each x_i by assigning it to the cluster as its nearest centroid.

3.2.2 Nyström Method

A Nyström method for accelerating the eigen-decomposition procedure has been proposed [18]. In a data set consisting of n samples, it randomly chooses p samples ($p \ll n$). It then calculates the adjacency matrix of the p samples, W_1 and the affinities between the chosen p samples and the remaining (n-p) samples, W_2.

The adjacency matrix of the remaining (n-p) samples can be approximated as $W_2^T W_1^{-1} W_2$. The adjacency matrix of the original data set can be approximated as:

$$\tilde{W} = \begin{bmatrix} W_1 & W_2 \\ W_2^T & W_2^T W_1^{-1} W_2 \end{bmatrix} \tag{3.1}$$

The approximation of the diagonal matrix $\tilde{D}$ can be obtained accordingly.

Define:

$$\overline{W}_1 = \tilde{D}^{-1/2}_{1:p,1:p} W_1 \tilde{D}^{-1/2}_{1:p,1:p}, \quad \overline{W}_2 = \tilde{D}^{-1/2}_{1:p,1:p} W_2 \tilde{D}^{-1/2}_{p+1:n,p+1:n} \tag{3.2}$$

Perform the following eigen-decomposition:

$$\overline{W}_1 = U_{\overline{W}_1} \Lambda_{\overline{W}_1} U^T_{\overline{W}_1} \tag{3.3}$$

The bottom k eigenvalues and the corresponding eigenvectors of the normalized Laplacian matrix $\tilde{L}_{sym} = I - \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2}$ can be approximated as:

$$\Lambda_{\tilde{L}_{sym}} = (n/p)(\Lambda_{\overline{W}_1})_{:,1:k}, \quad U_{\tilde{L}_{sym}} = \sqrt{p/n} \begin{bmatrix} \overline{W}_1 \\ \overline{W}_2^T \end{bmatrix} (U_{\overline{W}_1})_{:,1:k} (\Lambda^{-1}_{\overline{W}_1})_{1:k,1:k} \tag{3.4}$$

It then performs k-means to partition the approximated eigenvectors into k clusters.

3.2.3 Landmark-based Spectral Clustering

A landmark method based on the progress of sparse coding [29] has been proposed [10]. It represents the original data set by sampling a few landmark points. It then performs spectral embedding on the landmark-based representation.

3.3 Spectrum-Preserving Sparsification

A spectrum-preserving sparsification method has been proposed [16]. It involves two main steps: 1) Extract a low stretch spanning tree (LSST) from the original graph. It uses the LSST as the backbone for the sparsifier 2) To further improve the spectral similarity, it recovers a small amount of "spectral critical" off-tree edges to the LSST.

The off-tree edge budget b measures the amount of off-tree edges added to the LSST, which is defined as:

$$b = \frac{|E_P| - |V| + 1}{|V|}. \tag{3.5}$$

3.4 A Scheme for Eigenvalue Stability Checking

We propose a novel method for checking the stability of bottom eigenvalues of the graph Laplacian as follows: 1) in each iteration of adding "spectral critical" off-tree edges to the sparsifier, we compute and record the bottom eigenvalues of the graph Laplacian; 2) we determine whether more off-tree edges should be recovered by comparing with the eigenvalues computed in the previous iteration: if the variation ratio is large, more off-tree edges should be added. We store the bottom eigenvalues computed in the previous and current iterations into vectors v_p and v_{p+1}, respectively. The variation ratio of bottom eigenvalues is calculated by:

$$ratio_{var} = \frac{\|(v_p - v_{p+1})\|}{\|(v_p)\|}. \tag{3.6}$$

3.5 Algorithm Flow and Complexity Analysis

The complete scalable spectral clustering framework with the eigenvalue stability checking scheme is shown in Fig. 3.1.

We summarize the key components of the proposed framework as well as their time

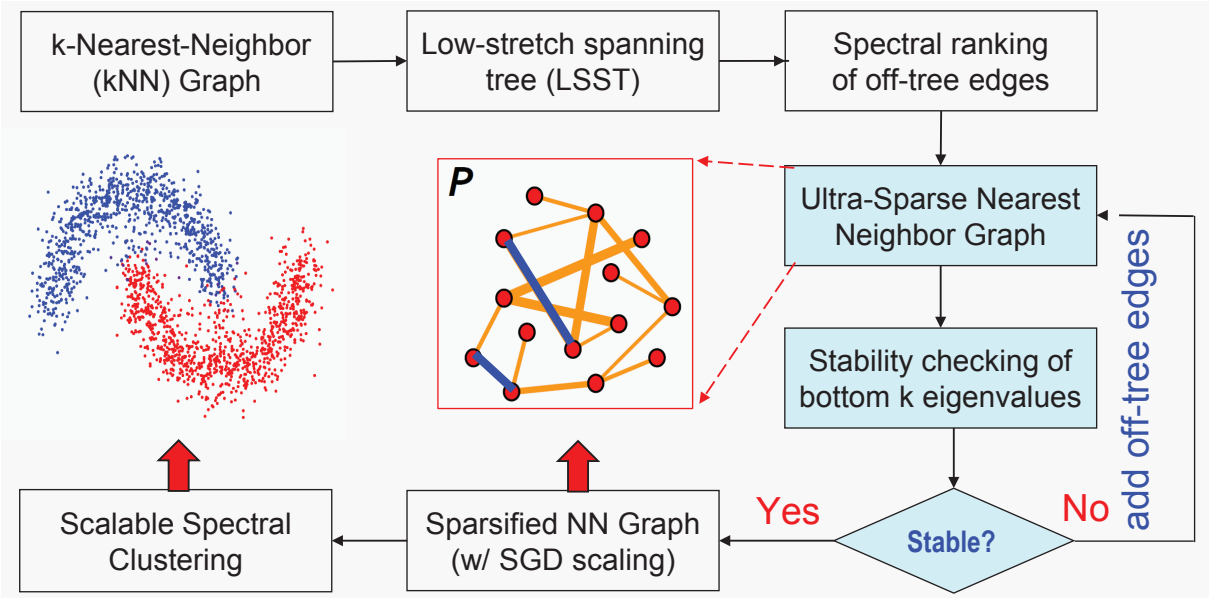


Figure 3.1: The proposed framework for scalable spectral clustering.

complexities: 1)Identify the edges to be kept in the sparsifer and check the eigen-stability in $O(m \log n)$ time, where m is the number of edges in the original graph and n is the number of nodes in the graph ; 2)Calculate the bottom k eigenvalue and corresponding eigenvectors. For ultra-sparse Laplacian matrices, the ARPACK solver is very efficient; 3)Run the k-means algorithm on approximate eigenvectors in $O(knd)$ time [1], where k is the number of clusters, and d is the dimension of the feature vectors of data points.

3.6 Experiments

All experiments for spectral clustering have been performed using MATLAB R2020b running on a Laptop with a 10th Intel(R) Core(TM) i5 CPU and 8GB RAM. The

Table 3.1
Statistics of data sets.

Data set	Size	Dimensions
PenDigits	7,494	16
USPS	9,298	256
MNIST	70,000	784
Covtype	581,012	54

spectral graph sparsification method has been implemented in C++. The reported numbers in our results have been averaged over 20 runs. Unnormalized spectral clustering is used.

3.6.1 Data Sets

We have conducted experiments on four real-world data sets. The statistics of these data sets are shown in Table 3.1.

3.6.2 The Effectiveness of Spectrally Critical Off-tree Edges for Spectral Clustering

Fig. 3.2 shows the impact of adding off-tree edges to the accuracy of spectral clustering. Fig. 3.3 shows the impact of adding off-tree edges to the NMI. The numerical results of ACC and NMI are reported in Table 3.2 and Table 3.3, respectively.

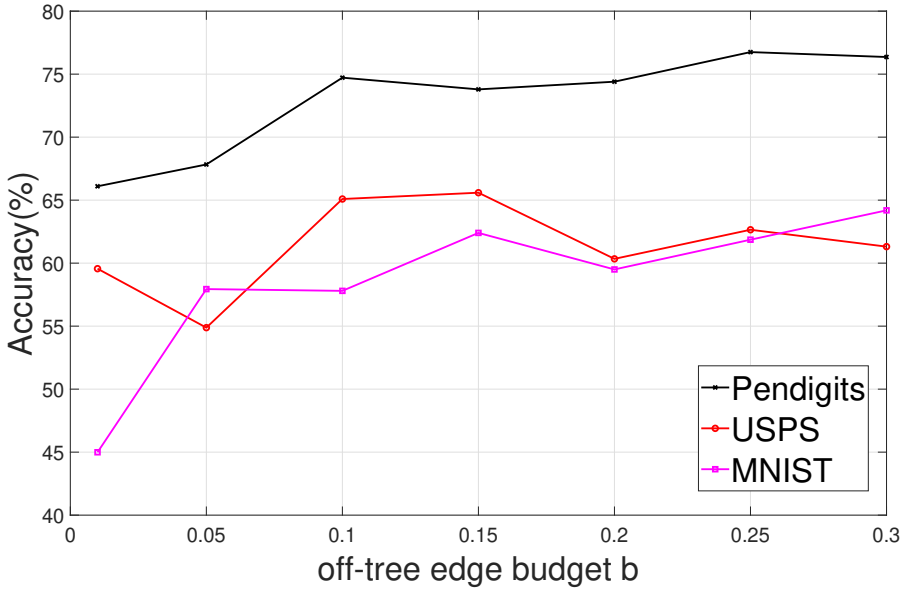


Figure 3.2: ACC VS off-tree edge budget b.

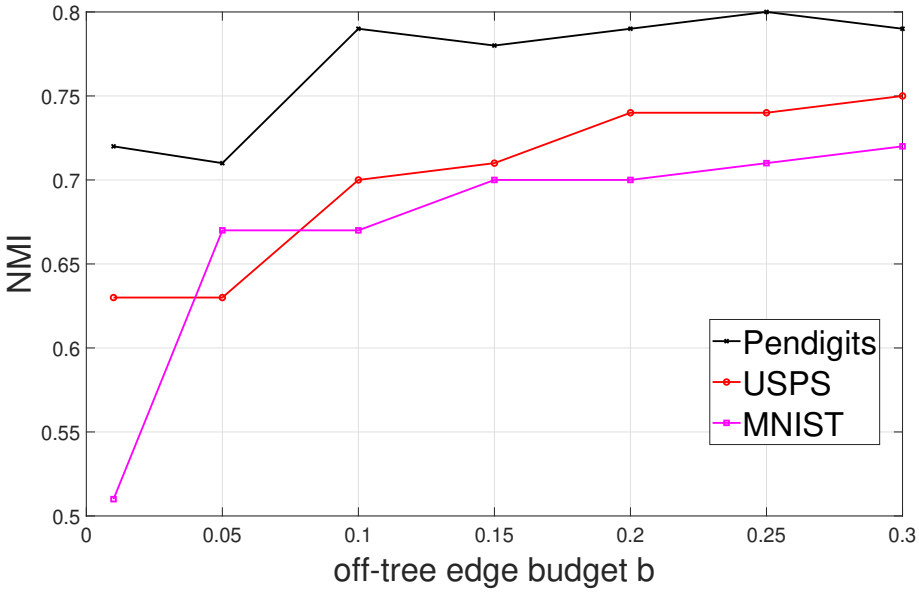


Figure 3.3: NMI VS off-tree edge budget b.

The experimental results show that 1) spectral clustering method can achieve better clustering results as the number of spectrally-critical off-tree edges increases. 2) it is necessary to prevent our method from adding redundant off-tree edges, which

Table 3.2
ACC with increasing off-tree edge budget b

Data Set	b=0.01	b=0.05	b=0.1	b=0.15	b=0.2	b=0.25	b=0.3
PenDigits	66.10	67.83	74.72	73.79	74.40	76.75	76.36
USPS	59.56	54.88	65.09	65.59	60.34	62.65	61.31
MNIST	44.99	57.94	57.80	62.40	59.50	61.86	64.19

Table 3.3
NMI with increasing off-tree edge budget b

Data Set	b=0.01	b=0.05	b=0.1	b=0.15	b=0.2	b=0.25	b=0.3
PenDigits	0.72	0.71	0.79	0.78	0.79	0.80	0.79
USPS	0.63	0.63	0.70	0.71	0.74	0.74	0.75
MNIST	0.51	0.67	0.67	0.70	0.70	0.71	0.72

would otherwise degrade the efficiency and the quality of spectral clustering algorithm. Therefore, using the proposed spectral stability checking method as the termination criterion is important for the proposed scalable spectral clustering framework.

3.6.3 Spectral Stability Checking Results

Fig. 3.4 shows the impact of adding off-tree edges to the spectral stability. The numerical results are reported in Table 3.4.

Note that the first result of $b = 0.1$ is calculated by:

$$ratio_{(b=0.1)} = \frac{\|(v_{(b=0.05)} - v_{(b=0.1)})\|}{\|(v_{(b=0.05)})\|}. \tag{3.7}$$

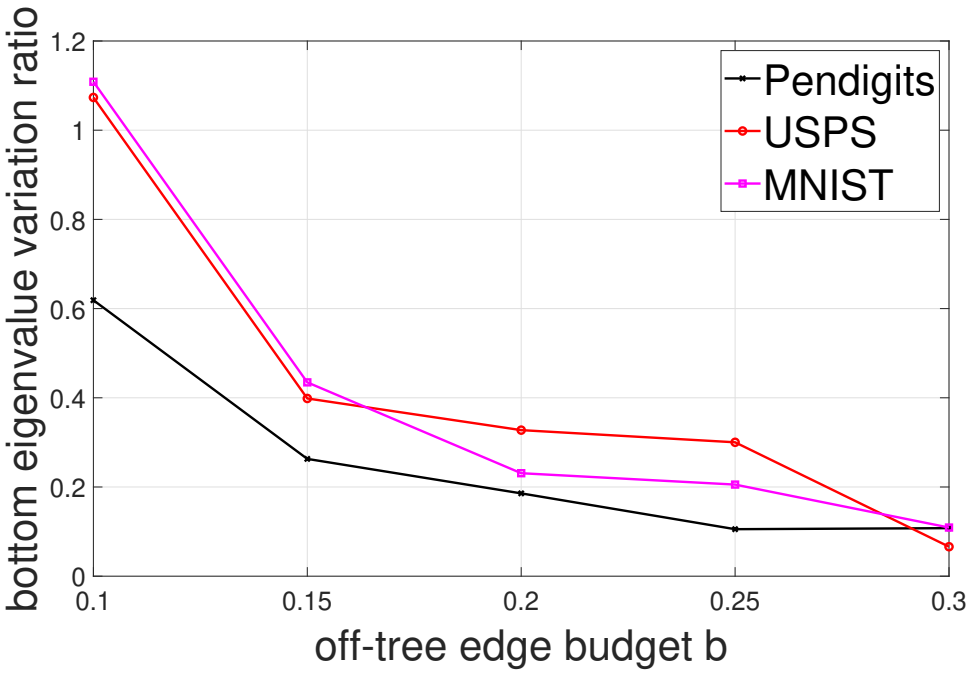


Figure 3.4: Variation ratio of bottom eigenvalues with increasing number of off-tree edges.

Table 3.4
Spectral stability checking results

	variation ratio of the bottom k eigenvalues				
Data Set	$b = 0.1$	$b = 0.15$	$b = 0.2$	$b = 0.25$	$b = 0.3$
Pendigits	0.6188	0.2630	0.1857	0.1054	0.1077
USPS	1.0734	0.3985	0.3275	0.3002	0.0661
MNIST	1.1086	0.4346	0.2309	0.2054	0.1091

As shown in Fig. 3.4 and Table 3.4, adding a small portion of off-tree edges will immediately stabilize the bottom eigenvalues of the graph Laplacian.

3.6.4 Clustering Results and Comparison

We compare our method against the following state-of-the-art algorithms: (1) the **Original SC algorithm** [9], (2) the **Nyström method** [18], (3) the **landmark-based SC (LSC) method** that uses random sampling for landmark selection [10], and (4) the **KASP SC algorithm** using k-means for centroids selection [56].

For fair comparison, we use the same parameter setting in [10] for compared algorithms: the number of sampled points in **Nyström method** (or the number of landmarks in **LSC**, or the number of centroids in **KASP**) is set to 500. Other parameters in **LSC** are set by default. We use MATLAB inbuilt k-means function with its default settings for all the methods. For the value of k in the k-NN graph, we use the setting in [51]: it is set to 10 for all the data sets.

Table 3.5 shows the ACC results. Table 3.6 shows the clustering runtime results. The graph complexities have also been provided in Table 3.7.

Table 3.5
Clustering accuracy (%)

Data Set	Orig	Nyström	KASP	LSC	Ours
PenDigits	74.36	71.99	71.56	74.25	76.75
USPS	64.31	69.31	70.62	66.28	65.59
MNIST	64.20	55.86	71.23	58.94	64.19
Covtype	48.81	33.24	27.56	22.60	48.79

Table 3.6
Clustering time (seconds)

Data Set	Orig	Nyström	KASP (centroid selection)	LSC	Ours
PenDigits	0.18	0.19	0.13 (0.49)	0.10	0.11
USPS	0.72	0.29	0.16 (3.35)	0.22	0.33
MNIST	252.59	0.95	0.18 (354.78)	0.49	3.62
Covtype	128.70	5.48	0.70 (1151.24)	3.77	5.80

Table 3.7
Graph complexity comparison

Data Set	$\|E_G\|$	$\|E_S\|$
PenDigits	50,608	9,199
USPS	68,381	10,692
MNIST	521,709	89,054
Covtype	3,312,029	610,041

The clustering time reported in Table 3.6 includes the runtime of eigen-decomposition and k-means steps in spectral clustering algorithm. For the KASP method, we also report the centroids selection time. It is a very time consuming step because it requires performing k-means on the whole data set in the original high-dimensional space. The other methods also use k-means, but they use it after the spectral embedding step, so their time costs of k-means are dramatically reduced in the low-dimensional space.

For large data sets such as **MNIST** and **Covtype**, our method achieved **70X** and **22X** times speedup, respectively. For the Covtype data set, our method achieves a significantly better clustering result than all the compared methods. Such a good clustering performance is mainly due to the superior capability of our method in

preserving the spectrum of the original graph Laplacian. On the other hand, without a good preservation of the spectrum of the original graph Laplacian, the quality of spectral clustering can be degraded dramatically. For example, the simple sampling methods such as k-means and random sampling used in the KASP and the LSC algorithms lead to substantial loss of critical structural information of the large data sets, and thus degraded the performance of spectral clustering; the clustering quality of the Nyström method strongly depends on the encoding power of the sample points chosen in its pre-processing steps [11], but it is very hard to use a small amount of chosen data points to truthfully encode the structure of the large data sets. The experimental results also show that the clustering results obtained by using spectrally sparsified graphs are even better than the results obtained by using the original graphs for some data sets, indicating that the proposed method can also help to improve the graph structure by removing redundant edges.

In the last, we show the original graph, the spanning tree, and the spectral sparsifier with b=0.1 corresponding to their adjacency matrices for the USPS data set in Fig. 3.5, Fig. 3.6, and Fig. 3.7. The initial spanning tree is a very poor approximation of the original graph while adding a small amount of spectrally critical off-tree edges leads to a good approximation.

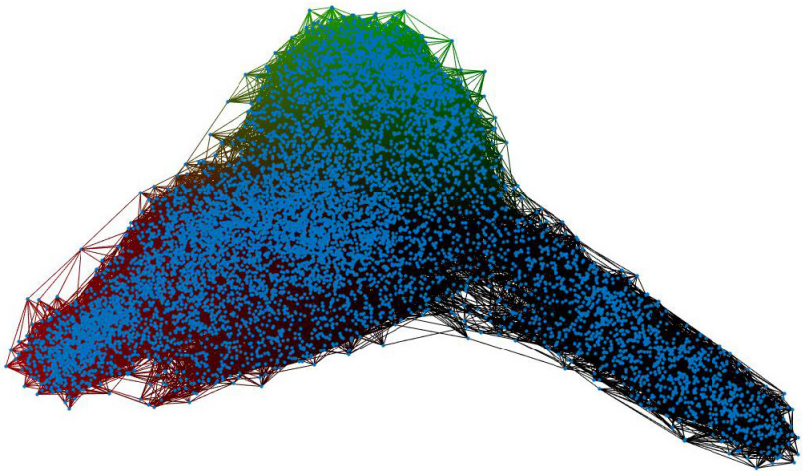


Figure 3.5: The graph corresponding to the original adjacency matrix

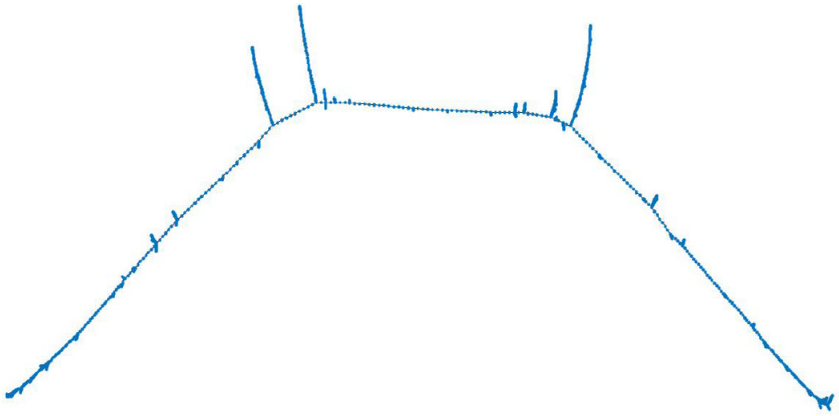


Figure 3.6: The spanning tree of the original graph

3.7 Summary

To fundamentally solve the computational challenge due to the eigen-decomposition step in spectral clustering, this work uses a spectrum-preserving spectral sparsification tool that enables to construct a very sparse sparsifier with guaranteed preservation of the original graph spectra. We propose a novel method for checking the stability

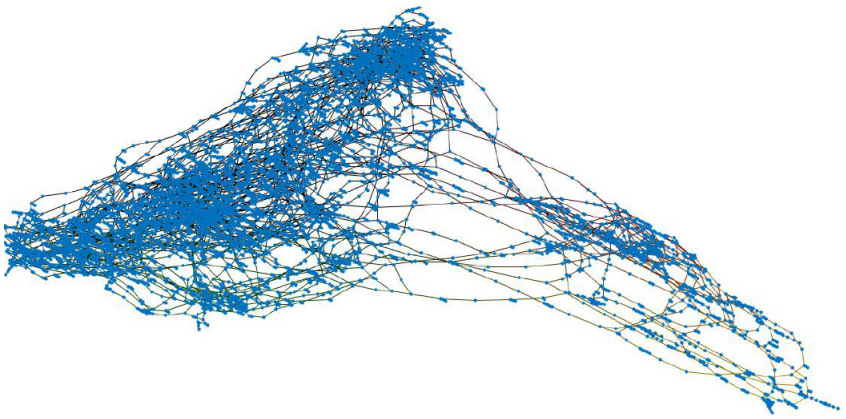


Figure 3.7: The sparsified graph corresponding to the adjacency matrix with b=0.1

of bottom eigenvalues of the graph Laplacian matrix. We experimentally demonstrate the effectiveness of the framework for spectral clustering. Experimental results on a variety of data sets show dramatically improved clustering performance when compared with state-of-the-art methods.

Chapter 4

GRASPEL: Graph Spectral Learning

In this chapter, we propose a highly-scalable graph learning (construction) method (GRASPEL). Our method aims to efficiently learn sparse undirected graphs from data.

4.1 Background

4.1.1 Multivariate Gaussian Distribution

A random variable X is said to be a Gaussian variable if its probability density function is given by:

$$f_{\mathbf{X}}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right), \tag{4.1}$$

where μ is the expectation of X, i.e. $E(X)$, and σ^2 is the variance of X, i.e. $Var(X)$, as shown in Fig. 4.1.

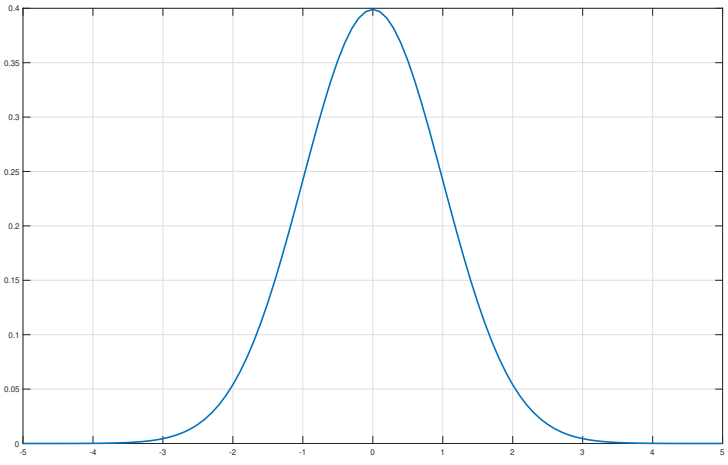


Figure 4.1: One-dimensional Gaussian distribution

Consider two random variables X and Y. The covariance between them is defined as:

$$\mathbf{Cov}(X, Y) = E[(X - EX)(Y - EY)]. \tag{4.2}$$

If X and Y are independent, then $\mathbf{Cov}(X, Y) = 0$.

Consider $\mathbf{X} = [X_1, ..., X_n]^T$, where X_1,...,X_n are n random variables, the covariance matrix of $\mathbf{X}$ is defined as:

$$\mathbf{Cov}(\mathbf{X}) = \begin{bmatrix} Var(X_1) & \mathbf{Cov}(X_1, X_2) & \cdots & \mathbf{Cov}(X_1, X_n) \\ \mathbf{Cov}(X_2, X_1) & Var(X_2) & \cdots & \mathbf{Cov}(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Cov}(X_n, X_1) & \mathbf{Cov}(X_n, X_2) & \cdots & Var(X_n) \end{bmatrix}$$

Multivariate Gaussian distribution is a multivariate generalization of the one-dimensional Gaussian distribution, as shown in Fig. 4.2. $\mathbf{X} = [X_1, ..., X_n]^T$ is said to have a multivariate Gaussian distribution if its probability density function is given by:

$$f_{\mathbf{X}}(x_1, ..., x_n) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \tag{4.3}$$

where $\mathbf{x} = [x_1, ..., x_n]^T$, $\boldsymbol{\mu} = [E(X_1), ..., E(X_n)]^T$, $\boldsymbol{\Sigma}$ is the covariance matrix of $\mathbf{X}$. We write it as $\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

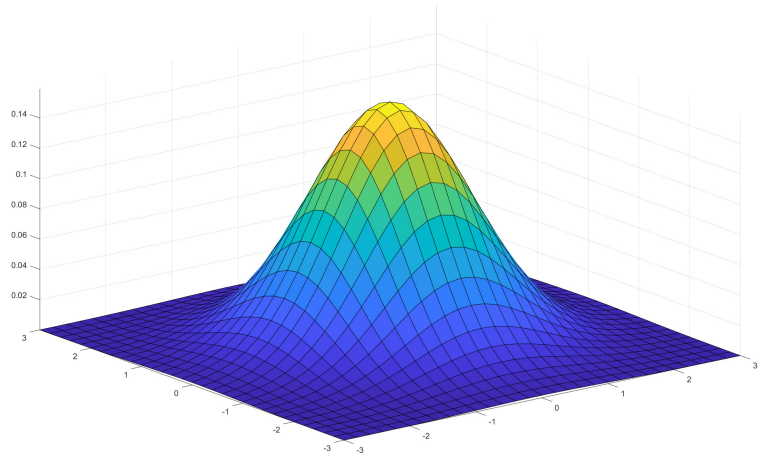


Figure 4.2: Two-dimensional Gaussian distribution

4.1.2 Maximum Likelihood Estimation

Maximum likelihood estimation is one of the most popular methods for estimating the parameters of a probability distribution. Consider that the probability density function f of a random variable X contains a parameter $\boldsymbol{\theta}$. Suppose that $X^{(1)}, ..., X^{(n)}$ are i.i.d. observations from the distribution, the likelihood function is:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(X^{(i)}) \tag{4.4}$$

The goal of maximum likelihood estimation is to find the value of $\boldsymbol{\theta}$ that maximizes the likelihood function:

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}). \tag{4.5}$$

4.1.3 Graphical Lasso

Based on the maximum likelihood estimation, a graphical lasso method has been proposed for estimating the precision (inverse covariance) matrix, or alternatively the underlying structure of multivariate Gaussian data [4, 6, 19].

Consider a random variable $\mathbf{X} = [X_1, ..., X_p]^T$ has a zero-mean multivariate Gaussian distribution, i.e. $\mathbf{X} \sim N(0, \boldsymbol{\Sigma})$. Suppose that $\mathbf{X}^{(1)}, ..., \mathbf{X}^{(n)}$ are i.i.d. observations of $\mathbf{X}$. The sample covariance matrix is:

$$S = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{X}^{(i)} - \overline{\mu})(\mathbf{X}^{(i)} - \overline{\mu})^T, \tag{4.6}$$

where

$$\overline{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}^{(i)}. \tag{4.7}$$

Let $\Theta = \Sigma^{-1}$ denotes the precision matrix, then the graphical Lasso method targets the following convex optimization task:

$$\max_{\Theta} : \log\det(\Theta) - Tr(S\Theta) - \beta\|\Theta\|_1, \tag{4.8}$$

over all non-negative definite precision matrices Θ. The precision matrix is restricted to be a graph Laplacian matrix in [45].

Under the zero-mean assumption [14], the sample covariance matrix becomes:

$$S = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}^{(i)}(\mathbf{X}^{(i)})^T. \tag{4.9}$$

Suppose that a vector $\mathbf{X}_o$ contains all the observations: $\mathbf{X}_o = [\mathbf{X}^{(1)}, ..., \mathbf{X}^{(n)}]$, then (4.9) becomes:

$$S = \frac{1}{n}\mathbf{X}_o\mathbf{X}_o^T. \tag{4.10}$$

Substituting (4.10) into (4.8) leads to:

$$\max_{\Theta} : \log\det(\Theta) - Tr(\frac{1}{n}\mathbf{X}_o\mathbf{X}_o^T\Theta) - \beta\|\Theta\|_1, \tag{4.11}$$

where $\|\cdot\|_1$ denotes the entry-wise $l1$ norm: $\|\Theta\|_1 = \sum_{i,j}|\Theta_{i,j}|$.

In [14], the precision matrix is set to:

$$\Theta = L + \frac{I}{\sigma^2}, \tag{4.12}$$

where I denotes the identity matrix and σ is a regularization parameter. The optimization model (4.11) aims to achieve the following desired characteristics:

The smoothness of graph signals. The graph signals corresponding to the real-world data should be sufficiently smooth on the learned graph structure: the signal values change gradually across connected neighboring nodes. The smoothness of a set of signals $\mathbf{X}_o$ over the graph G can be measured with the following matrix trace [26]:

$$Q(\mathbf{X}_o, L) = Tr(\mathbf{X}_o L {\mathbf{X}_o}^T). \tag{4.13}$$

When $\Theta = L + \frac{I}{\sigma^2}$, the term $Tr(\frac{1}{n}\mathbf{X}_o\mathbf{X}_o^T\Theta)$ in (4.11) includes $Tr(\mathbf{X}_oL\mathbf{X}_o^T)$. To maximize (4.11), $Tr(\mathbf{X}_oL\mathbf{X}_o^T)$ tends to have a small value.

The sparsity of the estimated graph (Laplacian). (4.11) has been proposed to learn the structure in an undirected Gaussian graphical model using $l1$ regularization to control the sparsity of the precision matrix. $\beta\|\Theta\|_1$ is used as a regularization term to enforce sparsity.

However, the high complexities of solving the log-determinant models hinder their applications in large-scale problems. Besides, tuning the parameters to control the graph sparsity is a nontrivial task.

4.2 Our Method

4.2.1 Spectral Analysis

The first term in (4.11) can be expressed as:

$$\begin{aligned} &\log\det(\Theta) = \log\det(L + \tfrac{I}{\sigma^2}) = \log\textstyle\prod_{i=1}^{n}(\tfrac{1}{\sigma^2} + \lambda_i) \\ &= \sum_{i=1}^{n}\log(\tfrac{1}{\sigma^2} + \lambda_i). \end{aligned} \tag{4.14}$$

[33] has shown that:

$$\mathbf{L} = \sum_{(u,v)\in E} w_{uv}\mathbf{e}_{uv}\mathbf{e}_{uv}^T, \tag{4.15}$$

where $e_u \in \mathbb{R}^N$ denotes the standard basis vector with all zero entries except for the u-th entry being 1, and $e_{uv} = e_u - e_v$.

By substituting (4.15) into the precision matrix $\Theta = L + \frac{I}{\sigma^2}$, the precision matrix can be expressed as:

$$\Theta = \frac{I}{\sigma^2} + \sum_{(u,v)\in E} w_{uv}\mathbf{e}_{uv}\mathbf{e}_{uv}^T. \tag{4.16}$$

Then the second term in (4.11) can be expressed as :

$$\begin{aligned}
&Tr(\tfrac{1}{n}\mathbf{X}_o\mathbf{X}_o^T\Theta) = \tfrac{1}{n}Tr(\mathbf{X}_o\mathbf{X}_o^T(\tfrac{I}{\sigma^2} + \sum_{(u,v)\in E} w_{uv}\mathbf{e}_{uv}\mathbf{e}_{uv}^T)) \\
&= \tfrac{1}{n}Tr(\tfrac{\mathbf{X}_o\mathbf{X}_o^T}{\sigma^2} + \mathbf{X}_o\mathbf{X}_o^T \sum_{(u,v)\in E} w_{uv}\mathbf{e}_{uv}\mathbf{e}_{uv}^T) \\
&= \tfrac{1}{n}Tr(\tfrac{\mathbf{X}_o\mathbf{X}_o^T}{\sigma^2}) + \tfrac{1}{n}Tr(\mathbf{X}_o\mathbf{X}_o^T \sum_{(u,v)\in E} w_{uv}\mathbf{e}_{uv}\mathbf{e}_{uv}^T) \\
&= \tfrac{1}{n}Tr(\tfrac{\mathbf{X}_o\mathbf{X}_o^T}{\sigma^2}) + \tfrac{1}{n}Tr(\sum_{(u,v)\in E} w_{uv}\mathbf{X}_o^T\mathbf{e}_{uv}\mathbf{e}_{uv}^T\mathbf{X}_o).
\end{aligned} \tag{4.17}$$

The trace has the property: $Tr(vv^t) = \|v\|_2^2$, where v is a column vector. So, if we

take $\mathbf{X}_o^T \mathbf{e}_{uv}$ as a column vector, we have:

$$\begin{aligned} &Tr(\mathbf{X}_o^T \mathbf{e}_{uv} \mathbf{e}_{uv}^T \mathbf{X}_o) = Tr((\mathbf{X}_o^T \mathbf{e}_{uv})(\mathbf{X}_o^T \mathbf{e}_{uv})^T) \\ &= \|\mathbf{X}_o^T \mathbf{e}_{uv}\|_2^2. \end{aligned} \tag{4.18}$$

Then the following expansion can be obtained:

$$Tr(\sum_{(u,v)\in E} w_{uv} \mathbf{X}_o^T \mathbf{e}_{uv} \mathbf{e}_{uv}^T \mathbf{X}_o) = \sum_{(u,v)\in E} w_{uv} \|\mathbf{X}_o^T \mathbf{e}_{uv}\|_2^2. \tag{4.19}$$

Substituting (4.19) into (4.17), we have:

$$Tr(\tfrac{1}{n} \mathbf{X}_o \mathbf{X}_o^T \Theta) = \tfrac{1}{n} Tr(\tfrac{\mathbf{X}_o \mathbf{X}_o^T}{\sigma^2}) + \tfrac{1}{n} \sum_{(u,v)\in E} w_{uv} \|\mathbf{X}_o^T \mathbf{e}_{uv}\|_2^2. \tag{4.20}$$

The third term in (4.11) can be expressed as:

$$\begin{aligned} &\beta \|\Theta\|_1 = \beta \|L + \tfrac{I}{\sigma^2}\|_1 \\ &= \beta \sum_{i,j} |(L + \tfrac{I}{\sigma^2})_{i,j}| \\ &= \beta \sum_{i,j} |L_{i,j}| + \tfrac{\beta}{\sigma^2} \sum_{i,j} |I_{i,j}|. \end{aligned} \tag{4.21}$$

$\sum_{i,j} |L_{i,j}|$ is proportional to $\sum_{(u,v)\in E} w_{uv}$. $\frac{\beta}{\sigma^2} \sum_{i,j} |I_{i,j}|$ is a constant number and it is proportional to the number of vertices in the graph. Then (4.21) can be expressed as:

$$\beta \|\Theta\|_1 = \overline{\beta} \sum_{(u,v)\in E} w_{uv} + \alpha, \tag{4.22}$$

where α denotes a constant number.

Substituting (4.14), (4.20), and (4.22) into (4.11), the objective function becomes:

$$\begin{aligned} F = & \sum_{i=1}^{n} \log(\tfrac{1}{\sigma^2} + \lambda_i) - \tfrac{1}{n} \tfrac{Tr(\mathbf{X}_o \mathbf{X}_o^T)}{\sigma^2} - \tfrac{1}{n} \sum_{(u,v)\in E} w_{uv} \|\mathbf{X}_o^T \mathbf{e}_{uv}\|_2^2 \\ & - \overline{\beta} \sum_{(u,v)\in E} w_{uv} - \alpha. \end{aligned} \tag{4.23}$$

We propose a novel way to analyze the impact of adding an edge into the graph to the objective function: adding an edge into the graph is equal to increasing its weight from 0 to its edge weight. For a candidate edge, we calculate the partial derivative with respect to its weight to evaluate the impact of adding it into the graph to the objective function.

For a candidate edge(s,t), the partial derivative with respect to its weight w_{st} is:

$$\frac{\partial F}{\partial w_{st}} = \sum_{i=1}^{n} \frac{1}{\frac{1}{\sigma^2} + \lambda_i} \frac{\partial \lambda_i}{\partial w_{st}} - \frac{1}{n} \|\mathbf{X}_o^T \mathbf{e}_{st}\|_2^2 - \overline{\beta}. \tag{4.24}$$

We propose the following spectral analysis to solve (4.24):

Let L_P denotes the Laplacian matrix of an undirected graph P, and u_i denote the eigenvector of L_P corresponding to the i-th eigenvalue λ_i that satisfies:

$$L_p u_i = \lambda_i u_i, \tag{4.25}$$

then we have the following eigenvalue perturbation analysis:

$$\left(L_P + \delta L_P\right)\left(u_i + \delta u_i\right) = \left(\lambda_i + \delta\lambda_i\right)\left(u_i + \delta u_i\right), \tag{4.26}$$

where a perturbation δL_P that includes a new edge is applied to L_P, resulting in perturbed eigenvalues and eigenvectors $(\lambda_i + \delta\lambda_i)$ and $(u_i + \delta u_i)$ for $i = 1, ..., n$, respectively.

Keeping only the first-order terms leads to:

$$L_P \delta u_i + \delta L_P u_i = \lambda_i \delta u_i + \delta\lambda_i u_i. \tag{4.27}$$

Write δu_i in terms of the original eigenvectors u_j for $j = 1, ..., n$:

$$\delta u_i = \sum_{j=1}^{n} \alpha_j u_j. \tag{4.28}$$

Substituting (4.28) into (4.27) leads to:

$$L_P \sum_{j=1}^{n} \alpha_j u_j + \delta L_P u_i = \lambda_i \sum_{j=1}^{n} \alpha_j u_j + \delta \lambda_i u_i. \tag{4.29}$$

Multiplying both sides of (4.29) by u_i^T leads to:

$$u_i^T L_P \sum_{j=1}^{n} \alpha_j u_j + u_i^T \delta L_P u_i = \lambda_i u_i^T \sum_{j=1}^{n} \alpha_j u_j + \delta \lambda_i u_i^T u_i. \tag{4.30}$$

Since u_i for $i = 1, ..., n$ are unit-length, mutually-orthogonal eigenvectors, we have:

$$u_i^T L_P \sum_{j=1}^{n} \alpha_j u_j = \alpha_i u_i^T L_P u_i, \qquad \lambda_i u_i^T \sum_{j=1}^{n} \alpha_j u_j = \alpha_i u_i^T \lambda_i u_i. \tag{4.31}$$

Multiplying both sides of (4.25) by $\alpha_i u_i^T$ leads to:

$$\alpha_i u_i^T L_P u_i = \alpha_i u_i^T \lambda_i u_i. \tag{4.32}$$

Substituting (4.32) into (4.31) leads to:

$$u_i^T L_P \sum_{j=1}^{n} \alpha_j u_j = \lambda_i u_i^T \sum_{j=1}^{n} \alpha_j u_j. \tag{4.33}$$

Substituting (4.33) into (4.30) leads to:

$$u_i^T \delta L_P u_i = \delta\lambda_i u_i^T u_i = \delta\lambda_i. \tag{4.34}$$

[33] has shown that an edge(s,t) leads to the following perturbation:

$$\delta \mathbf{L}_P = w_{st} \mathbf{e}_{st} \mathbf{e}_{st}^T. \tag{4.35}$$

Substituting (4.35) into (4.34), we have:

$$u_i^T w_{st} \mathbf{e}_{st} \mathbf{e}_{st}^T u_i = w_{st} (u_i^T \mathbf{e}_{st})^2 = \delta\lambda_i. \tag{4.36}$$

Because adding a new edge into the graph is equal to changing its weight from 0 to its edge weight, we have $w_{st} = \delta w_{st}$. Substituting it into (4.36), we have:

$$\frac{\delta\lambda_i}{\delta w_{st}} = (u_i^T \mathbf{e}_{st})^2, \tag{4.37}$$

Substituting (4.37) into (4.24), we have:

$$\frac{\partial F}{\partial w_{st}} = \sum_{i=1}^{n} \frac{(u_i^T \mathbf{e}_{st})^2}{\frac{1}{\sigma^2} + \lambda_i} - \frac{1}{n} \|\mathbf{X}_o^T \mathbf{e}_{st}\|_2^2 - \overline{\beta}. \tag{4.38}$$

According to (4.38), to precisely evaluate the impact of adding an edge into the graph to the objective function, all the Laplacian eigenvectors are required to be used. To reduce the computational cost, we approximate the calculation by using only the first r eigenvectors. Then, (4.38) can be expressed as:

$$\frac{\partial F}{\partial w_{st}} = \sum_{i=2}^{r} \frac{(u_i^T \mathbf{e}_{st})^2}{\frac{1}{\sigma^2} + \lambda_i} - \frac{1}{n} \|\mathbf{X}_o^T \mathbf{e}_{st}\|_2^2 - \overline{\beta}. \tag{4.39}$$

Construct a subspace matrix U for spectral graph embedding using the first $r-1$ nontrivial eigenvectors:

$$U = \left[\frac{u_2}{\sqrt{\frac{1}{\sigma^2} + \lambda_2}}, ..., \frac{u_r}{\sqrt{\frac{1}{\sigma^2} + \lambda_r}} \right], \tag{4.40}$$

then (4.39) can be rewritten as:

$$\frac{\partial F}{\partial w_{st}} = \|U^T \mathbf{e}_{st}\|_2^2 - \frac{1}{n}\|\mathbf{X}_o^T \mathbf{e}_{st}\|_2^2 - \overline{\beta}. \tag{4.41}$$

Define:

$$\eta_{st} = \frac{\|U^T \mathbf{e}_{st}\|_2^2}{\frac{1}{n}\|\mathbf{X}_o^T \mathbf{e}_{st}\|_2^2}, \tag{4.42}$$

then (4.41) can be rewritten as:

$$\frac{\partial F}{\partial w_{st}} = (1 - \frac{1}{\eta_{st}})\|U^T \mathbf{e}_{st}\|_2^2 - \overline{\beta}, \tag{4.43}$$

implying that as long as $\eta_{st} > 1$ holds for edge(s,t) and $\overline{\beta}$ is properly selected, the greater η_{st} and $\|U^T \mathbf{e}_{st}\|_2^2$ values can lead to more significant improvement (increase) of the objective function.

4.2.2 Clustering-Informative Edge

Graph clustering is an unsupervised task. It relies on the graph structure only to assign data points to clusters.

Based on (4.36), we propose the following theorem:

Theorem 1 *The criticality c_{st} of a candidate edge(s,t) on the Laplacian eigenvalue λ_i can be estimated by $c_{st} = w_{st}\left(u_i^T e_{st}\right)^2$.*

Theorem 1 is very useful due to the special significance of the Fiedler eigenvalue and the corresponding Fiedler vector in graph clustering and partitioning. [12, 46] have shown that Fiedler eigenvalue and Fiedler vector can provide a good approximation to the optimal cut. So the impact of an edge to the clustering structure can be approximately measured by estimating its criticality on the Fiedler eigenvalue. We define an edge to be a clustering-informative edge if it has significant impact on the Fiedler eigenvalue. Constructing a high-quality sparse graph for the clustering task implies that sufficient clustering information must be provided with a limited amount of edges. Therefore, clustering-informative edges should be identified and added to the graph.

4.2.3 Embedding Distortion

Consider a data matrix $\boldsymbol{Fea} \in \mathbb{R}^{n\times m}$. It has n data points, each data point is represented by a $\mathbb{R}^{1\times m}$ feature vector. The euclidean distance between the s-th data point and the t-th data point is:

$$dist(s,t) = \|\boldsymbol{Fea}(s) - \boldsymbol{Fea}(t)\| = \|\boldsymbol{Fea}^T \mathbf{e}_{st}\| \tag{4.44}$$

So, in (4.42), $\|\mathbf{X}_o^T \mathbf{e}_{st}\|_2^2$ is the squared euclidean distance between the s-th data point and the t-th data point in the original data vector space $\mathbf{X}_o$. $\|U^T \mathbf{e}_{st}\|_2^2$ is the squared euclidean distance between the two data points in the spectral embedding vector space $\mathbf{U}$.

Therefore, (4.42) is the *embedding distortion* of edge(s,t). The edges with large distortion should be added to the graph.

In practice, instead of using the whole embedding space U defined in (4.40), we only use the first non-trivial eigenvector to reduce the computational cost. When the cluster number is very large, more eigenvectors can be considered to be included in the embedding space.

4.2.4 A Distortion-based Edge Sampling Method

Evaluating all the potential connections in the graph can be a very time-consuming procedure. To further reduce the computational cost, we propose a novel distortion-based edge sampling method.

We propose to find a small amount of "promising" candidate edges and we only examine these "promising" candidate edges. By embedding the latest graph G using its Fiedler vector, each node is embedded to one element in the Fiedler vector. Thus the original data vector space is embedded into a 1D array. The distance between two nodes in this embedding space is simply the difference of their corresponding values in the Fiedler vector. Therefore, we sort the Fiedler vector, and subsequently find the connections between the top and bottom few nodes in the 1D sorted node array. These edges have large distances in the embedding space.

As illustrated in Section 4.2.3, the distortion of an edge (s, t) between original data vector space and the embedding space is defined as:

$$distortion(s,t) = \frac{dist_{emb}(s,t)^2}{dist_{data}(s,t)^2}, \tag{4.45}$$

so an edge(s, t) with large $dist_{emb}(s, t)$ has a high probability to have large distortion. Only a small portion of edges with large distances in the embedding space are chosen as the "promising" candidate edges. We only calculate their distortions. The candidate edges with top embedding distortions will be added to the latest graph.

4.2.5 Edge Weight Calculation

To measure the similarity between two data points, many sophisticated similarity functions have been proposed. The Gaussian kernel is most commonly used for spectral clustering. Based on Gaussian kernel, the edge weight (similarity) between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is calculated as:

$$w_{i,j} = exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{4.46}$$

However, the proper σ value is usually problem-dependent and can be very difficult to find. The performance of spectral clustering can be very unstable under different choices of the σ value [52, 59]. σ is a human-specified parameter and it is commonly selected manually[59] . [35] proposed a systematic way for determining the σ value by performing clustering algorithms on many different σ values in a manually-selected range to find the one with the tightest clusters. However, in many cases, a single σ value cannot well capture the different local statistical patterns in the graph. It has been shown that even the optimal σ value can lead to unsatisfactory clustering results when multiple scales appear in the data [59].

To address these issues, a self-tuning technique has been proposed [59]. Instead of

using a single σ value for all the data points, each data point $\mathbf{x}_i$ has its own scaling parameter σ_i. The distance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is different from the view of $\mathbf{x}_i$ and $\mathbf{x}_j$. For $\mathbf{x}_i$, the distance is $\frac{1}{\sigma_i}\|\mathbf{x}_i - \mathbf{x}_j\|$, while for $\mathbf{x}_j$, the distance is $\frac{1}{\sigma_j}\|\mathbf{x}_i - \mathbf{x}_j\|$. The edge weight (similarity) between $\mathbf{x}_i$ and $\mathbf{x}_j$ is calculated as:

$$w_{i,j} = exp\left(-\frac{1}{2\sigma_i\sigma_j}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{4.47}$$

The scaling parameter σ_i for the data point $\mathbf{x}_i$ is defined as the distance between $\mathbf{x}_i$ and its t-th nearest neighbor. In spectral clustering tasks, it can also be defined as the average of its distances to the k nearest neighbors [9]. Thus, to determine the scaling parameter for a data point, all the edges connected to it have to be determined first. That is, the graph has to be constructed.

However, in the GRASPEL, edges are added to the graph iteratively. Thus, the self-tuning technique is not suitable for GRASPEL. Consider the following case: when we add an edge to the graph, the scaling parameters of its two nodes will be changed because the edge changes the neighborhood patterns of the two nodes. As a result, the weights of all the edges connected to the two nodes needed to be updated based on their new scaling parameters. More importantly, different edge evaluation orders lead to different graphs.

To resolve it, we simply use the reciprocal of the distance between two data points as their similarity. By doing so, the impact of the scaling parameter for edge weight calculation can be excluded, so the graph quality improvement of our method due to solely the identified critical edge connections.

4.2.6 Detailed Steps in GRASPEL

GRASPEL takes advantages of both the nearest neighbor connection strategy and the optimization problem-solving strategy to learn the graph from data.

4.2.6.1 Initial Graph Construction

In order to perform spectral analysis, an initial graph is required to be given. Intuitively, k-NN graph can be used as the initial graph because it is able to approximate the local manifold of the data set [41]. However, it has several drawbacks as aforementioned.

To preserve its capability of representing the most essential local manifold while solving its drawbacks, GRASPEL starts with constructing a k-NN graph using a very small k value (e.g. k=2), and strive to iteratively improve the graph quality by adding a small portion of critical edges through using a novel way to solve the optimization

problem (4.11).

In addition to the consideration of graph sparsity, noisy edge pruning is also an important factor when considering the use of 2-NN graph. Spectral clustering is well known for its vulnerability to noise: even very few noisy bridges between two internally well-connected clusters can mislead the algorithm to report the two clusters as a single cluster [24].

4.2.6.2 Critical Edge Identification

Once the eigenvectors of the latest graph Laplacian are available, we sample a small portion of "promising" candidate edges by using the distortion-based edge sampling method proposed in Section 4.2.4. We then calculate the distortions of the sampled "promising" candidate edges. The edges with top embedding distortions are identified as the critical edges and added to the graph.

4.2.6.3 Termination Criterion

Our method employs an iterative procedure to repeatedly add a few critical edges into the graph and thereby improving the graph quality. The natural termination criterion is when the objective function can no longer be increased. The convergence

of our method can also be determined based on the spectral stability: that is when the bottom eigenvalues become stable. Thus, we use the eigenvalue stability checking scheme proposed in Section 3.6.3 as the termination criterion. In practice, we found that the algorithm converges rapidly within very few iterations.

4.2.7 Algorithm Flow and Complexity Analysis

The GRASPEL algorithm is shown in Algorithm 8. Theoretically, the latest efficient Laplacian eigensolver has potential to compute the spectral embedding space in nearly-linear time. The embedding distortion of each candidate edge can be calculated in constant time. Thus, by limiting the searching scope within only a small number of top and bottom nodes in the sorted 1D array, critical edges can be identified very efficiently.

4.3 Experiments

The performance of spectral clustering highly depends on the quality of the underlying graph[2, 21, 34, 36]. Classical graph construction methods are often not suitable for complex real-world data sets, causing degraded spectral clustering performance[36]. Thus, there is a pressing need to develop an effective graph construction method

Algorithm 8 Fiedler based GRASPEL

Input: A data set with n data points $x_1, ..., x_n \in \mathbb{R}^d$, sample ratio ϵ, edge selection ratio ζ. **Output:** The learned graph.

1: Construct a 2-NN graph G as the initial graph.
2: **while** Termination criterion not met **do**
3: Embed the latest graph G using its Fiedler vector and sort the nodes into a 1D array I_{node}
4: Obtain node set N_{top} (N_{bottom}) by including the top (bottom) $\frac{\epsilon N}{2}$ nodes in I_{node}
5: Evaluate the embedding distortions of all the edges with one node in N_{top} and another node in N_{bottom} ;
6: Select top ζN edges based on the evaluation results and add them to the graph G;
7: Check the termination criterion.
8: **end while**

for spectral clustering. In this section, we apply GRASPEL to learn the underlying graph from data sets. Experimental results show that the learned graph can result in drastically improved efficiency and accuracy in spectral clustering tasks.

The following experiments are performed using MATLAB R2020b running on a Laptop with 10th Intel(R) Core(TM) i5 CPU and 8GB RAM. Spectral clustering algorithm has intrinsic randomness and the clustering result of each run is different. So the reported numbers in our experiments have been averaged over 20 runs. Unnormalized spectral clustering is used.

4.3.1 Spectral Stability Checking Results

To determine the number of iterations needed for the GRASPEL, we perform spectral stability checking and report the results in Table 4.1 and Fig. 4.3.

Table 4.1
Spectral stability checking results

	variation ratio of the bottom 50 eigenvalues		
Data Set	1st iter	2nd iter	3rd iter
COIL20	0.1483	0.2038	0.0908
Pendigits	5.0048	0.2609	0.2065
USPS	0.5869	0.0902	0.0246
MNIST	0.9544	0.0171	0.0131

Experimental results show that the variation ratio becomes very small after three iterations and thus justifies another iteration for adding more critical edges into the graph is not necessary.

4.3.2 Convergence Results

To show the convergence property of the proposed method, we report the changes of ACC and NMI with increasing number of iterations in Table 4.2. Graph complexity changing results are also provided in Table 4.3.

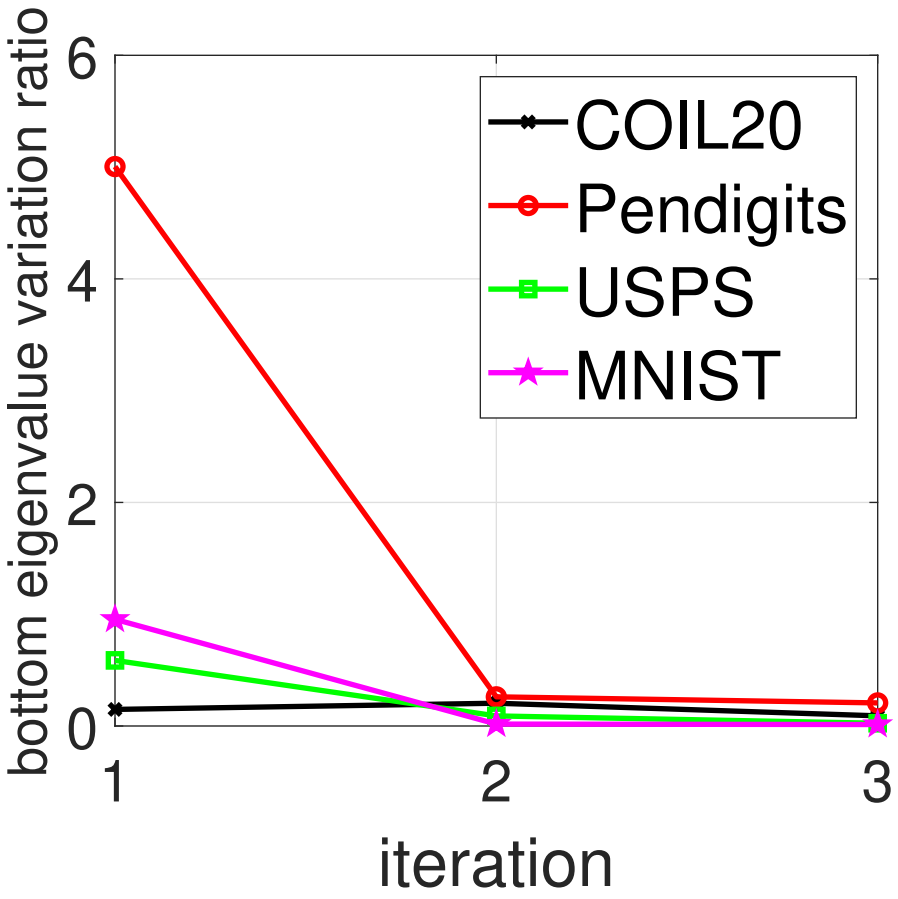


Figure 4.3: Variation ratio of bottom eigenvalues with increasing number of iterations

Table 4.2
Convergence results

	ACC(%)/ NMI			
Data Set	2-NN	1st iter	2nd iter	3rd iter
COIL20	74.75/0.90	76.23/0.91	84.07/0.94	86.46/0.94
Pendigits	10.94/0.05	21.07/0.29	59.07/0.71	82.40/0.79
USPS	16.92/0.03	83.56/0.83	84.10/0.83	87.62/0.83
MNIST	11.29/0.01	71.83/0.78	72.78/0.78	74.63/0.78

Table 4.3
Graph complexity results

	$\|E\|$			
Data Set	2-NN	1st iter	2nd iter	3rd iter
COIL20	1561	1705	1849	1993
Pendigits	10932	14679	18426	22173
USPS	14427	14891	15355	15819
MNIST	110061	113561	117061	120561

Fig. 4.4 and Fig. 4.5 show that GRASPEL has a very good convergence property. The peak performance is achieved within very few iterations. This good convergence property also enables to generate a very sparse graph, as shown in Fig. 4.6

4.3.3 Clustering Results and Comparison

For the compared standard k-NN method, we follow the experimental settings in [9]: first generate a distance matrix. Then convert the distance matrix to a sparse similarity matrix by using the self-tuning technique. For the value of k in k-NN graph, we use the setting in [51]: it is set to 10 for all the data sets.

For the compared consensus method, for fair comparison with the standard k-NN method, we use the same 10-NN graph used in the standard k-NN method as its

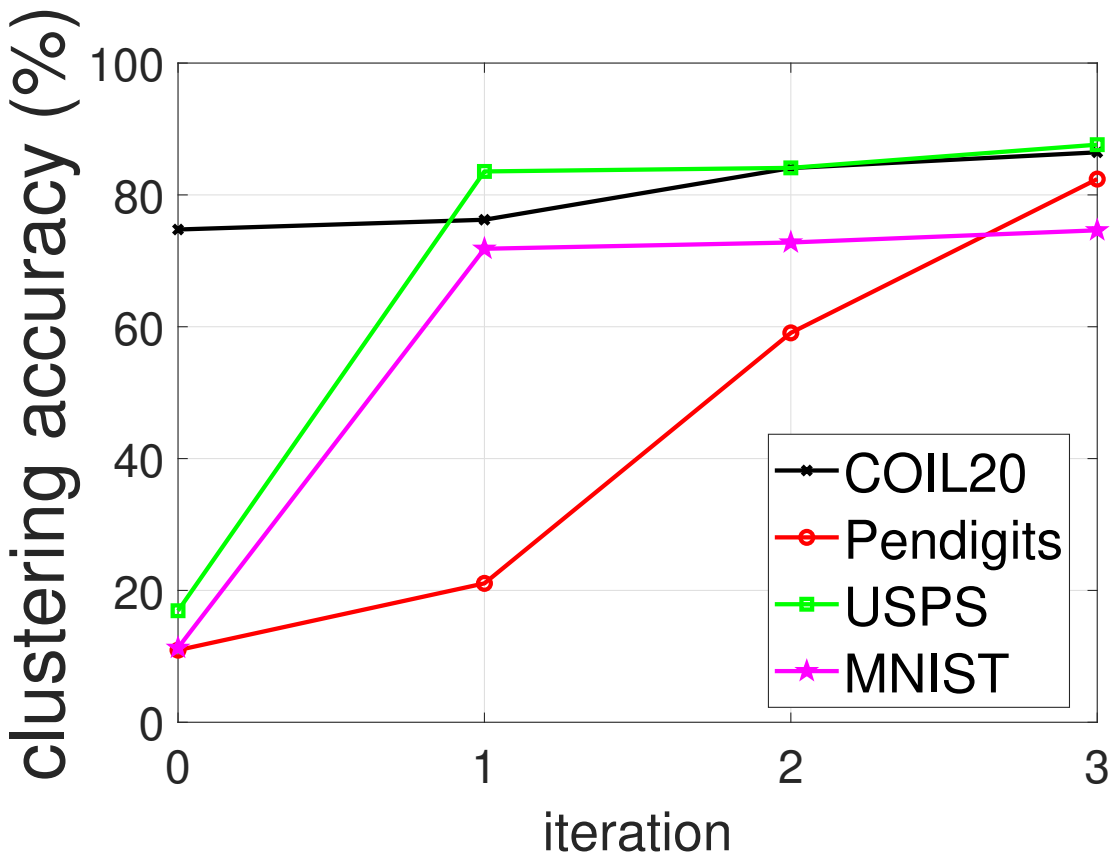


Figure 4.4: ACC with increasing number of iterations

input graph. [38] doesn't indicate how to set the threshold, which is a critical parameter for edge pruning. We search a set of different threshold values and report the corresponding clustering results in Table 4.4. The best clustering result for each data set is used to compare with other methods.

Table 4.4
Clustering results of the consensus method with different threshold values

	ACC(%)/ NMI					
Data Set	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$	$\tau = 5$	$\tau = 6$
COIL-20	76.25/0.86	77.09/0.86	77.60/0.87	80.80/0.88	81.60/0.90	58.99/0.80
Pendigits	71.08/0.79	63.77/0.76	36.69/0.56	25.99/0.41	18.37/0.26	10.73/0.03
USPS	67.56/0.81	68.54/0.81	31.67/0.43	16.82/0.02	16.80/0.02	16.71/0.02

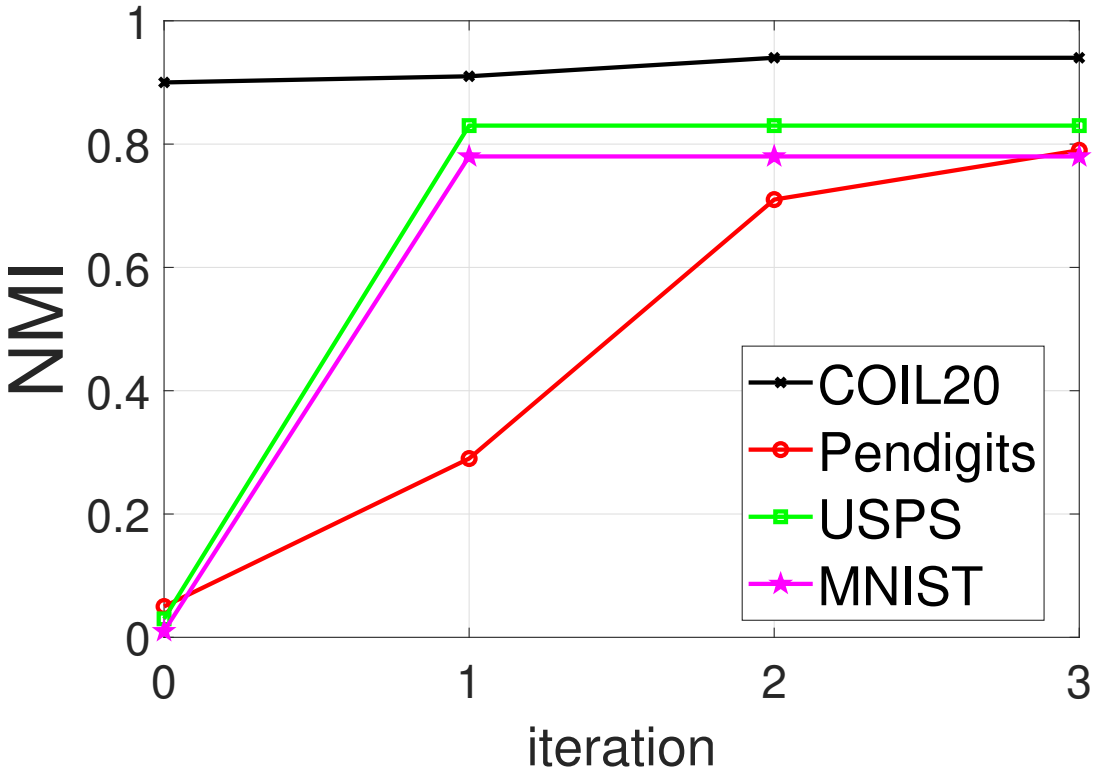


Figure 4.5: NMI with increasing number of iterations

We show the spectral clustering performance of the four methods in Table 4.5 and Table 4.6.

Table 4.5
ACC and NMI results

	ACC(%)/ NMI			
Data Set	Standard k-NN	Consensus	LGSS	Our method
COIL20	75.72/0.86	81.60/0.90	85.49/**0.95**	**86.46**/0.94
PenDigits	74.36/0.79	71.08/0.79	74.53/0.77	**82.40**/0.79
USPS	64.31/0.79	68.54/0.81	81.50/**0.84**	**87.62**/0.83
MNIST	64.20/0.74	-	-	**74.63/0.78**

The time reported in Table 4.6 includes the time costs of the eigen-decomposition

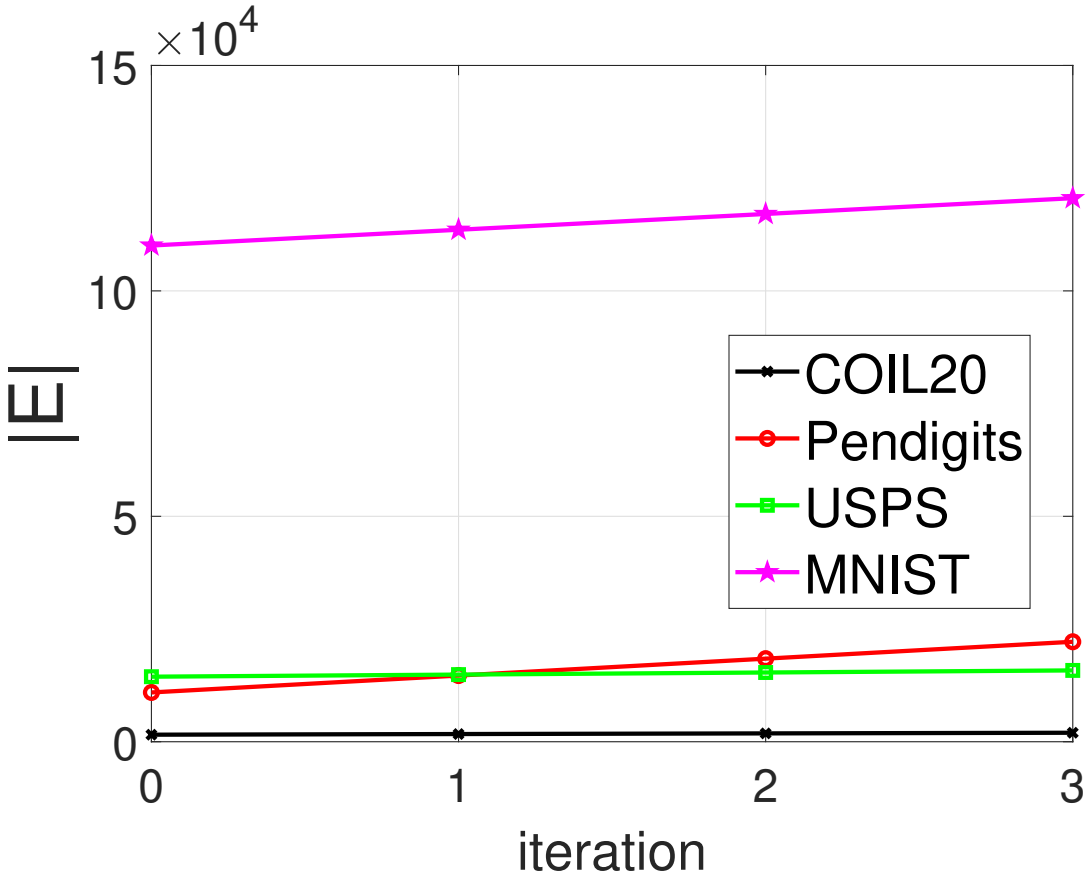


Figure 4.6: Graph complexity with increasing number of iterations

Table 4.6
Spectral clustering time results (seconds)

Data Set	Standard k-NN	Consensus	LGSS	Our method
COIL20	0.03	0.03	0.08	**0.02**
PenDigits	0.18	**0.16**	4.42	0.17
USPS	0.72	0.56	7.05	**0.28**
MNIST	252.59	-	-	**3.06**

and k-means steps in spectral clustering algorithm.

The graph density results are shown in Table 4.7.

Table 4.7
Graph density results

	$\frac{\lvert E\rvert}{\lvert V\rvert}$			
Data Set	Standard k-NN	Consensus	LGSS	Our method
COIL20	6.12	5.06	11.99	**1.39**
PenDigits	6.76	6.70	186.52	**2.96**
USPS	7.30	6.58	29.97	**1.70**
MNIST	7.46	-	-	**1.72**

Table 4.8
Graph learning (construction) time results

	runtime (seconds)		
Data Set	Consensus	LGSS	Our method
COIL20	2.43	13.56	**0.29**
PenDigits	172.51	1085.43	**2.04**
USPS	574.28	2074.78	**3.37**
MNIST	-	-	**208.89**

In Table 4.8, the run time of the consensus method is the time cost of consensus information calculation and edge pruning. The run time of our method is the time cost of edge densification. The run time of the LGSS method is the time cost of running the gsp_learn_graph_log_degrees function.

The experimental results show that the consensus method can improve the graph quality for the COIL20 and the USPS data sets. But for the Pendigits data set, it leads to an obvious clustering performance degradation. This is due to the fact that it relies on the consensus information only to prune out edges. Some useful

structural information is discarded. Even for the COIL20 and the USPS data sets, the improvement is only around 5% because it is hard to extract much useful consensus information. Therefore, the capability of using consensus information from a given k-NN graph in improving graph quality is very limited. Besides, it is very hard to find a proper threshold value for edge pruning. As shown in Table 4.4, for the COIL20 data set, the clustering performance keeps improving with the increase of the threshold value and achieves the peak performance when $\tau = 5$. However, the clustering accuracy drops more than 20% when τ increases to 6. For the USPS and the Pendigits data sets, the clustering performance drops sharply when τ increases to 3, implying that the graph is very easy to become disconnected under this graph construction strategy.

In the LGSS method, for the Pendigits data set, the unnormalized spectral clustering provides poor clustering accuracy, but the normalized spectral clustering according to [35] works well. Thus we report the clustering result based on normalized spectral clustering for Pendigits in the LGSS's results.

[27] uses a method to automatically select the parameters of the model introduced in [26] to reduce the cost of parameter tuning. For fair comparison with the standard k-NN method, in the method of [27], we set the desired edges per node on average to 10. For the COIL20 data set, the clustering accuracy is 81.16% and the NMI score is 0.93, which are not as good as the results generated by the method of [26]. So we

use the method of [26] in Table 4.5.

In our method, for the USPS data set, the mode of accuracy results is 92.27%. For the Pendigits data set, the mode of accuracy results is 84.39%. A few poor random initializations in several runs lead to a drop of the average score.

We show the effectiveness and benefits of our method in Fig. 4.7, Fig. 4.8, and Fig. 4.9.

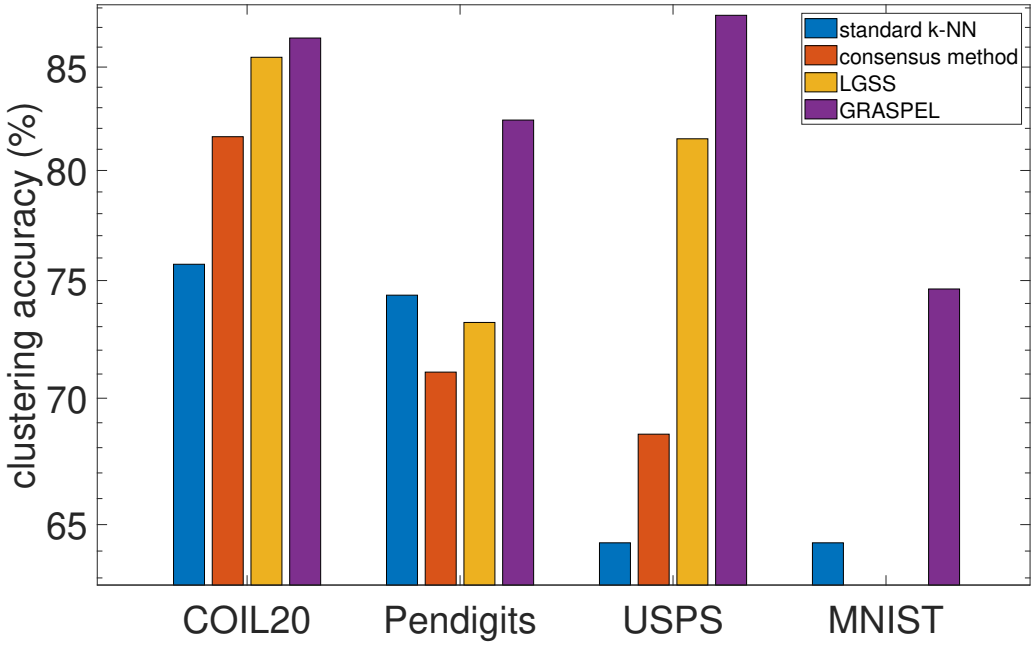


Figure 4.7: Clustering accuracy comparison

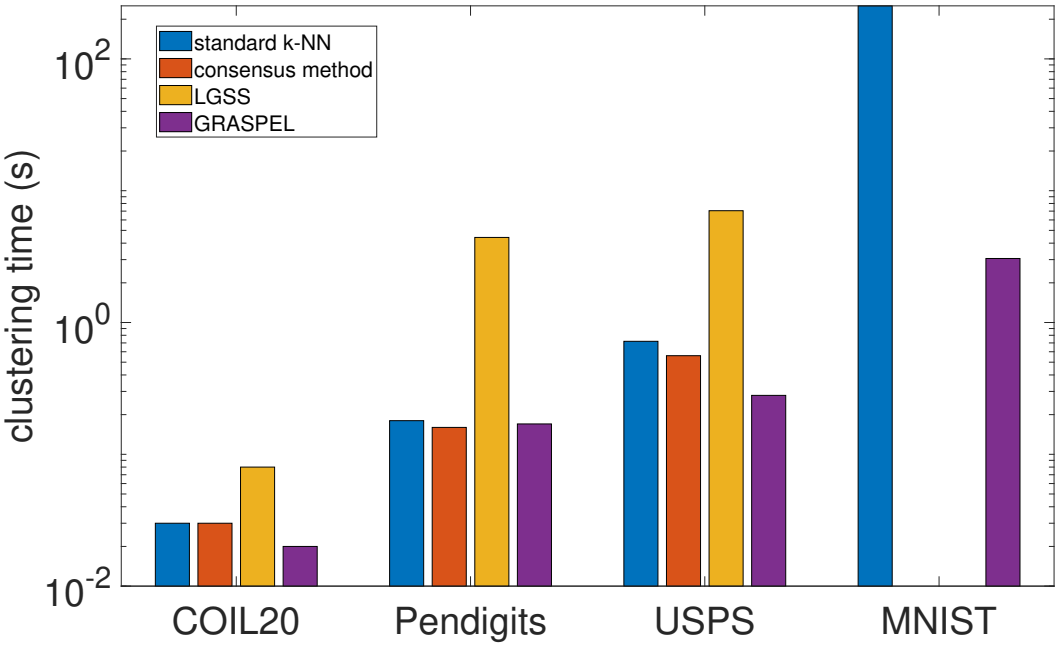


Figure 4.8: Clustering time comparison

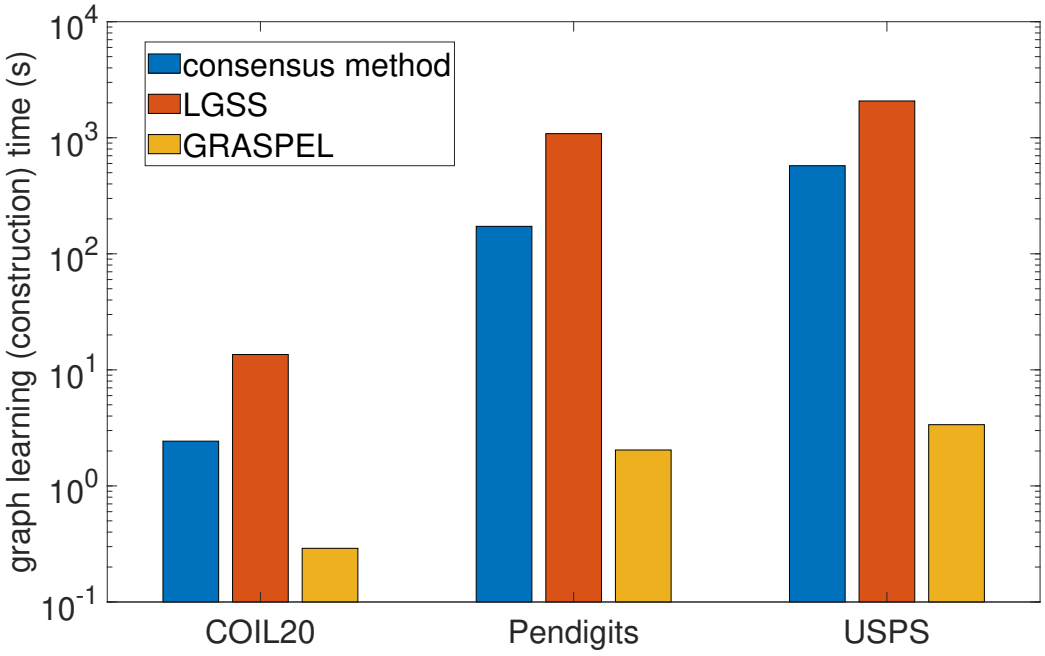


Figure 4.9: Graph learning (construction) time comparison

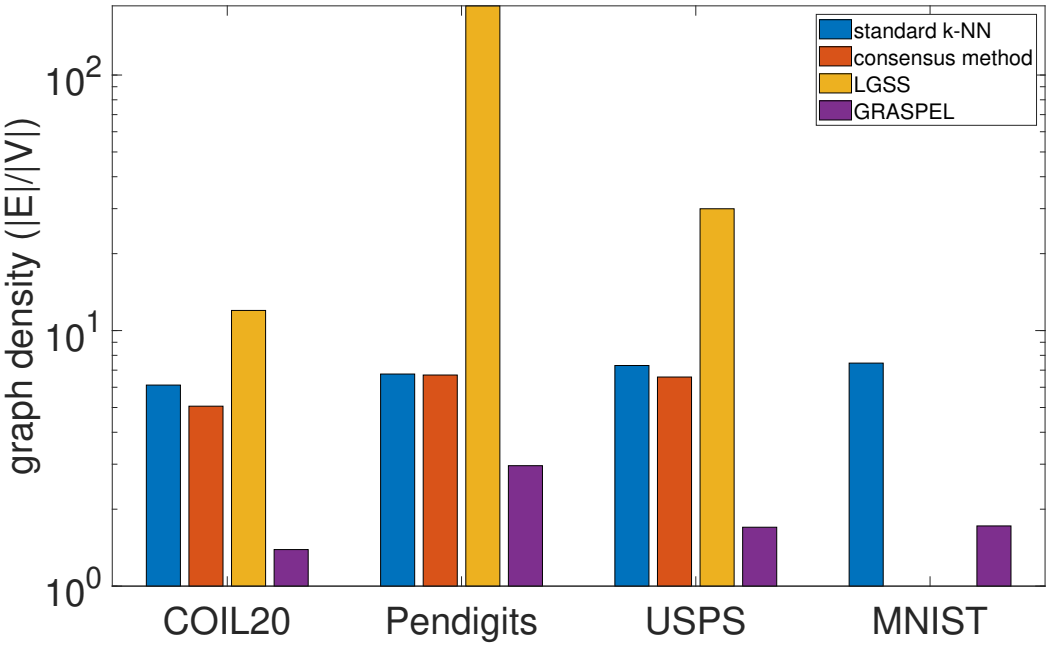


Figure 4.10: Graph density comparison

4.4 Summary

This work proposes a novel spectral graph learning approach (GRASPEL). Our method is based on iteratively identifying critical edges that have large embedding distortions to significantly improve the graph quality. The relation between the proposed method to the graphical lasso model is provided. Experiments show that the proposed method is very efficient and leads to significant improvement of spectral clustering performance.

Chapter 5

Conclusions And Future Work

5.1 Conclusions

This dissertation first provides a detailed study of the application of the spectrum-preserving spectral graph sparisification method on spectral clustering. It demonstrates the usefulness of the spectral-critical edges in improving the performance of spectral clustering. We propose a novel spectral-stability checking method that can be used as the termination criterion for both the scalable spectral clustering framework and the spectral graph learning method. We perform extensive experiments to demonstrate that the proposed framework can dramatically accelerate the computational bottleneck of spectral clustering. Among a set of state-of-the-art spectral

clustering acceleration methods, our method is the only one that can accelerate the clustering of the extremely large **Covtype** data set with $581,012$ instances without losing clustering accuracy.

This dissertation proposes a highly-scalable spectral graph learning method (GRASPEL). Our method aims to learn sparse undirected graphs from potentially high-dimensional input data. Sparse yet high quality graphs can be learned by identifying and including the edges with large spectral embedding distortion into the graph. Spectral analysis of the graphical lasso model is provided in this dissertation. By limiting the precision matrix to be a graph-Laplacian-like matrix in the graphical Lasso-based graph learning model, we show the connection between the proposed method and the log-likelihood Gaussian graphical model. Experimental results show that compared with the standard and state-of-the-art graph learning and construction methods, GRASPEL is more scalable and allows significantly improving computing efficiency and algorithm performance.

5.2 Future Work

It has been shown that the global geometric approach for manifold learning and dimension reduction provides a better representation of the data's global structure [44]. For the graph-based nonlinear dimension reduction methods such as Isomap,

there is great potential for improving their performances by using a high-quality graph.

GRASPEL also has potential to be applied in circuit design and simulation. Circuit partitioning is a fundamental problem in circuit layout [20]. GRASPEL has potential use for understanding the structure of circuits, which enables to achieve a higher quality of the layout.

References

[1] E. Agustsson, R. Timofte, and L. V. Gool. k2-means for fast and accurate large scale clustering. *arXiv preprint arXiv:1605.09299*, 2016.

[2] C. Aksoylar, J. Qian, and V. Saligrama. Clustering and community detection with imbalanced clusters. *IEEE Transactions on Signal and Information Processing over Networks*, 3(1):61–76, 2017.

[3] C. J. Alpert and S.-Z. Yao. Spectral partitioning: the more eigenvectors, the better. In *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference*, pages 195–200, 1995.

[4] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine learning research*, 9(Mar):485–516, 2008.

[5] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng. Spectral sparsification

of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013.

[6] J. Bien and R. J. Tibshirani. Sparse estimation of a covariance matrix. *Biometrika*, 98(4):807–820, 2011.

[7] T. A. Bjørklund, M. Götz, J. Gehrke, and N. Grimsmo. Workload-aware indexing for keyword search in social networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 535–544, 2011.

[8] M. E. Celebi and H. A. Kingravi. Deterministic initialization of the k-means algorithm using hierarchical clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(07):1250018, 2012.

[9] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):568–586, 2011.

[10] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. In *Twenty-fifth AAAI conference on artificial intelligence*. Citeseer, 2011.

[11] A. Choromanska, T. Jebara, H. Kim, M. Mohan, and C. Monteleoni. Fast spectral clustering via the nyström method. In *International Conference on Algorithmic Learning Theory*, pages 367–381. Springer, 2013.

[12] F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[13] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Laplacian matrix learning for smooth graph signal representation. In *2015 IEEE international conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3736–3740. IEEE, 2015.

[14] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[15] H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning from data under laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

[16] Z. Feng. Spectral graph sparsification in nearly-linear time leveraging efficient spectral perturbation analysis. In *Proceedings of the 53rd Annual Design Automation Conference*, page 57. ACM, 2016.

[17] M. Fiedler. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 25(1):57–70, 1989.

[18] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–225, 2004.

[19] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[20] J. Garbers, H. J. Promel, and A. Steger. Finding clusters in vlsi circuits. In *1990 IEEE International Conference on Computer-Aided Design*, pages 520–521. IEEE Computer Society, 1990.

[21] X. Guo. Robust subspace segmentation by simultaneously learning data representations and their affinity matrix. In *IJCAI*, pages 3547–3553, 2015.

[22] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2005.

[23] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

[24] S. Hess, W. Duivesteijn, P. Honysz, and K. Morik. The spectacl of nonconvex clustering: a spectral approach to density-based clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3788–3795, 2019.

[25] T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 441–448. ACM, 2009.

[26] V. Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pages 920–929, 2016.

[27] V. Kalofolias and N. Perraudin. Large scale graph learning from smooth signals. In *International Conference on Learning Representations*, 2019.

[28] P. Kolev and K. Mehlhorn. A note on spectral clustering. *arXiv preprint arXiv:1509.09188*, 2015.

[29] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2007.

[30] J. R. Lee, S. O. Gharan, and L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014.

[31] J. Liu, C. Wang, M. Danilevsky, and J. Han. Large-scale spectral clustering on graphs. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1486–1492. AAAI Press, 2013.

[32] Y. Liu, Q. Gao, Z. Yang, and S. Wang. Learning with adaptive neighbors for image clustering. In *IJCAI*, pages 2483–2489, 2018.

[33] M. W. Mahoney. Lecture notes on spectral graph methods. *arXiv preprint arXiv:1608.04845*, 2016.

[34] M. Maier, U. V. Luxburg, and M. Hein. Influence of graph construction on graph-based clustering measures. In *Advances in neural information processing systems*, pages 1025–1032, 2009.

[35] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[36] F. Nie, D. Xu, I. W. Tsang, and C. Zhang. Spectral embedded clustering. In *IJCAI*, pages 1181–1186, 2009.

[37] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[38] V. Premachandran and R. Kakarala. Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1594–1601, 2013.

[39] J. Qian, V. Saligrama, and M. Zhao. Graph-based learning with unbalanced clusters. *arXiv preprint arXiv:1205.1496*, 2012.

[40] M. G. Rabbat. Inferring sparse graphs from smooth signals with theoretical guarantees. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6533–6537. IEEE, 2017.

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

[42] S. E. Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[43] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[44] V. D. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in neural information processing systems*, pages 721–728, 2003.

[45] M. Slawski and M. Hein. Estimation of positive definite m-matrices and structure learning for attractive gaussian markov random fields. *Linear Algebra and its Applications*, 473:145–179, 2015.

[46] D. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Foundations of Computer Science (FOCS), 1996. Proceedings., 37th Annual Symposium on*, pages 96–105. IEEE, 1996.

[47] D. A. Spielman. Algorithms, graph theory, and linear equations in laplacian matrices. In *Proceedings of the International Congress of Mathematicians*, volume 4, pages 2698–2722, 2010.

[48] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[49] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[50] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[51] A. Szlam and X. Bresson. A total variation-based graph clustering algorithm for cheeger ratio cuts. *UCLA Cam Report*, pages 09–68, 2009.

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

[53] D. Wagner and F. Wagner. Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. Springer, 1993.

[54] J. Wang, Z. Zhang, and H. Zha. Adaptive manifold learning. In *Advances in neural information processing systems*, pages 1473–1480, 2005.

[55] Y. Wang and Z. Feng. Towards scalable spectral clustering via spectrum-preserving sparsification. *arXiv preprint arXiv:1710.04584*, 2017.

[56] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In

Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 907–916. ACM, 2009.

[57] X. Yang, L. Prasad, and L. J. Latecki. Affinity learning with diffusion on tensor product graph. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):28–38, 2012.

[58] J. Yin and J. Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 233–242, 2014.

[59] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, 2005.