ABSTRACT

LIU, NING. Spectral Clustering for Graphs and Markov Chains. (Under the direction of Dr. William J. Stewart.)

Spectral graph partitioning based on spectral theory has become a popular clustering method over the last few years. The starting point is the work of Fiedler who shows that an eigenvector of the Laplacian matrix of an undirected graph (symmetric system) provides the minimum cut of graph nodes. The spectral technique can also be applied to a Markov chain to cluster states and, in general, is more broadly applicable to nonsymmetric systems. Enlightened by these facts, we combine them to show that Markov chains, due to two different clustering techniques they offer, are effective approaches for clustering in more general situations. In this dissertation, we advance the state of the art of spectral clustering and introduce a new algorithm to decompose matrices into blocks.

We first prove that the second eigenvector of the signless Laplacian provides a heuristic solution to the NP-complete state clustering problem which is the dual problem of graph partitioning. A new method for clustering nodes of a graph that have negative edge weights is also proposed. Second, a connection between the singular vectors obtained from an SVD decomposition and the eigenvectors from spectral algorithms on data clustering is revealed. We show that the singular vectors of the node-edge incidence matrix generate not only clusters on the nodes but also clusters on the edges. Third, relating spectral clustering and state clustering of Markov chains, we present two clustering techniques for Markov chains based on two different measures and suggest a mean of incorporating both techniques to obtain comprehensive information concerning state clusters. Fourth, we display the connection between spectral clustering and dimension reduction techniques in statistical clustering. Also, the results obtained from spectral and statistical clustering are shown to be related. Finally, we develop a new improved spectral clustering procedure for decomposing matrices into blocks. This algorithm works well in several applications, especially in problems of detecting communities in complex networks, where some existing methods, e.g. MARCA and TPABLO, fail.

Spectral Clustering for Graphs and Markov Chains

by
Ning Liu

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research and Computer Science

Raleigh, North Carolina

2010

APPROVED BY:

Dr. Laurie Williams	Dr. Michael Devetsikiotis
Dr. William J. Stewart Chair of Advisory Committee	Dr. Harry G. Perros Co-Chair of Advisory Committee

BIOGRAPHY

Ning Liu was born in Beijing, the capital of China, and grew up in a big family. He is the only child of Jian Liu and Ling Wang. His father works for the government and his mother is a doctor in a hospital.

In September of 2000, Ning Liu finished his study in the middle school and started his college life in Beijing University of Aeronautics and Astronautics (Beihang University) for four years. His major is Computer Science and Technology. In July of 2004, Ning graduated with the bachelor of engineering. Then he came to US for graduate study in the University of Memphis, Tennessee. In May of 2006, Ning was awarded a master of science degree in Mechanical Engineering. Thereafter, he pursued the graduate study continuously in the Department of Operations Research and Computer Science at North Carolina State University toward his doctoral degree.

ACKNOWLEDGMENTS

I want to first express my sincere thanks to my advisor, Dr. William J. Stewart, for his guidance, time and effort he put into supervising this dissertation. Without his support, it is really hard to complete all research work involved in my Ph.D. dissertation.

I am also grateful to my advisory committee members, Dr. Harry G. Perros, Dr. Laurie Williams and Dr. Michael Devetsikiotis, for their valuable suggestions and guidance on my dissertation.

I wish to give my special thanks to Dr. Shu-Cherng Fang for the discussion of my part of research work. Also the same to Dr. Simon M. Hsiang.

As always, I enjoyed working in Department of Operations Research and Computer Science at North Carolina State University with many my best friends.

Finally, I thank my parent and my family for their continual support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Clustering is an important problem with many applications and it has always received much attention. The aim of clustering is to group objects together on the basis of a similarity measure so as to explore intrinsic data structures. There are many clustering algorithms and methods that have emerged over the years. Roughly speaking, they can be divided into two classes, namely hierarchical and partitioning [13]. In many real world systems, the data can be described as a network or graph, in which nodes represent the objects of interest and edges represent the relationships between them. In this dissertation, we mainly discuss graph partitioning techniques, which focus on clustering the nodes of a graph [1, 18].

1.1 Motivation

It is well known that graph partitioning is an NP-complete problem [14] and hence spectral graph partitioning is a heuristic approach to data clustering. Our starting point is the work of Fiedler who in the early 70's developed a spectral based partitioning method to obtain the minimum cut on an undirected graph (symmetric system) i.e., minimizing the total weight of cut edges. The eigenvector that results from the spectral decomposition, called the Fiedler vector, allows the nodes of the graph to be partitioned into two subsets. Stewart proposed a spectral method using the set of dominant eigenvectors of a Markov chain to cluster states, almost at the same time that Fiedler proposed his spectral partitioning. Stewart's result is more widely suitable for nonsymmetric systems, and it uses a

distance from the steady state of a system to measure similarity among the states. In multivariate statistical analysis, some dimension reduction approaches are available, such as the factorial method and principal components analysis, where several important eigenvectors of a multivariate dataset capture the majority of data information.

All these results are somewhat orthogonal, but they involve the eigendecomposition of a data matrix, and the eigenvectors and eigenvalues provide meaningful clustering information that helps us to group the objects in which we are interested. Due to this reason, we wonder if there is an intrinsic connection behind these approaches through the spectral property of a data structure. In fact, enlightened by the relation between results from spectral clustering, Markov chains and dimension reduction techniques in multivariate statistics, and combining them together, we have made some novel discoveries for improving spectral clustering procedures and have developed a new method to decompose matrices into blocks. In this dissertation, we shall address ourselves mainly to these problems.

1.2 Background

The starting point for applying spectral partitioning on a graph is to create a vector space model or matrix representation of the graph, e.g., the Laplacian matrix. This provides us with a matrix as an mathematical object for partitioning. Fiedler first derived well known representations of the Laplacian to show the connection to the minimum cut problem [11]. The eigenvector corresponding to the second smallest eigenvalue, called the Fiedler vector, provides a heuristic solution to the minimum cut problem. Another partitioning objective is maximum association, i.e., maximizing the total weight of edges within clusters, which is the dual problem of minimum cut. A more balanced and more popular objective is normalized cut (balanced minimum cut) [34], which has been shown to be better than minimum cut in obtaining desired partitions and can be minimized by solving the eigenvalue problem of the normalized Laplacian matrix of a graph [34, 13]. All these graph partitioning problems consider the case of graphs with nonnegative edge weight. The spectral based methods are popular because they are easy to implement. They have been useful in many applications, such as circuit layout [2], image segmentation [34] and so on.

The singular value decomposition (SVD) has also been applied to do the clustering analysis. For example, Sherriff et al. [33] use the SVD on an analysis matrix to generate

clusters of files in a *Software Change Impact Analysis*. Elsewhere, the SVD has been performed on a term-document matrix to cluster terms and documents [7, 28]. In multivariate statistical analysis, the SVD can generate lower-dimensional representations of both observations and variables from a multivariate data matrix. The benefit of the SVD is that it can be applied on a rectangular matrix rather than a square matrix. We shall take advantage of this when using spectral graph partitioning to obtain clusters of both vertices and edges.

An interesting observation from Meila et al. [24] is that the spectral clustering can be depicted in the framework of Markov random walks on a graph structure. Solving the eigenvalue problem of the transition probability matrix of a Markov random walk on a graph can achieve the normalized cut on this graph. Here the spectral clustering is viewed as states clustering of Markov chains. A well known result proposed by Stewart [35] shows that the right-hand eigenvectors belonging to the dominant eigenvalues of the transition matrix of a Markov chain provide a means of grouping the states of the chain. This clustering method is based on the distance measure of states from the steady state. We shall incorporate these two techniques on Markov chains to obtain more comprehensive information concerning clustering states.

Spectral clustering not only provides bi-partitioning result, i.e. two clusters of graph nodes, but multiple clusters as well. Roughly, there are two classes of methods available in spectral clustering for obtaining multiple clusters: (a) recursive bi-partitioning, such as the Shi and Malik (SM) [34] algorithm and the Kannan, Vempala and Vetta (KVV) [38] algorithm; (b) using multiple eigenvectors, such as the Ng, Jordan and Weiss (NJW) [27] algorithm and the Meila-Shi [25] algorithm. In most algorithms, the number of clusters is fixed and is given in advance. We shall propose a new clustering procedure that uses the information in multiple eigenvectors to do a direct multiway partitioning without losing the information of whole graph structure, and the number of clusters in our procedure is not known a priori. One possible application of our spectral clustering method is to decompose matrices into blocks, especially to find nearly-completely-decomposable (NCD) components of Markov chains. Obtaining NCD blocks of a Markov chain makes computing the stationary distribution of this chain more efficient. For this purpose, some effective methods already exist, e.g., MARCA [36] and TPABLO [3] algorithms; they are less expensive in computation and complexity than spectral clustering. However here we focus on matrix partitioning, which is a more difficult problem than calculating the stationary distribution of a Markov

chain. We shall see that our spectral method can decompose matrices into meaningful blocks in the situation of community complex networks [15] where MARCA and TPABLO cannot.

Many systems in nature have the form of networks or graphs in which set of nodes represent objects in a system and edges represent the interaction between nodes. For many networks, such as social and biological networks, it is a common property that such networks have communities in them, i.e. subsets of nodes having dense structure and strong interaction between nodes within them, but weak interaction between subsets themselves [15]. This property tells us that detecting the community structure in a network is similar to clustering nodes of a graph. Since networks in real world can be represented by graphs with dynamic interactions on them, we may apply the clustering techniques of Markov chains to understand the community structure in complex networks.

1.3 Contributions

In this dissertation, we first introduce the signless Laplacian matrix [6] to model the maximum association in spectral clustering, and show that the eigenvalues of the signless Laplacian have important physical interpretations. Furthermore, we prove that the normalized cut can also be minimized by solving the eigenvalue problem of the normalized signless Laplacian matrix. In the case of a graph having negative edge weights, a matrix constructed in the same way as the signless Laplacian can be used to obtain meaningful clusters of nodes by computing the eigenvector corresponding to the largest eigenvalue.

Second, we show the relation between singular vectors used in clustering and eigenvectors in spectral clustering. This connection indicates that the clustering from an SVD is based on a graph partitioning. Considering features of the SVD, we develop a new approach of using the SVD technique on the node-edge incidence matrix of a graph, using the signless Laplacian as a bridge, to obtain both node and edge clusters. We also show that this idea can be extended to the generalized incidence structure matrix of two classes of states.

Third, we present two clustering measures based on two clustering techniques of the states of Markov chains, namely, (a) the normalized cut measure and (b) the distance measure of states from the steady state. We mainly discuss the clustering information of graph nodes provided by the second clustering measure (b). This allows us to present

a novel result concerning clustering on graph nodes that is based on incorporating both clustering techniques in Markov chains.

Fourth, we display the relation between spectral clustering and dimension reduction techniques of multivariate statistical analysis, e.g., the factorial method and principal components analysis. Spectral clustering is a spectral case of the factorial method on node-edge incidence matrix. We also present a way of using the graph partitioning technique to analyze a multivariate data matrix, and this result is related to the result obtained from principal components analysis.

Fifth, we propose a heuristic method to determine the number of clusters in a graph based on the number of eigenvectors we choose to partition graph nodes. The choice of important eigenvectors for clustering is controlled by the input of two parameters. Then we develop a new regular spectral clustering algorithm. For large-scale data, a novel hierarchical algorithm is presented. Our spectral procedure works well in detecting communities in real-world networks.

1.4 Organization

This dissertation is organized as follows. Chapter 2 reviews some preliminary mathematics including graph theory, matrix analysis, Markov chains, multivariate analysis and some numerical methods. Chapter 3 describes procedures of spectral clustering using the Laplacian and signless Laplacian matrices, and a method of clustering nodes in the case of a graph having negative edge weights. The SVD technique used to obtain both vertex clusters and edge clusters is also discussed in this chapter. Chapter 4 talks about constructing a transition probability matrix of a random walk from a graph for spectral clustering, and two clustering measures on the states of Markov chains associated with two clustering techniques. The relation between spectral clustering and dimension reduction techniques in multivariate statistical analysis is discussed in Chapter 5. Some heuristic methods for decomposing real unsymmetric matrices including MARCA, TPABLO and our new spectral clustering procedures are presented in Chapter 6. In Chapter 7, we demonstrate some applications and experiment results. One important application is studying communities within complex networks. Finally, Chapter 8 provides the conclusions, and outlines future research.

Chapter 2

Preliminary Mathematics

In this chapter, we introduce some preliminary mathematics including graph theory, matrix analysis and linear algebra, Markov chains, multivariate statistics, and some numerical methods, which are helpful for this dissertation.

2.1 Graph Theory

Many practical problems in this world involve graph theory, which is studied in mathematics and computer science. A graph is a mathematical structure which displays pairwise relations between objects of interest.

Definition 2.1.1 (**Graph**). *A graph G consists of three sets: a set of vertices $V(G)$, a set of edges $E(G)$ and a set of relations $R(G)$. A relation associates an edge with two vertices, called its endpoints. Two vertices are adjacent if they are the endpoints of an edge.*

Definition 2.1.2 (**Simple Graph**). *A loop is an edge having the same endpoints. Multiple edges are edges sharing the same pair of endpoints. A simple graph is a graph without loops and multiple edges.*

For a simple graph, we treat each edge in its edge set as a pair of endpoints (vertices). Thus we use a shorter notation $G = (V, E)$ to represent a simple graph, where V is the vertex set and E is the edge set. Unless otherwise stated, all graphs discussed in this dissertation are simple graphs.

Example 2.1.1. Some basic graphs are shown below:

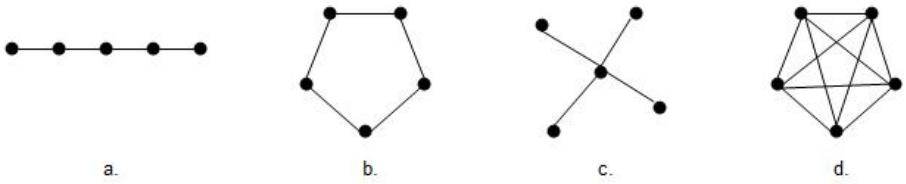

Figure 2.1: Basic graphs with five vertices: a) a path, b) a cycle, c) a star, d) a complete graph.

A complete graph is a simple graph in which every two vertices are adjacent. A clique in a graph is a vertex set in which vertices are pairwise adjacent.

Definition 2.1.3. *If a vertex is an endpoint of an edge, then this vertex and this edge are* **incident**. *In a graph without loops, the* **degree** *of a vertex is the number of incident edges.*

Definition 2.1.4 (**Subgraph**). *A subgraph S of a graph G has a vertex set $V(S) \subseteq V(G)$, a edge set $E(S) \subseteq E(G)$ and a relation set $R(S) \subseteq R(G)$. An induced subgraph H of a graph G is a subgraph obtained by deleting a set of vertices of G.*

The induced subgraph H of G contains the vertex set $V(H)$ and all edges whose endpoints are included in $V(H)$.

Definition 2.1.5 (**Connected Graph**). *A graph G is connected if each pair of vertices in G is connected by a path; otherwise G is disconnected.*

Definition 2.1.6 (**Line Graph**). *The line graph of a graph G, denoted $L(G)$, is a graph whose vertices correspond to the edges of G; two vertices are adjacent in $L(G)$ if and only if their corresponding edges in G share a common endpoint.*

An example of a graph G and its line graph $L(G)$ is shown in Figure 2.2.

Definition 2.1.7 (**Weighted Graph**). *A weighted graph is a graph whose edges are associated with numerical labels.*

An unweighted graph can also be seen as a graph with numerical labels 1 on each edge. Many practical problems are modeled as weighted graphs. Spectral clustering techniques partition the graph nodes of weighted graphs; in general only nonnegative edge

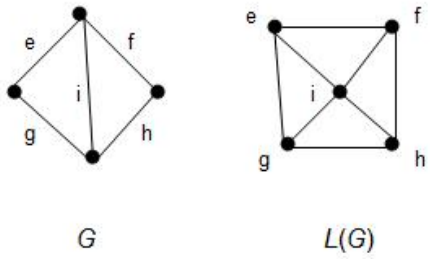

Figure 2.2: A graph G and its line graph $L(G)$.

weights are considered. However numerical labels on the edges may be negative. In Section 3.4, we shall discuss spectral methods that can cluster nodes of a graph with negative edge weights.

Definition 2.1.8 (**Directed Graph).** *A directed graph or digraph G, consists of a set of vertices $V(G)$, a set of edges $E(G)$ and a function assigning each edge an ordered pair of vertices. The first vertex in this ordered pair is called the tail of the edge, and the second is called the head. An edge is from its tail to its head.*

Definition 2.1.9 (**Simple Digraph).** *A loop in a digraph is an edge whose tail and head are the same. Multiple edges are edges who have been assigned the same ordered pair of endpoints. A simple digraph is a digraph with no multiple edges and at most one loop per vertex.*

From this definition, each edge in a simple digraph can be identified by its ordered pair of endpoints.

Example 2.1.2. Figure 2.3 shows two basic digraphs: a path and a cycle.

In graph theory, many concepts involve the words "maximal" and "maximum". The difference between these two words is that, **maximum** means "largest size", while **maximal** means "no larger one includes this one". For example, a maximum path is also a maximal path, however a maximal path is not necessarily a maximum one.

Definition 2.1.10 (**Weakly and Strongly Connected).** *A digraph G is weakly connected if its underlying graph (get rid of direction from each edge) is connected. A digraph G is*

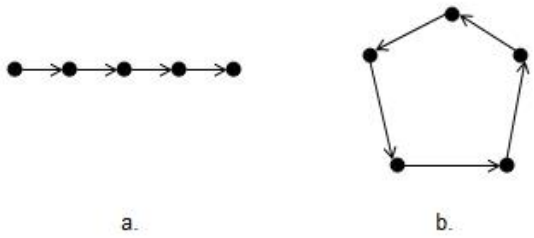

Figure 2.3: Basic digraphs: a) a path and b) a cycle.

strongly connected if for each ordered pair of vertices s and t in G, there is a path from s to t.

Definition 2.1.11 (Strongly Connected Component). *The strongly connected components (SCC) in a digraph G are its maximal strongly connected subgraphs.*

The concept of strongly connected components is used in the algorithm MARCA, a method for finding blocks structure of matrices, which is described in Section 6.2.

2.2 Matrix Analysis and Linear Algebra

This section is a review of some basic concepts from matrix analysis and linear algebra, which will be widely used in this dissertation.

2.2.1 Matrix Basics and Vectors

A matrix is a rectangular array of scalars. For example, a matrix A with m rows and n columns is

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix},$$

where a_{ij} is a scalar in the position of row i and column j. A submatrix of a given matrix A is a matrix obtained by deleting some rows and columns from A. Matrices having only one single row or one single column are called row vectors or column vectors, respectively.

Example 2.2.1. Some special matrices and vectors are listed in Table 2.1

Table 2.1: Special matrices and vectors.

Matrix $A_{m\times n}$	Definition
column vector	$n = 1$
row vector	$m = 1$
vector of ones	$(1, \ldots, 1)^T$
square matrix	$m = n$
diagonal matrix	$a_{ij} = 0$ for $i \neq j$, and $m = n$
identity matrix I	diagonal matrix with $a_{ii} = 1$ for all i
symmetric matrix	$a_{ij} = a_{ji}$
orthogonal matrix	$A^T A = AA^T = I$

Definition 2.2.1 (**Rank**). *A set of vectors $v_1, \ldots, v_n$ is linearly independent if the only solution of the scalars c_i in the equation*

$$c_1 v_1 + c_2 v_2 + \cdots + c_n v_n = 0$$

is the trivial solution $c_1 = c_2 = \cdots = c_n = 0$. The rank of a matrix A, denoted as rank(A), is the maximum number of linearly independent rows (columns) in A.

Given an $m \times n$ matrix A, the following statements about matrix rank are true.

- $\text{rank}(A) \leq \min(m, n)$.
- $\text{rank}(A) = \text{rank}(A^T)$.
- $\text{rank}(A^T A) = \text{rank}(A) = \text{rank}(AA^T)$.

Definition 2.2.2 (**Trace**). *The trace of a matrix $A_{n\times n}$ is defined to be the sum of its diagonal elements*

$$\text{trace}(A) = \sum_{i=1}^{n} a_{ii}. \tag{2.1}$$

Definition 2.2.3 (**Euclidean Vector Norm**). *For a real vector $x_{n\times 1}$, the Euclidean norm of x is*

$$\|x\| = (\sum_{i=1}^{n} {x_i}^2)^{1/2} = (x^T x)^{1/2}, \tag{2.2}$$

and this is also called 2-norms, written as $\|x\|_2$.

Definition 2.2.4 (**Angles**). *For two real nonzero vectors $x_{n\times 1}$ and $y_{n\times 1}$, the angle $\theta \in [0, \pi]$ between x and y is defined by the cosine:*

$$\cos\theta = \frac{x^T y}{\|x\|\|y\|}. \tag{2.3}$$

2.2.2 Eigenvalues and Eigenvectors

Definition 2.2.5 (**Eigenvalues and Eigenvectors**). *For a square matrix $A_{n\times n}$, if there are scalars λ and vectors $v_{n\times 1} \neq 0$ satisfying the equation*

$$Av = \lambda v, \tag{2.4}$$

then λ and v are called eigenvalues and eigenvectors of A, respectively. The set of distinct eigenvalues is called the spectrum of A.

Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of $A_{n\times n}$, then the following equation is true.

$$\text{trace}(A) = \sum_{i=1}^{n} \lambda_i. \tag{2.5}$$

A generalized eigenvalue problem of a square matrix $A_{n\times n}$ is given by

$$Av = \lambda Bv, \tag{2.6}$$

where B is an $n \times n$ matrix, and λ and v are generalized eigenvalues and eigenvectors of A with respect to B.

Definition 2.2.6 (**Similarity**). *Two $n \times n$ matrices A and B are similar if there exists a nonsingular matrix P such that*

$$P^{-1}AP = B. \tag{2.7}$$

Theorem 2.2.1. *Similar matrices have the same characteristic polynomial, so they have the same eigenvalues.*

Theorem 2.2.2 (**Jordan Decomposition**). *A real symmetric matrix $A_{n\times n}$ has real eigenvalues and a complete set of n orthogonal eigenvectors. This symmetric matrix A is orthogonally similar to a real-diagonal matrix D, such that*

$$P^T AP = D \tag{2.8}$$

with P orthogonal, where the columns of P are the orthogonal eigenvectors of A, and the diagonal elements of D are the corresponding eigenvalues.

2.2.3 Singular Value Decomposition

Theorem 2.2.3 (**Singular Value Decomposition**). *For each $A_{m\times n}$ of rank r, there are orthogonal matrices $U_{m\times m}, V_{n\times n}$ and a diagonal matrix $D_{r\times r} = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$ such that*

$$A = U \begin{pmatrix} D_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}_{m\times n} V^T \quad \text{with} \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0.$$

The σ_i's are called the nonzero singular values of A. The columns in U and V are called left-hand and right-hand singular vectors for A, respectively.

The following statements are true.

- The nonzero eigenvalues of A^TA and AA^T are equal and positive.
- The nonzero singular values of A are the positive square roots of the nonzero eigenvalues of A^TA and AA^T.
- If A is symmetric with nonzero eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_r\}$, then the nonzero sigular values of A are $\{|\lambda_1|, |\lambda_2|, \ldots, |\lambda_r|\}$.
- The right-hand and left-hand singular vectors of A are the orthogonal eigenvectors of A^TA and AA^T, respectively.

2.2.4 Quadratic Forms

A quadratic form of a symmetric matrix $A_{n\times n}$ is x^TAx, where $x_{n\times 1}$ is a vector. The term x^TAx/x^Tx is called a Rayleigh quotient.

Theorem 2.2.4. *The eigenvalues of a real symmetric matrix A are nonnegative if and only if A can be factored into the form of $A = B^TB$, where B is a square matrix; and A has all positive eigenvalues if and only if B is nonsingular.*

Definition 2.2.7 (**Definiteness**). *A symmetric matrix A is called positive definite if its eigenvalues are positive, and A is positive semidefinite if its eigenvalues are just nonnegative.*

There is another way to define the definiteness of matrices. A matrix A is positive definite if the corresponding quadratic form $x^TAx > 0$ for all $x \neq 0$, and A is positive semidefinite if the corresponding quadratic form $x^TAx \geq 0$ for all $x \neq 0$.

Theorem 2.2.5. *For real symmetric matrices A and B with B positive definite, the maximum and minimum of x^TAx/x^TBx are given by the largest and smallest eigenvalues of $B^{-1}A$, respectively, such that*

$$\lambda_1 \leq \frac{x^TAx}{x^TBx} \leq \lambda_n, \quad \textit{for all } x \neq 0, \tag{2.9}$$

where $\lambda_1 \leq \cdots \leq \lambda_n$ are the eigenvalues of $B^{-1}A$. The vector x that maximizes (minimizes) x^TAx/x^TBx is the eigenvector of $B^{-1}A$ corresponding to the largest (smallest) eigenvalue. If $B = I$ the identity matrix, then the Rayleigh quotient satisfies

$$\lambda_1 \leq \frac{x^TAx}{x^Tx} \leq \lambda_n, \quad \textit{for all } x \neq 0,$$

Theorem 2.2.6 (Courant-Fischer Theorem). *Let V^k denote a k dimensional subspace of R^n and $x \perp V^k$ meaning that $x \perp y$ for all $y \in V^k$. Then the eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ of a symmetric matrix $A_{n\times n}$ are*

$$\lambda_i = \max_{V^{i-1}} \Big(\min_{\substack{x\perp V^{i-1} \\ x\neq 0}} \frac{x^TAx}{x^Tx} \Big) \quad \text{and} \quad \lambda_i = \min_{V^{n-i}} \Big(\max_{\substack{x\perp V^{n-i} \\ x\neq 0}} \frac{x^TAx}{x^Tx} \Big) \tag{2.10}$$

2.2.5 Nonnegative Matrices

Definition 2.2.8 (Nonnegative Matrix). *A real matrix A is said to be a nonnegative matrix of real-valued elements if each element $a_{ij} \geq 0$. This is denoted as $A \geq 0$. Similarly, A is said to be positive if each element $a_{ij} > 0$, written as $A > 0$.*

Definition 2.2.9 (Reducibility). *An $n \times n$ matrix A is a reducible matrix if the following equation is valid.*

$$P^TAP = \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}, \tag{2.11}$$

where P is a permutation matrix, and X and Z are both square. Otherwise A is an irreducible matrix.

For nonnegative matrices, a primary result concerns the relation between the properties of $A \geq 0$ and their spectral properties is called the Perron-Frobenius theory. Here we present one part of Perron-Frobenius theorem which is used later in this dissertation.

Theorem 2.2.7 (**Perron-Frobenius Theorem**). *Let $A_{n\times n} \geq 0$ be an irreducible matrix. Then*

- *A has a real positive eigenvalue λ_1, which is equal to its spectral radius.*
- *The spectral radius λ_1 is a simple eigenvalue of A.*
- *The unique eigenvector v of A corresponding to λ_1 is positive, i.e.,*

$$Av = \lambda_1 v, \quad v > 0.$$

There are no other nonnegative eigenvectors for A.

Nonnegative irreducible matrices can be divided into two important classes based on the property of having only one or more eigenvalues on the spectral circle.

Definition 2.2.10 (**Primitive Matrix**). *A nonnegative irreducible matrix A is a primitive matrix if it has only one eigenvalue on its spectral circle. If A has more than one eigenvalues on its spectral circle, then A is not primitive.*

This concept of primitive and non-primitive matrices is related to the periodicity of Markov chains described in the next section.

2.3 Markov Chains

In this section, we first introduce the concept of the Markov process which is one of the most commonly used stochastic processes.

Definition 2.3.1 (**Markov Process**). *A Markov process is a stochastic process that has the Markov property. This is, let $X(t)$ be a continuous-time stochastic process that denotes the state of a system at time t, then for all integers n and for any time sequence $t_0, t_1, \ldots, t_n, t_{n+1}$ with $t_0 < t_1 < \cdots < t_n < t_{n+1}$, we have*

$$\begin{aligned}&\text{Prob}\{X(t_{n+1}) \leq x_{n+1} | X(t_0) = x_0, X(t_1) = x_1, \ldots, X(t_n) = x_n\} \\ &= \text{Prob}\{X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n\}.\end{aligned}$$

This equation of conditional probabilities shows that the next state of the system depends only on its current state and not on its past history. This memoryless property is called the Markov property. If the transitions of $X(t)$ are independent of time, then the Markov process is said to be homogeneous.

When the state space of a Markov process is discrete, the process is called a Markov chain. There are two types of Markov chains, discrete-time Markov chains (DTMC) and continuous-time Markov chains (CTMC).

2.3.1 Discrete-Time Markov Chains (DTMC)

Definition 2.3.2 (**Discrete-Time Markov Chain**). *For all numbers n and all states x_n, a Markov chain X_n is a discrete-time Markov chain if*

$$\text{Prob}\{X_{n+1} = x_{n+1} | X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n\}$$

$$= \text{Prob}\{X_{n+1} = x_{n+1} | X_n = x_n\}.$$

In a DTMC, the states of a system are observed at a discrete set of times. When it is a homogeneous DTMC, the transition probabilities are independent of the step n. We can write the transition probability p_{ij} from state i to state j as

$$p_{ij} = \text{Prob}\{X_{n+1} = j | X_n = i\}, \quad \text{for all} \quad n = 0, 1, \ldots.$$

Definition 2.3.3 (**Transition Probability Matrix**). *A transition probability matrix P is a square matrix formed by placing transition probability p_{ij} in row i and column j, for all i and j.*

The elements of the matrix P have two properties:

- $0 \leq p_{ij} \leq 1$;
- $\sum_j p_{ij} = 1$ for all i.

A matrix satisfying these properties is also said to be a stochastic matrix.

Now let us consider the probability that a homogeneous DTMC is in a given state at a certain step. We use $\pi_i(n)$ to represent the probability that a DTMC is in state i at step n, i.e.,

$$\pi_i(n) = \text{Prob}\{X_n = i\}.$$

Thus, a row vector $\pi(n) = (\pi_1(n), \pi_2(n), \dots, \pi_i(n), \dots)$ gives the probability distribution on the states of a DTMC chain at step n, and $\pi(0)$ denotes the initial probability distribution on the states.

Definition 2.3.4 (**Stationary Distribution**). *Given the transition probability matrix P of a DTMC, a real row vector z is a stationary distribution if and only if $zP = z$, where z gives a probability distribution on states such that the element z_i of z is the probability of being in state i, i.e.,*

$$0 \leq z_i \leq 1 \quad \textit{and} \quad \sum_i z_i = 1.$$

Hence

$$z = zP = zP^2 = \cdots = zP^n = \cdots .$$

If an initial probability distribution is $\pi(0) = z$, then for all n, we have $\pi(n) = z$.

Definition 2.3.5 (**Limiting Distribution**). *Let P be the transition probability matrix of a homogenous DTMC, if the limit $\lim_{n\to\infty} P^n$ exists, then the probability distribution $\pi = \lim_{n\to\infty} \pi(n)$ also exists, since*

$$\pi = \lim_{n\to\infty} \pi(n) = \lim_{n\to\infty} \pi(0)P^n = \pi(0) \lim_{n\to\infty} P^n,$$

where $\pi(0)$ is an initial probability distribution. This limit π is said to be a limiting distribution of the Markov chain.

The stationary distribution does not always mean the limiting distribution of a Markov chain. A finite, irreducible Markov chain has a unique stationary distribution z, but it may not have a limiting distribution. In order to ensure the existence and uniqueness of the limiting distribution, the additional property of aperiodic is needed.

Definition 2.3.6 (**Periodicity**). *An irreducible DTMC is periodic if the number of transition steps required to return to a state i when starting from this state i (for all i) is a multiple of some integer $p > 1$. The smallest p is called the period of the chain. If $p = 1$, then the irreducible DTMC is aperiodic.*

The limiting distribution of a finite, irreducible and aperiodic (ergodic) Markov chain always exists and is unique. Furthermore, it is independent of the initial probability distribution. If a Markov chain is periodic, then the limit π does not exist and this chain does not have a limiting distribution.

2.3.2 Continuous-Time Markov Chains (CTMC)

Definition 2.3.7 (**Continuous-Time Markov Chain**). *For all integers n and any time sequence $t_0, t_1, \ldots, t_n, t_{n+1}$ with $t_0 < t_1 < \cdots < t_n < t_{n+1}$, a stochastic process $\{X(t), t \geq 0\}$ is a continuous-time Markov chain if*

$$\text{Prob}\{X(t_{n+1}) = x_{n+1} | X(t_0) = x_0, X(t_1) = x_1, \ldots, X(t_n) = x_n\}$$

$$= \text{Prob}\{X(t_{n+1}) = x_{n+1} | X(t_n) = x_n\}.$$

The states of a CTMC are discrete, however this chain may change state at any point of time.

For continuous-time Markov chains, an infinitesimal generator matrix Q is used to represent its transitions. The element $q_{ij}, i \neq j$ of Q is the rate at which the transitions occur from state i to state j, and the diagonal element q_{ii} is equal to $-\sum_{j \neq i} q_{ij}$. Therefore the matrix Q satisfies

$$\sum_j p_{ij} = 0, \quad \text{for all } i.$$

A finite, irreducible CTMC has a limiting distribution which is identical to the stationary distribution of the Markov chain. The stationary distribution vector z can be obtained by solving the linear system of equations

$$zQ = 0 \quad \text{and} \quad \sum_i z_i = 1.$$

The infinitesimal generator matrix Q of a CTMC can also be transformed to a stochastic matrix P by using

$$P = Q\Delta t + I,$$

where I is the identity matrix and Δt can be any value satisfying

$$0 < \Delta t \leq \frac{1}{\max_i |q_{ii}|}.$$

The stationary distribution vector z of this CTMC can also be determined by solving

$$zP = z \quad \text{and} \quad \sum_i z_i = 1.$$

2.3.3 Some Properties of Stochastic Matrices

A stochastic matrix $P_{n\times n}$ has the following properties:

- Every stochastic matrix P has a unit eigenvalue that is on its spectral circle.
- The modulus of eigenvalues of P must less than or equal to 1.
- If P represents an irreducible Markov chain, then it has a simple unit eigenvalue.
- The vector $(1, 1, \ldots, 1)^T$ is a right-hand eigenvector of P corresponding to a unit eigenvalue 1.
- A vector z is a stationary distribution vector of P if and only if it is a left-hand eigenvector corresponding to a unit eigenvalue 1.

There is a result concerning the relation between the property of a stochastic matrix and the periodicity of a Markov chain.

Result 2.3.1 ([26]). *For an irreducible Markov chain, if its transition probability matrix P is primitive, i.e., it has only one unit eigenvalue, then this Markov chain is aperiodic; otherwise if P is non-primitive, i.e., there is more than one eigenvalue on the unit circle, then this chain is periodic and the period is the number of eigenvalues on the unit circle of P.*

Therefore we can determine the periodicity of a Markov chain based on the eigenvalues of its transition probability matrix.

2.4 Multivariate Statistics

Covariance and Correlation

The covariance is a measure of dependency between two random variables.

Definition 2.4.1 (**Covariance**). *Given two real random variables X and Y, the covariance of X and Y is defined*

$$\sigma_{XY} = \mathrm{Cov}(X, Y) = E(XY) - E(X)E(Y), \tag{2.12}$$

where $E(X)$ is the expectation of the random variable X.

If X and Y are independent, then we have $E(XY) = E(X)E(Y)$. This is followed by $\mathrm{Cov}(X, Y) = 0$. If X and Y are identical, then the covariance becomes the variance

$$\sigma_X^2 = \mathrm{Var}(X) = \mathrm{Cov}(X, X) = E(X^2) - E(X)^2.$$

The covariance is dependent of the scale of measurement. If the variables are measured in a different scale, then the value of the covariance between them will change. For this reason, the concept of correlation is introduced.

Definition 2.4.2 (**Correlation**). *The correlation of two random variables X and Y is defined as*

$$\rho_{XY} = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}, \tag{2.13}$$

where σ_X and σ_Y are the standard deviations of X and Y, respectively.

The correlation is independent of the scale, and its value ρ_{XY} lies between -1 and $+1$. The correlation is zero if and only if the covariance is zero, and in this case, these two variables are uncorrelated. If X and Y are identical, then it is obvious that $\rho_{XX} = 1$.

Now let us move to a random variable X to higher dimensions, i.e., a multivariate random variable. If X is p-dimensional multivariate, then it is composed of p one-dimensional random variables:

$$X = (X_1, X_2, \ldots, X_p).$$

The expectation of the multivariate variable X is given by the vector

$$EX = (EX_1, EX_2, \ldots, EX_p).$$

Definition 2.4.3 (**Covariance Matrix**). *The covariance matrix Σ of a p-dimensional multivariate variable X is given by*

$$\Sigma = \begin{pmatrix} \sigma_{X_1}^2 & \sigma_{X_1X_2} & \cdots & \sigma_{X_1X_p} \\ \sigma_{X_2X_1} & \sigma_{X_2}^2 & \cdots & \sigma_{X_2X_p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{X_pX_1} & \sigma_{X_pX_2} & \cdots & \sigma_{X_p}^2 \end{pmatrix}, \tag{2.14}$$

where $\sigma_{X_i}^2 = \mathrm{Var}(X_i)$ and $\sigma_{X_iX_j} = \mathrm{Cov}(X_i, X_j)$ for $i \neq j$.

The covariance matrix of X provides the variance of X. The reason is as follows:

$$\begin{aligned}
\mathrm{Var}(X) &= \mathrm{Cov}(X, X) \\
&= E[(X - EX)^T (X - EX)] \\
&= E[(X_1 - EX_1, \ldots, X_p - EX_p)^T (X_1 - EX_1, \ldots, X_p - EX_p)] \\
&= \begin{pmatrix} \sigma^2_{X_1} & \sigma_{X_1X_2} & \cdots & \sigma_{X_1X_p} \\ \sigma_{X_2X_1} & \sigma^2_{X_2} & \cdots & \sigma_{X_2X_p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{X_pX_1} & \sigma_{X_pX_2} & \cdots & \sigma^2_{X_p} \end{pmatrix}.
\end{aligned} \tag{2.15}$$

It is not difficult to see that

$$\Sigma = E(X^T X) - E(X)^T E(X). \tag{2.16}$$

Definition 2.4.4 (Correlation Matrix). *The correlation matrix $\mathcal{P}$ of a p-dimensional multivariate variable X is defined as*

$$\mathcal{P} = \begin{pmatrix} 1 & \rho_{X_1X_2} & \cdots & \rho_{X_1X_p} \\ \rho_{X_2X_1} & 1 & \cdots & \rho_{X_2X_p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{X_pX_1} & \rho_{X_pX_2} & \cdots & 1 \end{pmatrix}, \tag{2.17}$$

where $\rho_{X_iX_j}$ is the correlation between X_i and X_j for $i \neq j$.

We can see that the correlation matrix $\mathcal{P}$ can be computed by dividing the ith row of the covariance matrix Σ by σ_{X_i}, and dividing the jth column of Σ by σ_{X_j}.

The K-means Algorithm

The K-means algorithm is an efficient clustering method for partitioning n individuals with high dimensions into k clusters. Each individual belongs to the cluster which has the nearest centroid (mean).

Suppose there are n individuals $x_i \in \mathcal{R}^p$ for $i = 1, \ldots, n$, and we want to group them into k disjoint subsets $S_j, j = 1, \ldots, k$, so as to minimize the sum of squares:

$$\sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - \mu_j\|_2^2, \tag{2.18}$$

where μ_j is the centroid of cluster S_j. The centroid μ_j is given by

$$\mu_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i, \tag{2.19}$$

where $|S_j|$ is the number of individuals within cluster S_j.

The standard K-means algorithm consists of the following steps:

1. Specify the initial k cluster centroids $\mu_j, j = 1, \ldots, k$ which may be selected randomly or may be obtained by some heuristics.

2. Assign each individual x_i to a cluster whose centroid is closest to it:

$$\arg\min_j \|x_i - \mu_j\|_2^2.$$

   Assign x_i to cluster S_j.

3. Compute the new k cluster centroids μ_j by Equation (2.19).

4. Repeat step 2 and 3 until there is no further change of clusters.

The assignment of individuals in step 2 will reduce the value of the sum of squares in Equation (2.18), and this algorithm will finally converge.

The K-means algorithm can only converge to a local minimum. To find the global minimum, it is better to run this algorithm procedure multiple times by using different initial random cluster centroids and then choosing the lowest value of Equation (2.18).

2.5 Numerical Methods for Solving Eigenvalue Problems

In this section we introduce some numerical methods for obtaining a set of dominant eigenvalues and the corresponding eigenvectors of a large matrix. The most efficient methods are called projection methods; they consist of extracting approximations of the exact eigensolutions from small-dimensional subspaces. There are mainly two objects that need to be considered, namely,

- A small-dimensional subspace $\mathcal{K}$ from which the approximations are extracted,
- Another subspace $\mathcal{L}$ that sets constraints of generating the approximations from $\mathcal{K}$ (orthogonality constraints are usually used).

Projection Processes for Eigenvalue Problems

For the eigenvalue problem of a matrix $P_{n\times n}$

$$Px = \lambda x,$$

we seek to obtain an approximation of λ and an approximation of $x \in \mathcal{K}$ with orthogonality constraints such that the residual $Px - \lambda x$ is orthogonal to a subspace $\mathcal{L}$. Let $V = (v_1, \ldots, v_m)$ and $W = (w_1, \ldots, w_m)$ be the bases of subspaces $\mathcal{K}$ and $\mathcal{L}$, respectively ($m \ll n$), and letting $x = Vy$, we have

$$W^T(PVy - \lambda Vy) = 0$$

or

$$W^T PVy = \lambda W^T Vy. \tag{2.20}$$

If $W^T V$ is close to the identity matrix, then the approximate eigenvalues of P are the eigenvalues of the $m \times m$ matrix $W^T PV$. The corresponding approximations of the eigenvectors are $x = Vy$. Now we obtain m approximate eigenpairs of P. When the subspaces $\mathcal{K} = \mathcal{L}$ and V is an orthogonal basis of $\mathcal{K}$, this projection process is said to be an orthogonal projection process.

A Lopsided Simultaneous Iteration Algorithm (LOPSI)

The Lopsided simultaneous iteration algorithm is used to obtain one set of eigenvalues of largest modulus of a large sparse matrix and their corresponding left-hand or right-hand eigenvectors. Each iteration of the simultaneous iteration method is carried out with some trail vectors that converge to the eigenvectors associated with the dominant eigenvalues of the matrix.

Let $P_{n\times n}$ be a real nonsymmetric matrix for which the right-hand eigenvectors corresponding to the dominant eigenvalues are required. The Lopsided iteration algorithm consists of a following sequence of steps:

1. Choose a set of m initial normalized trail vectors $U = (u_1, u_2, \ldots, u_m)$.
2. Premultiply U by P to obtain the resulting set of vectors V as $V = PU$.
3. Compute $m \times m$ matrices $G = U^T U$ and $H = U^T V$.

4. Obtain an $m \times m$ interaction matrix B from the equation $GB = H$.

5. Solve the eigenvalues and right-hand eigenvectors of B. Here the eigenvalues of B are approximations of the dominant eigenvalues of P.

6. Sort these eigenvalues and their corresponding right-hand eigenvectors of B. Let Z be the matrix form of right-hand eigenvectors of B.

7. Calculate $W = VZ$, which gives improved right-hand eigenvectors of P.

8. Normalize W. If the convergence test is not satisfied, go Step 2; otherwise the algorithm stops.

Lopsided iteration has been found to be simple and efficient in computer time and memory space.

The Method of Arnoldi for Eigensolutions

The method of Arnoldi can be used to approximate the largest eigenvalues and corresponding eigenvectors of a real nonsymmetric matrix $P_{n\times n}$. The Arnoldi algorithm is an orthogonal projection process onto the Krylov subspace $\mathcal{K}_m$ for some vector v_1

$$\mathcal{K}_m = \text{span}\{v_1, Pv_1, P^2v_1, \ldots, P^{m-1}v_1\},$$

from which the approximations of eigensolutions are extracted. This method is attractive for large sparse matrices.

The Krylov projection method starts with a nonzero normalized approximation v_1, and then the modified Gram-Schmidt orthogonalization procedure is used to construct an orthonormal basis of the Krylov subspace. Let $V_m = (v_1, v_2, \ldots, v_m)$ be the orthonormal basis of the Krylov subspace; then $H_m = V_m^T P V_m$ where H_m is an $m \times m$ upper Hessenberg matrix, i.e., the nonzero elements are only in positions $\{i, j\}$ with $1 \leq i \leq \min(j+1, m), 1 \leq j \leq m$. Approximations to some eigenvalues of P and the corresponding eigenvectors can be obtained from the eigenvalues and eigenvectors of H_m. This basic Arnoldi algorithm for eigensolutions is as follows:

1. Choose an integer m as the size of the Krylov subspace and an initial approximation vector v_1 having $\|v_1\|_2 = 1$.

2. Generate the upper Hessenberg matrix H_m whose elements are h_{ij}:
   For $j = 1, 2, \ldots, m$

   - Compute $w = Pv_j$.
   - For $i = 1, 2, \ldots, j$
     - Compute $h_{ij} = v_i^T w$.
     - Compute $w = w - h_{ij} v_i$.
   - Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$.

3. Compute approximations to the eigenvalues and eigenvectors of P:

   - Solve eigenvalue problem: $H_m y = \lambda y$. The eigenvalues λ are approximate eigenvalues of P.
   - Compute approximate eigenvectors $V_m y$ of P.

The Lanczos Algorithm

Let P be a real $n \times n$ sparse matrix. If n is large, the Lanczos method can also be applied to approximate the extreme eigenvalues and eigenvectors of P. The algorithm of Lanczos is a projection process onto the Krylov subspaces, i.e., the approximations to exact eigensolutions of P are extracted from a small-dimensional Krylov subspace $\mathcal{K}_m$ with constraints on another Krylov subspace $\mathcal{L}_m$.

The main idea of the Lanczos method is to transform P into a real $m \times m$ tridiagonal matrix T_m. Let $V_m = (v_1, \ldots, v_m)$ and $W_m = (w_1, \ldots, w_m)$ be the bases of subspaces $\mathcal{K}_m$ and $\mathcal{L}_m$, respectively. If P is symmetric, the methods of Arnoldi and Lanczos are, in fact, the same. The upper Hessenberg matrix H_m calculated in Arnoldi's method turns into a real symmetric tridiagonal matrix T_m such that $T_m = V_m^T P V_m$, where V_m is the orthonormal basis of $\mathcal{K}_m$. This means that $V_m = W_m$ in the Lanczos method. Approximations to a subset of the eigenvalues and the corresponding right-hand eigenvectors of P can be obtained from the eigensolutions of T_m. If P is a nonsymmetric matrix, then a general similarity transformation may be applied to reduce P to a tridiagonal matrix. Based on this idea, we construct the bases V_m and W_m of the Krylov subspaces $\mathcal{K}_m$ and $\mathcal{L}_m$ such that $W_m^T V_m = I$, where I is an $m \times m$ identity matrix, and the expression $W_m^T P V_m$ is equal to a

tridiagonal matrix T_m. Thus we can compute the eigensolution of $T_m y = \lambda y$, and the values λ and vectors $V_m y$ are taken as approximations to a set of the eigenvalues and right-hand eigenvectors of P.

The details of the procedures for Lanczos algorithms are described in [35].

Chapter 3

Spectral Graph Partitioning and Clustering

3.1 Introduction and Literature Review

A graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, can be partitioned into two disjoint subsets V_1 and V_2, $V_1 \cup V_2 = V$, by deleting those edges that connect a vertex in one of subsets to a vertex in the other subset. It is frequently desirable that each subset have approximately the same number of vertices. There are many reasons for seeking a graph partition: a common objective is to partition the graph in such a way that the partition has minimum cost, e.g., minimum number of edges cut. In a weighted graph this involves finding the particular set of edges which once eliminated, partitions the graph in the desired manner and for which the sum of the weights on the removed edges is as small as possible. In graph theoretic language, this is called the minimum cut. Given two sets of vertices V_1 and V_2 which partition a graph, the numerical value assigned to the cut, called the "cut value", is given by

$$\text{cut}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij},$$

where w_{ij} is the weight on the edge that connects vertex i to vertex j. In an unweighed graph, w_{ij} is equal to 1 if there is an edge from vertex i to vertex j and is equal to 0 otherwise.

It turns out that a minimum cut partition is also the partition which maximizes the

total weight of the remaining edges within subsets. In certain applications it is important to determine which parts of a graph have strong cohesion, i.e., the sum of total edge weight in a subset, as represented by the strength of the edges that connect the vertices in the different subsets. Thus the cohesion problem of how tightly certain states are clustered together may be viewed as the dual of the minimum cut problem of a graph.

Since graph partitioning is an NP-complete problem [14], there are a number of heuristic methods that can be used to find good approximations. Such methods include the Fiduccia-Mattheyses (FM) [10] algorithm and the Kernighan-Lin (KL) [20] algorithm. Both FM and KL provide a local minimum based on given initial partitionings [7].

Spectral graph partitioning is another heuristic approach for finding good solutions to the minimum cut problem. It was introduced in the early 1970s [11], and popularized in the early 1990s [30]. Spectral clustering methods are based on spectral graph theory [4, 5] and generally consist of three stages:

(a) Construct a matrix representation based on a 'distance' between data points.

(b) Calculate an eigen-decomposition of this matrix and map each point into a lower-dimensional representation using one or more eigenvectors.

(c) Group data points into clusters based on the lower-dimensional representation.

Fiedler was the first to use the eigenvalues and eigenvectors of the Laplacian matrix of a graph to create tools for measuring algebraic connectivity. Fiedler has shown that the eigenvector of the Laplacian matrix corresponding to the second smallest eigenvalue (the so-called Fiedler vector) provides a heuristic approach for determining the minimum cut of a graph [11, 12]. Besides minimum cut and maximum association, there are a number of different spectral graph clustering objectives including normalized cut [34] and ratio cut/association [2]. Maximum association and minimum cut can be viewed as dual problems of each other and computed at the same time [40, 8]. In recent research, multiple eigenvectors have been used simultaneously to obtain a K-way partition, rather than a 2-way partition [34]. Generally, spectral partitioning gives better global solutions than the FM and KL algorithms [7].

The singular value decomposition (SVD) has also been used to generate clusters on graph nodes. For example, Sherriff et al. [33] perform the SVD on an analysis matrix

of change records in a software system to generate clusters in a *Software Change Impact Analysis*. Elsewhere, the SVD had been applied on a term-document matrix to cluster terms and documents [7, 28]. It is only natural to wonder if the clustering from an SVD is also based on a graph partitioning and we show that the answer is in the affirmative. The mathematical relationship between an SVD and an eigenanalysis [26, 16], exposes the fact that the singular vectors obtained when SVD is used in clustering are closely related to the eigenvectors from an eigenanalysis during spectral clustering. This leads us to suggest that it is beneficial to consider the SVD of the node-edge incidence matrix since it can be used to obtain edge clusters of a graph and not just vertex clusters.

3.2 Minimum Cut in Spectral Clustering

In this section, we first review the basic procedure of spectral clustering.

Definition 3.2.1. *Given a finite, undirected graph $G = (V, E)$ having n vertices, the adjacency matrix A of G is an $n \times n$ matrix, such that*

$$a_{ij} = \begin{cases} w_{ij}, & \text{if } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

The non-diagonal element a_{ij} of A is the weight w_{ij} of the edge $\{i, j\}$ connecting vertex i to vertex j, and the diagonal element a_{ii} is the weight on a loop (if any) on vertex i. For a simple graph with no self-loops, the diagonal elements of A are all zero. For an unweighted graph, the entries of A have the value 1 or 0 in position (i, j) according to whether vertices i and j are adjacent or not. Thus, for a simple undirected graph, the adjacency matrix is a symmetric matrix. We now introduce the *Laplacian matrix* of a graph, since it plays an important role in spectral graph partitioning.

Definition 3.2.2. *The Laplacian matrix L of a graph G is the $n \times n$ matrix whose elements are as follows.*

$$L_{ij} = \begin{cases} \sum_k w_{ik}, & \text{if } i = j \\ -w_{ij}, & \text{if } i \neq j, \ i \text{ and } j \text{ are adjacent} \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

Thus L is a symmetric matrix and from our previous definition of the adjacency matrix A, it can be obtained as $L = D - A$, where D is a diagonal matrix whose i^{th} diagonal element is $D_{ii} = \sum_k w_{ik}$.

Definition 3.2.3. *The oriented incidence matrix of a graph G, denoted by B_G, is an $n \times m$ matrix having one row per vertex and one column per edge. The column of B_G corresponding to an edge from vertex i to j has zeros everywhere except in positions i and j: the two nonzero elements in this column are given by $\sqrt{w_{ij}}$ and $-\sqrt{w_{ij}}$ and it matters not which one has the negative sign.*

For example, the matrix B_G shown below displays three edges, connecting vertex 2 and vertex 3; vertex 1 and vertex 4; and finally connecting vertex 4 and vertex 5.

$$\mathbf{B_G} = \begin{pmatrix} 0 & \sqrt{w_{14}} & 0 & \dots \\ \sqrt{w_{23}} & 0 & 0 & \dots \\ -\sqrt{w_{23}} & 0 & 0 & \dots \\ 0 & -\sqrt{w_{14}} & \sqrt{w_{45}} & \dots \\ 0 & 0 & -\sqrt{w_{45}} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

It is easy to show from the definitions that $L = B_G {B_G}^T$.

Theorem 3.2.1. *The Laplacian matrix L of a graph G has a number of interesting properties, including:*

1. *L is symmetric positive semi-definite. As such its eigenvalues are all real and non-negative. Furthermore the eigenvectors of L constitute a full set of n real orthogonal vectors.*

2. *$Le = 0$, where e is a column vector whose elements are all equal to 1. Thus 0 is the smallest eigenvalue of L and e is its corresponding eigenvector.*

3. *For any vector x, we have*

$$x^T L x = \sum_{\{i,j\} \in E} w_{ij}(x_i - x_j)^2. \tag{3.3}$$

Proof. Given the Laplacian matrix L of G:

1. Since $L = B_G {B_G}^T$, $x^T L x = x^T B_G {B_G}^T x = ({B_G}^T x)^T {B_G}^T x = y^T y \geq 0$, for all x and therefore L is symmetric positive semi-definite. From matrix theory [16], the eigenvalues of such matrices are non-negative and real; and the eigenvectors constitute a full set of n orthogonal vectors.

2. From its definition, the sum of elements in each row of L is equal to 0. Hence it follows that $Le = 0$, where $e = (1, 1, \ldots, 1)^T$.

3. Let k be the element of ${B_G}^T x$ that corresponds to edge $\{i, j\}$. Then

$$({B_G}^T x)_k = \sqrt{w_{ij}}(x_i - x_j), \tag{3.4}$$

and it follows that

$$\begin{aligned} x^T L x = x^T B_G {B_G}^T x &= (x^T B_G)_{1\times m}({B_G}^T x)_{m\times 1} \\ &= \sum_{\{i,j\}\in E} w_{ij}(x_i - x_j)^2. \end{aligned}$$

□

Let us now focus on finding the minimum cut of a graph based on these properties of the matrix L.

Definition 3.2.4. *Given a partition of V into V_1 and V_2 ($V_1 \cup V_2 = V$), a partition vector p is defined as*

$$p_i = \begin{cases} +1, & \textit{vertex } i \in V_1, \\ -1, & \textit{vertex } i \in V_2. \end{cases} \tag{3.5}$$

Clearly we see that $p^T p = n$. Given a Laplacian matrix L and a partition vector p, we have, from Equation (3.3),

$$p^T L p = \sum_{\{i,j\}\in E} w_{ij}(p_i - p_j)^2.$$

Observe from Equation (3.5), that the weight of edges within each set V_1 or V_2 is not counted in this sum, while the weight of each edge connecting a vertex of V_1 to a vertex in V_2 is multiplied by a factor of 4. Given that we have defined the cut value as $\text{cut}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij}$, it follows that $p^T L p = 4\,\text{cut}(V_1, V_2)$ and the Rayleigh quotient is thus

$$\frac{p^T L p}{p^T p} = \frac{1}{n} \cdot 4\,\text{cut}(V_1, V_2). \tag{3.6}$$

Here we see that the minimum-cut problem has been translated into a new format, that of finding an optimal partition vector p so as to minimize the Rayleigh quotient [7]. The problem can be presented as

$$\begin{aligned} &\text{Minimize} && \frac{p^T L p}{p^T p}, \\ &\text{such that} && p_i = \pm 1, \quad p^T p = n, \quad \text{and} \\ & && |e^T p| = 0 \text{ or } 1, \text{ where } e^T = (1, 1, \ldots, 1). \end{aligned} \tag{3.7}$$

The constraint $|e^T p| = 0$ or 1 ensures that we have two almost equally-sized subsets from partitioning. This constraint will be dropped later in the spectral heuristic approach, as will the constraint on p.

Theorem 3.2.2. *The maximum and minimum of the Rayleigh quotient of the Laplacian matrix L are the largest and smallest eigenvalues of L respectively:*

$$\lambda_{max} = \max_{x \neq 0} \frac{x^T L x}{x^T x} \quad \text{and} \quad \lambda_{min} = \min_{x \neq 0} \frac{x^T L x}{x^T x}, \tag{3.8}$$

where x is the eigenvector of L corresponding to λ_{max} and λ_{min} respectively.

Proof. The eigenvalues of a Laplacian matrix L are real and can be ordered as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Also, since L is real-symmetric, it is orthogonally similar to a diagonal matrix [26]. Therefore there exist a matrix U such that $U^T L U = D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, where D is a diagonal matrix whose diagonal elements are the eigenvalues of L, and U is orthogonal, i.e., the columns of U constitute a complete orthonormal set of eigenvectors of L. Equivalently, we may write $L = UDU^T$. Setting $y = U^T x$, we have

$$\|x\|_2 = 1 \Longleftrightarrow \|y\|_2 = \|U^T x\|_2 = (x^T U U^T x)^{1/2} = (x^T x)^{1/2} = \|x\|_2 = 1,$$

since U is orthogonal and hence $U^TU = I$. This leads to the following equivalent form of the Rayleigh quotient

$$\begin{aligned}
\max_{\|x\|_2=1} x^TLx &= \max_{\|x\|_2=1} (U^Tx)^TD(U^Tx) \\
&= \max_{\|y\|_2=1} y^TDy \\
&= \max_{\|y\|_2=1} \sum_{i=1}^{n} \lambda_i |y_i|^2 \\
&\leq \max_{\|y\|_2=1} \lambda_1 \sum_{i=1}^{n} |y_i|^2 = \lambda_1 \qquad (3.9)
\end{aligned}$$

with equality when $y = e_1 = [1, 0, 0, \ldots, 0]^T$, and x is an eigenvector of unit norm associated with λ_1. The expression for the smallest eigenvalue, λ_n, is obtained by writing

$$\begin{aligned}
\min_{\|x\|_2=1} x^TLx &= \min_{\|x\|_2=1} (U^Tx)^TD(U^Tx) \\
&= \min_{\|y\|_2=1} y^TDy \\
&= \min_{\|y\|_2=1} \sum_{i=1}^{n} \lambda_i |y_i|^2 \\
&\geq \min_{\|y\|_2=1} \lambda_n \sum_{i=1}^{n} |y_i|^2 = \lambda_n \qquad (3.10)
\end{aligned}$$

with equality being satisfied at an eigenvector of unit norm associated with λ_n. Consequently,

$$\lambda_1 \geq \frac{x^TLx}{x^Tx} \geq \lambda_n \quad \text{for all } x \neq 0.$$

□

The minimum value of the Rayleigh quotient is zero, which is the smallest eigenvalue of L corresponding to the eigenvector e. This partition vector indicates that all the vertices of the graph are in the same set, which means nothing is cut. This is the trivial partition. It is the second smallest eigenvalue (referred to as the Fiedler value) of L that provides the optimal value, and its corresponding eigenvector (referred to as the Fiedler

vector) gives the real valued solution to our minimum cut problem. Notice that the reason the Fiedler vector is not necessarily the solution to our original problem, Equation (3.7), is the constraint of Equation (3.5) on p (that p_i takes on two discrete values) and the constraint of obtaining two almost equally-sized subsets is not automatically satisfied. In fact, relaxing these constraints is what makes this optimization problem easy to solve and lets us get a better solution for decomposing a graph, since well-decomposed subsets based on the minimum cut are not necessarily equally-sized.

Once the eigenvector corresponding to the second smallest eigenvalue has been computed, we can partition the vertices into two subsets. In the ideal case, the eigenvector takes on only two discrete values and the signs of the values tell us exactly how to partition the graph. However, our eigenvectors assume continuous values and we need to choose a splitting point to partition it into two parts. Here we take value zero. Naturally, there are many other different ways of choosing such a splitting point. One can take the median value as the splitting point or one can search for the splitting point such that the resulting partition has the best cut value. A scientific way is to use the K-means clustering algorithm.

3.3 The Signless Laplacian for Graph Node Clustering

Sometimes we wish to determine clustering information for a graph G, i.e., to determine which vertices in G have strong mutual cohesion. The problem of maximizing the total weight within two clusters is the dual of the minimum cut problem. This clustering objective is called maximum association. In this section, we propose a new method for clustering nodes with the objective of maximum association, by introducing an eigenanalysis of the signless Laplacian matrix M. We shall see that the clustering solution thereby obtained coincides with the solution of the minimum cut problem. We now proceed through the details of this dual problem and describe our approach to solving it.

Definition 3.3.1. *Given a graph $G = (V, E)$ and two clusters V_1, V_2 where $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$, the maximum association is defined as*

$$\max_{V_1,V_2}(assoc(V_1) + assoc(V_2)) = \max_{V_1,V_2}(\sum_{i,j\in V_1} w_{ij} + \sum_{i,j\in V_2} w_{ij}). \tag{3.11}$$

The partition vector p has been defined in Equation (3.5) as

$$p_i = \begin{cases} +1, & \text{vertex } i \in V_1, \\ -1, & \text{vertex } i \in V_2. \end{cases}$$

Equation (3.3) suggests how we might find the maximum total weight within clusters. If we change the minus sign in Equation (3.3) and construct a matrix M to obtain

$$p^T M p = \sum_{\{i,j\} \in E} w_{ij}(p_i + p_j)^2, \tag{3.12}$$

then we see that the edges connecting two subsets V_1 and V_2 do not contribute to the value of this equation, but that the edges within each cluster contribute 4 times their weight. The dual problem can then be solved, and the optimal p which maximizes $p^T M p$ provides our sought-after clustering information.

Definition 3.3.2. *The unoriented incidence matrix of a graph G, denoted by I_G, is an $n \times m$ matrix having one row per vertex and one column per edge. The column of I_G corresponding to an edge from vertex i to j has zeros everywhere except in positions i and j: the two nonzero elements are both $\sqrt{w_{ij}}$.*

To obtain the matrix M, we need the unoriented incidence matrix I_G of G. I_G can be obtained by taking the absolute value of each entry in the oriented incidence matrix, which forces I_G to be nonnegative. An example of I_G is shown below

$$\mathbf{I_G} = \begin{pmatrix} 0 & \sqrt{w_{14}} & 0 & \dots \\ \sqrt{w_{23}} & 0 & 0 & \dots \\ \sqrt{w_{23}} & 0 & 0 & \dots \\ 0 & \sqrt{w_{14}} & \sqrt{w_{45}} & \dots \\ 0 & 0 & \sqrt{w_{45}} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Given any vector x, let k be the element of ${I_G}^T x$ that corresponds to edge $\{i, j\}$. Then,

$$({I_G}^T x)_k = \sqrt{w_{ij}}(x_i + x_j). \tag{3.13}$$

If we now construct the matrix $M = I_G {I_G}^T$, then

$$x^T M x = x^T I_G {I_G}^T x = (x^T I_G)_{1\times m}({I_G}^T x)_{m\times 1}, \quad \text{for all } x.$$

This implies that

$$x^T M x = \sum_{\{i,j\}\in E} w_{ij}(x_i + x_j)^2.$$

Replacing x with the partition vector p yields Equation (3.12). The matrix M is called the signless Laplacian matrix [6].

Definition 3.3.3. *The signless Laplacian matrix M of a graph G is the $n \times n$ matrix whose elements are as follows.*

$$M_{ij} = \begin{cases} \sum_k w_{ik}, & \text{if } i = j \\ w_{ij}, & \text{if } i \neq j,\ i \text{ and } j \text{ are adjacent} \\ 0, & \text{otherwise.} \end{cases}$$

It can also be obtained as $M = D + A$, where A is the adjacency matrix of the graph and D is a diagonal matrix with diagonal entry $D_{ii} = \sum_k w_{ik}$.

The Rayleigh quotient $p^T Mp/p^T p$ provides a quantitative evaluation of the cohesion of vertices within clusters. We can maximize the Rayleigh quotient $p^T Mp/p^T p$ to obtain the two clusters. Indeed, since $M = 2D - L$, maximizing $p^T Mp/p^T p$ is equivalent to minimizing $p^T Lp/p^T p$ in the minimum cut problem. The mathematical model of the graph clustering problem in maximum association is

$$\begin{aligned} &\text{Maximize} && \frac{p^T Mp}{p^T p}, \\ &\text{such that} && p_i = \pm 1, \quad p^T p = n. \end{aligned} \tag{3.14}$$

Theorem 3.3.1. *All the eigenvalues of M are real and the right-hand eigenvector corresponding to the largest eigenvalue is the only eigenvector of M whose elements are all nonzero and positive.*

Proof. Since the matrix M is real-symmetric by definition, all its eigenvalues are real. The graph G is an undirected, connected graph, which means the adjacency matrix A of G is an irreducible matrix. Thus, the matrix $M = D + A$ is also an irreducible matrix. By the Perron-Frobenius theorem, since M is nonnegative and irreducible, there is only one eigenvalue λ_{max} on the spectral radius of M and the unique vector p defined by $Mp = \lambda_{max}p$ is the only positive eigenvector. □

We already know that the eigenvalues of M can be depicted as the maximum, minimum and intermediate values of the Rayleigh quotient. Since all the elements in the eigenvector corresponding to the largest eigenvalue of M are positive, using this eigenvector implies that all vertices are in one cluster which does not satisfy our objective of obtaining two clusters. Therefore the second largest eigenvalue of M is the maximum value that we want and it approximately estimates the value of the cohesion of vertices within the clusters (maximum association); the associated eigenvector generates the clustering information among these vertices. The clustering result indicated here coincides with the minimum cut result.

This leads us to ask why we still need to model the dual problem if the results from the dual and the original are the same. One significant reason is that the eigenvalues of the signless Laplacian have important physical interpretations. They provide a quantitative evaluation of the total weight of edges within clusters. Since the eigenvalues of the Laplacian matrix provide a measure of the total weight of edges connecting clusters, the ratio of these eigenvalues can be used to analyze and evaluate the effects of partitioning and clustering. When we turn to applications in Chapter 7, and in particular to the application "Software Change Impact Analysis" we shall see that the signless Laplacian matrix is used.

Example 3.3.1. Given an undirected graph $G = (V, E)$ with six vertices and eight weighted edges as shown in the figure below, we wish to find a partition with subsets of vertices V_1^*, V_2^* of V such that $\text{cut}(V_1^*, V_2^*) = \min_{V_1,V_2}\{\text{cut}(V_1, V_2)\}$. In this simple example, it is easy to see that the partition which generates the minimum cut is $V_1 = \{1, 2, 3\}$ and $V_2 = \{4, 5, 6\}$. The minimum cut value is 0.3.

This partition can be found from the Fiedler vector. To compute this vector, we first need to get the Laplacian matrix. From their definitions, the adjacency matrix A and Laplacian matrix L are

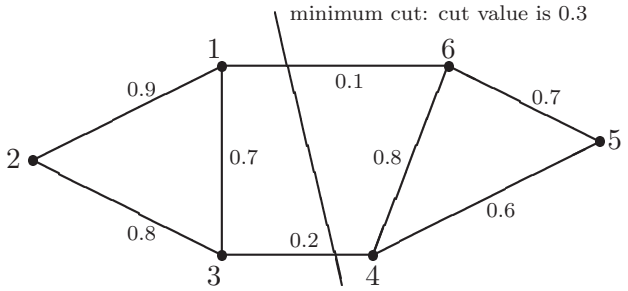

Figure 3.1: A sample graph with six vertices.

$$\mathbf{A} = \begin{pmatrix} 0 & 0.9 & 0.7 & 0 & 0 & 0.1 \\ 0.9 & 0 & 0.8 & 0 & 0 & 0 \\ 0.7 & 0.8 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0.6 & 0.8 \\ 0 & 0 & 0 & 0.6 & 0 & 0.7 \\ 0.1 & 0 & 0 & 0.8 & 0.7 & 0 \end{pmatrix},$$

$$\mathbf{L} = \begin{pmatrix} 1.7 & -0.9 & -0.7 & 0 & 0 & -0.1 \\ -0.9 & 1.7 & -0.8 & 0 & 0 & 0 \\ -0.7 & -0.8 & 1.7 & -0.2 & 0 & 0 \\ 0 & 0 & -0.2 & 1.6 & -0.6 & -0.8 \\ 0 & 0 & 0 & -0.6 & 1.3 & -0.7 \\ -0.1 & 0 & 0 & -0.8 & -0.7 & 1.6 \end{pmatrix}.$$

The Laplacian matrix L can also be obtained as $L = B_G {B_G}^T$, where

$$\mathbf{B_G} = \begin{pmatrix} \sqrt{0.9} & 0 & \sqrt{0.7} & \sqrt{0.1} & 0 & 0 & 0 & 0 \\ -\sqrt{0.9} & \sqrt{0.8} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\sqrt{0.8} & -\sqrt{0.7} & 0 & \sqrt{0.2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\sqrt{0.2} & \sqrt{0.8} & \sqrt{0.6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\sqrt{0.6} & \sqrt{0.7} \\ 0 & 0 & 0 & -\sqrt{0.1} & 0 & -\sqrt{0.8} & 0 & -\sqrt{0.7} \end{pmatrix}.$$

The eigenvalue of L, in increased order, are

$$\textbf{Eigenvalues} = \begin{pmatrix} 0 & 0.1876 & 1.9832 & 2.2582 & 2.5487 & 2.6222 \end{pmatrix},$$

and the corresponding eigenvectors are the columns

$$\textbf{Eigenvectors} = \begin{pmatrix} 0.4082 & -0.4080 & 0.0864 & -0.4285 & 0.3379 & 0.6014 \\ 0.4082 & -0.4401 & 0.1094 & -0.0975 & 0.1841 & -0.7644 \\ 0.4082 & -0.3731 & -0.1359 & 0.5501 & -0.5755 & 0.2046 \\ 0.4082 & 0.3670 & -0.5473 & 0.3544 & 0.5229 & 0.0091 \\ 0.4082 & 0.4514 & 0.7652 & 0.2025 & 0.0271 & 0.0483 \\ 0.4082 & 0.4027 & -0.2778 & -0.5810 & -0.4966 & -0.0990 \end{pmatrix}.$$

The Fiedler value is $\lambda_2 = 0.1876$ and the Fiedler vector (partition vector) is

$$\begin{pmatrix} -0.4080 \\ -0.4401 \\ -0.3731 \\ 0.3670 \\ 0.4514 \\ 0.4027 \end{pmatrix}.$$

Therefore, if we use zero as the splitting point, the signs of components of the Fiedler vector give the minimum cut, $V_1 = \{1, 2, 3\}$ and $V_2 = \{4, 5, 6\}$, which coincides with the result obtained previously.

It is obvious that the solution of the minimum cut problem also provides information which may be used to group the vertices of the graph. Vertices 1, 2 and 3 constitute one cluster, and vertices 4, 5 and 6 form another cluster. Each cluster has strong cohesion. We can derive this same result by calculating the eigenvector of the signless Laplacian matrix M. Using the definition $M = D + A$, we have

$$\mathbf{M} = \begin{pmatrix} 1.7 & 0.9 & 0.7 & 0 & 0 & 0.1 \\ 0.9 & 1.7 & 0.8 & 0 & 0 & 0 \\ 0.7 & 0.8 & 1.7 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 1.6 & 0.6 & 0.8 \\ 0 & 0 & 0 & 0.6 & 1.3 & 0.7 \\ 0.1 & 0 & 0 & 0.8 & 0.7 & 1.6 \end{pmatrix},$$

with eigenvalues and associated eigenvectors given by

$$\textbf{Eigenvalues} = \begin{pmatrix} 0.6918 & 0.7581 & 0.8153 & 1.1043 & 2.8996 & 3.3309 \end{pmatrix},$$

$$\textbf{Eigenvectors} = \begin{pmatrix} -0.3318 & -0.2012 & 0.5223 & 0.4930 & -0.1426 & 0.5597 \\ 0.1546 & 0.5654 & -0.5136 & 0.1750 & -0.1764 & 0.5752 \\ 0.1784 & -0.4393 & -0.0195 & -0.6850 & -0.1041 & 0.5429 \\ -0.3566 & 0.5115 & 0.3130 & -0.3852 & 0.5802 & 0.1677 \\ -0.4591 & -0.4137 & -0.5880 & 0.1696 & 0.4830 & 0.1015 \\ 0.7045 & -0.1182 & 0.1389 & 0.2828 & 0.6064 & 0.1509 \end{pmatrix}.$$

The elements of the eigenvector (column) associated with the largest eigenvalue are all non-zero and of the same sign, as guaranteed by the theory. If zero is chosen to be the splitting point, this eigenvector shows that all the vertices of the graph are in one cluster: it provides the trivial cluster. Instead we focus on the eigenvector corresponding to the second largest eigenvalue. This vector gives the same clustering information as the Fiedler vector. The second largest eigenvalue 2.8996 represents the relative quantitative evaluation of total weight within clusters, but not the exact one. The term "relative" here is used according to the Fiedler value $\lambda_2 = 0.1876$, which represents the relative quantitative evaluation of total weight of edges cut. Since the eigenvalues here are both relative, the ratio of these two values has real meaning. For this small example, the exact total weight of edges within clusters is 4.5, and the exact total weight of edges between the clusters (edges cut) is 0.3. The proportion is $0.3/4.5 = 0.0667$, which is very close to the ratio of the two eigenvalues $0.1876/2.8996 = 0.0647$.

3.4 Spectral Graph Clustering with Negative Edge Weight

In a weighted graph, each edge is assigned with a numerical value, called its weight. In this section, let us consider the spectral graph clustering for the case of a graph having negative edge weights.

Given a weighted graph G, the adjacency matrix A of G is given by Equation (3.1)

$$a_{ij} = \begin{cases} w_{ij}, & \text{if vertex } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise,} \end{cases}$$

where w_{ij} is the weight on edge $\{i, j\}$ and w_{ij} may be negative.

Definition 3.4.1. *For a weighted graph G with negative edge weights, the vertex-edge incidence matrix I_G of G is an $n \times m$ matrix having one row per vertex and one column per edge: for an edge with $w_{ij} > 0$, the corresponding column of I_G has zeros everywhere except $\sqrt{w_{ij}}$ in positions i and j; for an edge with $w_{ij} < 0$, the corresponding column of I_G has only nonzero elements $\sqrt{|w_{ij}|}$ and $-\sqrt{|w_{ij}|}$ in positions i and j (the position of the negative value is arbitrary).*

An example of I_G below illustrates three edges in G, connecting vertex 2 and vertex 3; vertex 1 and vertex 4; and vertex 4 and vertex 5. The edge connecting vertex 4 and vertex 5 has a negative weight $w_{45} < 0$.

$$\mathbf{I_G} = \begin{pmatrix} 0 & \sqrt{w_{14}} & 0 & \dots \\ \sqrt{w_{23}} & 0 & 0 & \dots \\ \sqrt{w_{23}} & 0 & 0 & \dots \\ 0 & \sqrt{w_{14}} & \sqrt{|w_{45}|} & \dots \\ 0 & 0 & -\sqrt{|w_{45}|} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

We may obtain an $n \times n$ symmetric matrix $M = I_G {I_G}^T$ just as we did for the signless Laplacian matrix. The matrix M can also be obtained as $M = D + A$, where A is the adjacency matrix of G and D is a diagonal matrix whose diagonal element is $d_{ii} = \sum_j |w_{ij}|$.

Theorem 3.4.1. *For any vector x,*

$$x^T M x = \sum_{w_{ij}>0} w_{ij}(x_i + x_j)^2 + \sum_{w_{ij}<0} |w_{ij}|(x_i - x_j)^2. \tag{3.15}$$

Proof. Let the kth element of the vector ${I_G}^T x$ be associated with edge $\{i, j\}$. Then from the Definition 3.4.1 of I_G, for edge $w_{ij} > 0$:

$$({I_G}^T x)_k = \sqrt{w_{ij}}(x_i + x_j);$$

and for edge $w_{ij} < 0$

$$({I_G}^T x)_k = \sqrt{|w_{ij}|}(x_i - x_j).$$

Since $M = I_G {I_G}^T$, it follows that

$$\begin{aligned} x^T M x &= x^T I_G {I_G}^T x \\ &= ({I_G}^T x)^T ({I_G}^T x) \\ &= \sum_{w_{ij}>0} w_{ij}(x_i + x_j)^2 + \sum_{w_{ij}<0} |w_{ij}|(x_i - x_j)^2. \end{aligned}$$

□

In the context of graph partitioning, we seek to partition of V into subsets V_1 and V_2. Hence, a partition vector p is defined as in the Equation (3.5)

$$p_i = \begin{cases} +1, & \text{vertex } i \in V_1, \\ -1, & \text{vertex } i \in V_2. \end{cases}$$

Then we have, from Equation (3.15),

$$p^T M p = \sum_{w_{ij}>0} w_{ij}(p_i + p_j)^2 + \sum_{w_{ij}<0} |w_{ij}|(p_i - p_j)^2.$$

It can be observed that the positive weight of edges within V_1 or V_2 is counted into this sum, and the absolute value of the negative weight of edges connecting V_1 and V_2 is also counted into this sum. However, positive edges connecting V_1 and V_2, and negative edges within V_1 or V_2 are not counted. Therefore, to maximize $p^T Mp/p^T p$ is to seek a partition with more positive edges within subsets V_1 or V_2 and more negative edges cut between the subsets.

If we obtain a matrix $L = D - A$ similar to the construction of the Laplacian matrix, then we have $M = D + A = 2D - L$. Since $p^T Dp/p^T p$ is a constant, maximizing $p^T Mp/p^T p$ is equivalent to minimizing $p^T Lp/p^T p$.

Result 3.4.1. *By Theorem 2.2.5, the vector p that maximizes the Rayleigh quotient $p^T Mp/p^T p$ is an eigenvector of M corresponding to the largest eigenvalue, and the vector q that minimizes the Rayleigh quotient $q^T Lq/q^T q$ is an eigenvector of L corresponding to the smallest eigenvalue. Therefore, for a graph G with negative weights, vector p or q provides a same heuristic solution of clustering vertices that is more positive edges within clusters and more negative edges cut between clusters.*

Example 3.4.1. A graph with negative edge weights is given as below:

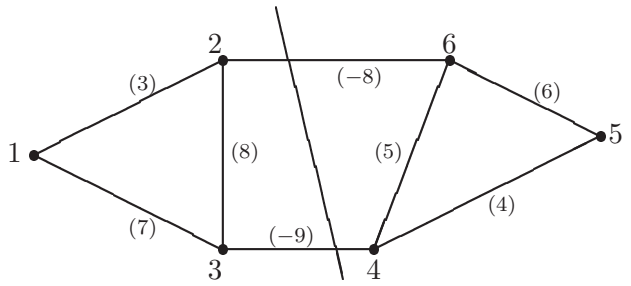

Figure 3.2: A sample graph with negative edge weights.

The vertex-edge incidence matrix I_G based on this graph is

$$\mathbf{I_G} = \begin{pmatrix} \sqrt{3} & 0 & \sqrt{7} & 0 & 0 & 0 & 0 & 0 \\ \sqrt{3} & \sqrt{8} & 0 & \sqrt{8} & 0 & 0 & 0 & 0 \\ 0 & \sqrt{8} & \sqrt{7} & 0 & \sqrt{9} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\sqrt{9} & \sqrt{5} & \sqrt{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{4} & \sqrt{6} \\ 0 & 0 & 0 & -\sqrt{8} & 0 & \sqrt{5} & 0 & \sqrt{6} \end{pmatrix}.$$

The matrix $M = I_G {I_G}^T$, also can be obtained by $M = D + A$, is

$$\mathbf{M} = \begin{pmatrix} 10 & 3 & 7 & 0 & 0 & 0 \\ 3 & 19 & 8 & 0 & 0 & -8 \\ 7 & 8 & 24 & -9 & 0 & 0 \\ 0 & 0 & -9 & 18 & 4 & 5 \\ 0 & 0 & 0 & 4 & 10 & 6 \\ 0 & -8 & 0 & 5 & 6 & 19 \end{pmatrix}.$$

The eigenvalue of M, in decreasing order, are

$$\mathbf{Eigenvalues} = \begin{pmatrix} 37.7194 & 23.8417 & 19.7604 & 8.9039 & 5.6920 & 4.0827 \end{pmatrix},$$

and the corresponding eigenvectors are the columns

$$\textbf{Eigenvectors} = \begin{pmatrix} -0.2147 & 0.2288 & 0.1778 & 0.7217 & 0.4492 & 0.3837 \\ -0.4635 & -0.1823 & 0.6404 & -0.3308 & -0.2180 & 0.4299 \\ -0.6517 & 0.5307 & -0.0266 & 0.0287 & -0.1830 & -0.5087 \\ 0.4144 & 0.0253 & 0.6950 & 0.3155 & -0.2167 & -0.4450 \\ 0.1361 & 0.3224 & 0.2721 & -0.5053 & 0.7266 & -0.1422 \\ 0.3524 & 0.7268 & -0.0208 & -0.1180 & -0.3772 & 0.4369 \end{pmatrix}.$$

According to the first eigenvector associated with the largest eigenvalue, we can obtain cluster {1,2,3} and cluster {4,5,6} based on the sign of the corresponding elements. The partition here is to cut the negative edges and to obtain the minimum cut and maximum association.

3.5 Singular Value Decomposition

The Singular Value Decomposition (SVD) is a powerful tool that finds application in the analysis of a matrix. The SVD technique can be used to reveal clustering characteristics and to obtain the minimum cut of a graph. In these applications we shall see that the singular vector and eigenvector approaches coincide, and yield the same heuristic result. We shall also introduce the possibility of applying the SVD to rectangle vector space models (unoriented incidence matrices) to generate edge clusters of a graph; the eigen-decomposition approach can be applied to square vector space model (Laplacian and signless Laplacian matrices) only. In this section, we will proceed through the details in three different cases. First is the SVD on the Laplacian and signless Laplacian matrices, the second is the SVD on the oriented incidence matrix and the third is the SVD on the unoriented incidence matrix. The weighted graphs of this section all have nonnegative edge weights.

3.5.1 SVD on the Laplacian and Signless Laplacian Matrix

The Laplacian matrix L is symmetric by definition and therefore it is orthogonally similar to a diagonal matrix. L has a complete orthogonal set of eigenvectors; $L = UDU^T$, where U is an orthogonal matrix and D is a diagonal matrix whose elements are the eigenvalues of L. Since the eigenvalues of L are nonnegative, the singular value decomposition

and a decomposition by eigenanalysis are identical. This is why singular vectors can be used to compute a heuristic solution for the minimum cut problem: they have exactly the same interpretation as the eigenvectors. The signless Laplacian matrix M has the same property. In other words, the singular values and singular vectors of M have a meaning that is consistent with that of the eigenvalues and eigenvectors, respectively. However, it is not always true that the SVD of a symmetric matrix is equivalent to its eigendecomposition. The singular values in SVD are nonnegative, whereas the eigenvalues need not be.

3.5.2 SVD on the Oriented Incidence Matrix

The SVD has a significant advantage over an eigenanalysis in that the matrix need not be square. A similarity transformation can be applied to a square matrix only, whereas an SVD can be applied to a rectangular matrix. This property naturally raises the question as to the results obtained when we apply the SVD to a rectangular incidence matrix. The answer is that the singular vectors generated by the SVD can be used to indicate the structure of clusters and thereby provide a heuristic solution to the minimum cut problem. To see this, let us consider the oriented incidence matrix B_G. Recall that B_G is an $n \times m$ matrix having one row per vertex and one column per edge. Each column of B_G has zeros everywhere except in positions i and j with $\sqrt{w_{ij}}$ and $-\sqrt{w_{ij}}$. If we apply the SVD on B_G, then

$$B_G = U_{n\times n} \begin{pmatrix} D_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}_{n\times m} V^T_{m\times m}, \tag{3.16}$$

where $U_{n\times n}$ and $V_{m\times m}$ are orthogonal matrices and $D_{r\times r}$ is a nonsingular diagonal matrix with $r < n$. We shall return to this inequality momentarily. Since we know the Laplacian matrix is $L = B_G {B_G}^T$, we have

$$\begin{aligned} L &= B_G {B_G}^T \\ &= U \begin{pmatrix} D_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}_{n\times m} V^T V \begin{pmatrix} D_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}^T_{n\times m} U^T \\ &= U S_{n\times n} U^T, \end{aligned} \tag{3.17}$$

where the $n \times n$ matrix S is diagonal. Since L is symmetric, the formula above is actually an eigen-decomposition of L: the matrix U contains the eigenvectors of L. This means that the left-hand singular vectors for B_G are a complete orthogonal set of eigenvectors for L. This coincides with the fact that the left-hand singular vectors of the oriented incidence matrix and eigenvectors of a Laplacian matrix provide vertex partitioning and clustering information. Observe that the matrix S is singular, since the smallest eigenvalue of L, namely zero, is included in the spectrum of S. Since each diagonal entry of D is the square root of the diagonal entry of S, the inequality $r < n$ must follow.

3.5.3 SVD on the Unoriented Incidence Matrix

The benefit of SVD is that not only does it provide clustering information on the n vertices, but it can also be used to generate information relating to the m edges. We discover that, to obtain clusters of edges, SVD should be applied to the unoriented incidence matrix I_G rather than B_G. When applying the SVD to I_G, we find

$$I_G = {U'}_{n\times n} \begin{pmatrix} {D'}_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}_{n\times m} {V'^T}_{m\times m} \tag{3.18}$$

where U' and V' are orthogonal and contain the left and right singular vectors respectively. Since the signless Laplacian matrix $M = I_G {I_G}^T$, we have

$$\begin{aligned} M &= I_G {I_G}^T \\ &= U' \begin{pmatrix} D'_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}_{n\times m} V'^T V' \begin{pmatrix} D'_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}^T_{n\times m} U'^T \\ &= U' S'_{n\times n} U'^T, \end{aligned} \tag{3.19}$$

where the matrix S' is diagonal. The left-hand singular vectors for I_G are a complete orthogonal set of eigenvectors for M. This means that U' contains vertex clustering information. On the other hand, the matrix V' contains edge clustering information. The reason is now provided.

By Definition 2.1.6, the line graph of an undirected graph G, written $L(G)$, is a graph such that:

(a) each vertex of $L(G)$ is an edge of G, and

(b) two vertices of $L(G)$ are adjacent if and only if the corresponding edges in G have a common endpoint (adjacent).

We can see that the line graph $L(G)$ of a graph G is another graph that represents the adjacencies between edges of G. The intuition here is that the clusters of edges in G can be used to represent clustering information for vertices in $L(G)$. Let us consider the unweighted graph to prove why the right-hand singular vectors of the unoriented incidence matrix I_G provide edge clustering information in G, as well as information concerning the vertices of $L(G)$.

Given an unweighted graph G, the adjacency matrix A is known as

$$a_{ij} = \begin{cases} 1, & \text{if vertex } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases}$$

The unoriented incidence matrix I_G of G has one row per vertex and one column per edge. The column of I_G corresponding to an edge from vertex i to j has zeros everywhere except in positions i and j which contain a 1. The signless Laplacian has $M = I_G {I_G}^T$.

Theorem 3.5.1. *[5] Let A_L be the adjacency matrix of the line graph $L(G)$ obtained from a graph G without multiple edges or self-loops. Then $E = {I_G}^T I_G$ is a nonnegative matrix such that*

$$E = A_L + 2I, \tag{3.20}$$

where I is identity matrix.

Proof. Each column of I_G contains exactly two unit values, so the diagonal entries of $E = {I_G}^T I_G$ are all equal to 2. From the definition of I_G, we can see that if two edges are adjacent in G, then entry e_{ij} of $E = {I_G}^T I_G$ corresponding to the ith and jth edge is equal to 1; otherwise $e_{ij} = 0$. Since the edges of G are the vertices of $L(G)$, $E = A_L + 2I$ is established. □

Notice that if we use the oriented incidence matrix B_G instead of I_G, then the product ${B_G}^T B_G$ is not a nonnegative matrix. The positions of negative values in this product are arbitrary, so that we cannot use it to get clustering results.

To analyze the vertex clustering information in the line graph $L(G)$, we need to construct the signless Laplacian matrix M_L of $L(G)$ and to maximize the Rayleigh quotient $p^T M_L p / p^T p$, where p is the partition vector with values ± 1. The mathematical model of the $L(G)$ clustering problem is:

$$\begin{array}{ll} \text{Maximize} & \dfrac{p^T M_L p}{p^T p,} \\ \text{such that} & p_i = \pm 1, \quad p^T p = n. \end{array} \tag{3.21}$$

Since $M_L = D_L + A_L$ where D_L is a diagonal matrix whose diagonal elements are the vertex degrees of $L(G)$, and $E = A_L + 2I$, then $M_L = E + D_L - 2I$. Therefore Equation (3.21) is equivalent to

$$\begin{array}{ll} \text{Maximize} & \dfrac{p^T E p}{p^T p} + \dfrac{p^T D_L p - 2p^T I p}{p^T p}, \\ \text{such that} & p_i = \pm 1, \quad p^T p = n. \end{array} \tag{3.22}$$

It is easy to see that $(p^T D_L p - 2p^T I p)/p^T p$ is a constant. Therefore the p that maximizes $p^T E p / p^T p$ in Equation (3.22) is equivalent to that used to maximize $p^T M_L p / p^T p$ in Equation (3.21). Since E is a symmetric nonnegative and irreducible matrix, by the Perron-Frobenius theorem, the first eigenvector of E corresponding to the largest eigenvalue, is positive. Therefore, we can derive the second eigenvector of E which can be used to cluster the vertices in $L(G)$.

Recall the SVD of the unoriented incidence matrix I_G:

$$\begin{aligned} E &= {I_G}^T I_G \\ &= V' \begin{pmatrix} D'_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}^T_{n\times m} {U'}^T U' \begin{pmatrix} D'_{r\times r} & 0 \\ 0 & 0 \end{pmatrix}_{n\times m} {V'}^T \\ &= V' S'_{m\times m} {V'}^T. \end{aligned} \tag{3.23}$$

where S' is a diagonal matrix whose diagonal entries are nonnegative. Since $E = {I_G}^T I_G$ is positive semidefinite, the eigenvalues of E are nonnegative. Indeed, the right-hand singular vectors for I_G in V' are the orthogonal eigenvectors of the matrix E.

Result 3.5.1. *The right-hand singular vectors of the unoriented incidence matrix I_G of a graph G provide clustering information for edges in graph G, represented by the vertices of line graph $L(G)$.*

This conclusion can be directly extended to weighted graphs. We first have to define the edge weights in the line graph $L(G)$ of a weighted graph G.

Definition 3.5.1. *Let e and f be two edges having weights w_e and w_f respectively in a weighted graph G. Then we can draw the line graph $L(G)$ of G having vertex i and j corresponding to the edge e and f in G respectively, and define the weight w_{ij} of the edge $\{i, j\}$ in $L(G)$ as $\sqrt{w_e w_f}$.*

This definition shows that if two adjacent edges both have strong weight in G, then the corresponding edge in $L(G)$ also has strong weight.

Definition 3.5.2. *The adjacency matrix A_L of $L(G)$, which can be used to depict the adjacency degree of two edges in G, is*

$$a_{ij} = \begin{cases} \sqrt{w_e w_f}, & \text{if vertex } i \text{ and } j \text{ are adjacent,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.24}$$

Given the unoriented incidence matrix I_G of G, we have

$$E = {I_G}^T I_G = A_L + 2D, \tag{3.25}$$

where D is a diagonal matrix whose diagonal entries are the edge weights of G. Thus, based on our previous discussion, the eigenvectors of E contain the node clusters in $L(G)$, and it is especially the eigenvector of E corresponding to the second largest eigenvalue that implies that the right-hand singular vectors of I_G contain the weighted edge clusters in G.

Summary

We can now summarize this new approach for clustering states. If, from the point of view of the singular value decomposition of the node-edge unoriented incidence matrix, there are two classes of states X and Y, we can then construct a rectangular matrix I_G with incidence structure that shows the relationship between X and Y. The matrix I_G has one row for each state of X and one column for each state of Y. The entry in row x and column y is a nonnegative weight which shows the degree to which x and y are related. To obtain information on how to best group two classes of states respectively into clusters, we can use either the singular vectors of I_G or the eigenvectors of $I_G{I_G}^T$ and ${I_G}^T I_G$. This approach can be applied to a term-document matrix as we now describe.

A collection of documents may be represented by a term-document matrix which contains rows corresponding to the terms and columns corresponding to the documents. The entry value in position (i, j) is the number of times term i appears in document j. If we set terms as state X and documents as state Y, then the term-document matrix, like I_G, has incidence structure that show the relationship between X (terms) and Y (documents). The terms and documents can also be viewed as the "vertices" and "edges" in a graph G respectively. Then the term-document matrix has a similar structure as the node-edge incidence matrix. The difference, however, is that the G is abstract: one single edge connects multiple vertices rather than two, and one single vertex can communicate multiple vertices only through one edge. In graph G, two edges may share more than one common vertex, so that the degree of adjacency of two edges in G can be shown in the edge weight of the line graph $L(G)$. Therefore clustering the edges (documents) of G is equivalent to clustering the vertices of $L(G)$. This is the same conclusion we get when we discuss traditional graph models. It follows that the SVD can be applied to a term-document matrix to develop a spectral co-clustering algorithm for generating clusters of terms and documents simultaneously. The left and right singular vectors of an appropriately scaled term-document matrix can be used to partition the terms and documents [7, 28].

3.6 Different Clustering Objectives

In the previous sections, all graph clustering techniques are based on the objective of minimum cut and maximum association. It is already known that minimizing the cuts and maximizing the associations can be achieved simultaneously. It is appropriate to ask if

minimum cut and maximum association are good clustering objectives? Actually there are two limitations concerning minimum cut:

1. Minimum cut is noise-sensitive.
2. Minimum cut may not be the optimal cut.

Therefore, a number of different spectral graph clustering objectives have been proposed, e.g., normalized cut and ratio cut/association.

Definition 3.6.1 (**Normalized Cut [34]**). *The normalized cut objective is one of the most popular graph clustering objectives and is a more "balanced" objective than minimum cut. It is also called balanced minimum cut. It is expressed as*

$$\mathrm{Ncut}(V_1, V_2) = \min_{V_1, V_2} \frac{\mathrm{cut}(V_1, V_2)}{\mathrm{vol}(V_1)} + \frac{\mathrm{cut}(V_1, V_2)}{\mathrm{vol}(V_2)}, \tag{3.26}$$

where $\mathrm{vol}(V_i)$ *is the total weight of edges originating from* V_i, $i = 1, 2$.

Definition 3.6.2 (**Normalized Association**). *The normalized association objective is expressed as*

$$\mathrm{Nassoc}(V_1, V_2) = \max_{V_1, V_2} \frac{\mathrm{assoc}(V_1)}{\mathrm{vol}(V_1)} + \frac{\mathrm{assoc}(V_2)}{\mathrm{vol}(V_2)}, \tag{3.27}$$

where $\mathrm{assoc}(V_k) = \sum_{i,j \in V_k} w_{ij}$, $k = 1, 2$.

Since $\mathrm{vol}(V_i) = \mathrm{assoc}(V_i) + \mathrm{cut}(V_1, V_2)$ for $i = 1, 2$, we can see the normalized cut is equivalent to the normalized association. They are the dual problems.

Definition 3.6.3 (**Ratio Association [34]**). *The ratio association objective is expressed as*

$$\mathrm{Rassoc}(V_1, V_2) = \max_{V_1, V_2} \frac{\mathrm{assoc}(V_1)}{|V_1|} + \frac{\mathrm{assoc}(V_2)}{|V_2|}, \tag{3.28}$$

where $|V_i|$ *is the size of set* V_i *for* $i = 1, 2$.

Definition 3.6.4 (**Ratio Cut [2]**). *The ratio cut objective is expressed as*

$$\mathrm{Rcut}(V_1, V_2) = \min_{V_1, V_2} \frac{\mathrm{cut}(V_1, V_2)}{|V_1|} + \frac{\mathrm{cut}(V_1, V_2)}{|V_2|}. \tag{3.29}$$

3.7 The Signless Laplacian for Normalized Cut

Shi et al. [34] have shown that the normalized cut can be modeled as

$$\begin{aligned} &\text{Minimize} && \text{Ncut} = \frac{p^T L p}{p^T D p}, \\ &\text{such that} && p_i = \pm 1, \quad p^T D e = 0, \end{aligned} \tag{3.30}$$

where L is the Laplacian matrix of a graph, D is a diagonal matrix whose i^{th} diagonal element is $D_{ii} = \sum_k w_{ik}$ and e is a vector of all ones. Equation (3.30) is the Rayleigh quotient. Therefore, we can minimize Equation (3.30) by solving the generalized eigenvalue problem of the Laplacian matrix L,

$$Lv = \lambda D v, \tag{3.31}$$

if p takes on real values. The second eigenvector v_2 of Equation (3.31) corresponding to the second smallest eigenvalue λ_2 provides the solution for minimizing the normalized cut. The λ_2 evaluates the minimum normalized cut value.

If we premultiply $D^{-1/2}$ on both side of Equation (3.31), then it can be rewritten as

$$D^{-1/2} L D^{-1/2} x = \lambda x, \tag{3.32}$$

where $x = D^{1/2} v$. The matrix $D^{-1/2} L D^{-1/2}$ is called the normalized Laplacian, denoted as $\mathcal{L}$ [4, 13]. The Rayleigh quotient in Equation (3.30) can be also rewritten as

$$\frac{p^T L p}{p^T D p} = \frac{p^T D^{1/2} (D^{-1/2} L D^{-1/2}) D^{1/2} p}{p^T D^{1/2} D^{1/2} p} = \frac{y^T \mathcal{L} y}{y^T y}, \tag{3.33}$$

where $y = D^{1/2} p$. The normalized Laplacian $\mathcal{L}$ is symmetric positive semidefinite since the Laplacian L is positive semidefinite [5].

Theorem 3.7.1 ([34])**.** *The minimum of a normalized cut can be obtained by minimizing the Rayleigh quotient of the normalized Laplacian matrix $\mathcal{L}$ in Equation* (3.33)*. It is equivalent to solving the eigenvalue problem of $\mathcal{L}$*

$$\mathcal{L} x = \lambda x. \tag{3.34}$$

If x_2 is the eigenvector of $\mathcal{L}$ corresponding to the second smallest eigenvalue λ_2, then the vector $D^{-1/2}x_2$ provides the solution of minimizing the normalized cut.

Proof. Follows immediately from Equation (3.31), (3.32) and (3.33). □

Theorem 3.7.2. *If an eigenpair (λ, v) is a solution to the generalized eigenvalue problem* (3.31)*, then the pair (λ, v) is also a solution to the generalized eigenvalue problem of the signless Laplacian M*

$$Mv = (2 - \lambda)Dv. \tag{3.35}$$

Proof. Since the Laplacian $L = D - A$ and the signless Laplacian $M = D + A$, we have $L = 2D - M$. Given a pair (λ, v) which satisfies the generalized eigenvalue problem (3.31), then

$$\begin{aligned} Lv = \lambda Dv \quad &\Rightarrow \quad (2D - M)v = \lambda Dv \\ &\Rightarrow \quad 2Dv - Mv = \lambda Dv \\ &\Rightarrow \quad (2 - \lambda)Dv = Mv \end{aligned}$$

□

Definition 3.7.1. *The normalized signless Laplacian matrix $\mathcal{M}$ is defined as $\mathcal{M} = D^{-1/2}MD^{-1/2}$.*

Theorem 3.7.3. *The minimum of normalized cut can also be obtained by solving the eigenvalue problem of the normalized signless Laplacian matrix $\mathcal{M}$*

$$\mathcal{M}x = \mu x. \tag{3.36}$$

If x_2 is the eigenvector of $\mathcal{M}$ corresponding to the second biggest eigenvalue μ_2, then the vector $D^{-1/2}x_2$ provides the solution of minimizing the normalized cut.

Proof. It is already known that the eigenvector v_2 of Equation (3.31) corresponding to the second smallest eigenvalue λ_2 provides the solution of the normalized cut. By Theorem 3.7.2, λ_2 is shifted to $2 - \lambda_2$, so v_2 is the eigenvector of Equation (3.35) corresponding to the second largest eigenvalue $2 - \lambda_2$.

If we premultiply $D^{-1/2}$ on both side of Equation (3.35), then we have

$$\begin{aligned} D^{-1/2}MD^{-1/2}D^{1/2}v &= (2-\lambda)D^{1/2}v \\ \mathcal{M}x &= (2-\lambda)x \end{aligned}$$

where $x = D^{1/2}v$. Now the solution vector v_2 can be obtained by $v_2 = D^{-1/2}x_2$, where x_2 is the eigenvector of $\mathcal{M}$ corresponding to the second largest eigenvalue $2 - \lambda_2$. □

Chapter 4

Spectral Clustering and Markov Chains

4.1 Introduction

The main purpose of this chapter is to investigate the connection between spectral clustering and Markov chains. The importance of Markov chains in modeling diverse systems, including biological, physical, social and economic systems, has long been known and is well documented. More recently, Markov chains have proven to be effective when applied to internet search engines such as Google's PageRank model [19], and in data mining applications wherein data trends are sought.

At the same time that Fiedler proposed his spectral graph partitioning method, Stewart proposed a spectral based method that can be applied to a Markov chain in order to separate its states into meaningful clusters, and which is more broadly applicable to nonsymmetric systems. The right-hand eigenvectors belonging to the dominant eigenvalues of the transition matrix of a Markov chain provide a means whereby the states of the chain can be arranged into meaningful groups [35]. This approach uses distance from stationarity as a measure of similarity among the states of a cluster.

Recently, it was observed by Meila et al. [24] that spectral clustering can be depicted in the framework of Markov random walks on a graph structure. In the transition matrix of a random walk, the right-hand eigenvectors corresponding to positive eigenvalues close to 1 have been used by several authors to obtain spectral clustering [25], such as the

Ng, Jordan and Weiss (NJW) [27] algorithm and the Meila-Shi [25] algorithm. While this may seem to unify the results of Fiedler and Stewart, this is not actually the case. Given a stochastic transition probability matrix, the subdominant eigenvalue may be negative. There is little information in the literature concerning eigenvectors which correspond to negative eigenvalues close to -1. We shall see that such eigenvectors also provide important clustering information based on the distance measure of individual state from the steady state. We shall relate the techniques obtained from the two clustering approaches in Markov chains to obtain a more comprehensive result for graph nodes.

The rest of this chapter is organized as follows. In Section 4.2 we review Stewart's results on state clustering on Markov chains. In Section 4.3 we introduce the construction of a stochastic transition probability matrix of a random walk from a graph, in preparation for spectral clustering. Then in Section 4.4, we present two clustering measures on the states of Markov chains, namely, (a) the normalized cut measure and (b) the distance measure of states from the steady state. This allows us to present a novel result concerning clustering on graph nodes that is based on incorporating both clustering techniques in Markov chains. More examples will be given in Section 4.5.

4.2 Indicators of State Clustering on Markov Chains

Spectral based methods may also be applied to a directed graph viewed as a Markov chain for clustering purposes. In a Markov chain, the state transition probability diagram can be viewed as a graph with directed edges. It is well known that when a Markov chain is irreducible and aperiodic with a finite state space, its stochastic matrix has a single eigenvalue on the unit circle, the eigenvalue 1, and the left-hand eigenvector corresponding to this unit eigenvalue is the stationary distribution vector.

In 1974 Stewart [35] proved that the right-hand eigenvectors of a stochastic matrix corresponding to the dominant eigenvalues, especially the one corresponding to the eigenvalue with the second largest modulus, which is closest to, but strictly less than, 1, provide the information necessary to cluster states into meaningful groups. His approach is based on the concept of distance of each state to the stationary distribution.

Definition 4.2.1. *The steady state or equilibrium position of a system, represented by a finite, irreducible, aperiodic Markov chain, is given by its stationary distribution vector,*

i.e., the left-hand eigenvector corresponding to the unique unit eigenvalue of the single step transition probability matrix P of the Markov chain.

When the system starts in a given state, it needs several (and often many) transition steps to reach this equilibrium position. The number of iterations, or the length of time, required to reach the equilibrium position by the system started in a given state can be regarded as the "distance" from that state to the equilibrium position. Such distance measurements serve as a means of comparison among the states and provide a criterion for clustering states.

We use the row vector

$$w_i^{(1)} = (0, 0, \ldots, 1, \ldots, 0),$$

with i^{th} component equal to 1, to denote that the system is initially in state i. We shall assume that the stochastic matrix P possesses a full set of n linearly independent eigenvectors. Let $x_1, x_2, \ldots, x_n$ be the left-hand eigenvectors of P and $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of P in descending order of their modulus, then

$$x_j^T P = \lambda_j x_j^T \quad \text{for all} \quad j = 1, 2, \ldots, n. \tag{4.1}$$

If writing $w_i^{(1)}$ as a linear combination of these eigenvectors, we have the equation

$$w_i^{(1)} = c_{i1} x_1^T + c_{i2} x_2^T + \cdots + c_{in} x_n^T \tag{4.2}$$

where $c_{i1}, c_{i2}, \ldots, c_{in}$ are the constants in the linear combination. It is known that repeated postmultiplication of $w_i^{(1)}$ by P yields convergence to the stationary distribution vector. Therefore

$$\begin{aligned} w_i^{(1)} P &= c_{i1} x_1^T P + c_{i2} x_2^T P + \cdots + c_{in} x_n^T P & (4.3) \\ &= c_{i1} x_1^T + c_{i2} \lambda_2 x_2^T + \cdots + c_{in} \lambda_n x_n^T = w_i^{(2)}, & (4.4) \end{aligned}$$

and after k transition steps,

$$w_i^{(k+1)} = c_{i1} x_1^T + c_{i2} \lambda_2^k x_2^T + \cdots + c_{in} \lambda_n^k x_n^T. \tag{4.5}$$

The system initially starts from some other state $j \neq i$ is given by, after k steps,

$$w_j^{(k+1)} = c_{j1}x_1^T + c_{j2}\lambda_2^k x_2^T + \cdots + c_{jn}\lambda_n^k x_n^T.$$

If we consider all possible starting states, we obtain

$$\begin{pmatrix} w_1^{(k+1)} \\ w_2^{(k+1)} \\ \vdots \\ w_n^{(k+1)} \end{pmatrix} = \begin{pmatrix} c_{11}x_1^T + c_{12}\lambda_2^k x_2^T + \cdots + c_{1n}\lambda_n^k x_n^T \\ c_{21}x_1^T + c_{22}\lambda_2^k x_2^T + \cdots + c_{2n}\lambda_n^k x_n^T \\ \vdots \\ c_{n1}x_1^T + c_{n2}\lambda_2^k x_2^T + \cdots + c_{nn}\lambda_n^k x_n^T \end{pmatrix}. \tag{4.6}$$

Equation (4.6) in matrix form is

$$\begin{aligned} \mathbf{W^{(k+1)}} &= \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{pmatrix} \begin{pmatrix} 1 & & & \\ & \lambda_2^k & & \\ & & \ddots & \\ & & & \lambda_n^k \end{pmatrix} \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} \\ &= CD^kX^T. \end{aligned} \tag{4.7}$$

This shows that $\lambda_2, \ldots, \lambda_n$ and $x_1^T, \ldots, x_n^T$ are fixed and only the constant coefficients, c_{ij}, differ for each $w_i^{(k+1)}$, $i = 1, 2, \ldots, n$. The difference in the number of iterations taken to reach steady state from each different possible starting state depends only on these coefficients. If λ_2 is of greater modulus than $\lambda_3, \lambda_4, \ldots$, then for large k, $\lambda_2^k \gg \lambda_j^k$ for $j \geq 3$. This means the effect of the constant terms $c_{i3}, \ldots, c_{in}$ is negligible; it is, in particular, the terms c_{i1} and c_{i2} for all $i = 1, 2, \ldots, n$ that contribute to the difference.

To calculate the matrix C, we take $k = 0$, so that the matrix

$$W^{(1)} = (w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)})^T,$$

written in terms of the set of left-hand eigenvectors, is given as

$$W^{(1)} = CX^T$$

However, since $W^{(1)}$ is the identity matrix, we have

$$I = CX^T,$$

and therefore $C = (X^T)^{-1}$. Since we assume that P has a full set of n linearly independent eigenvectors, C is, in fact, the matrix form of the set of right-hand eigenvectors of P. It is obvious that $e = (1, 1, \ldots, 1)^T$ is the right-hand eigenvector of P corresponding to the unit eigenvalue. Therefore it is the second column of C, i.e., the right-hand eigenvector of P corresponding to the eigenvalue with the second largest modulus, that provides an appropriate measure of the relative "distance" of each state from the stationary distribution. Based on this criterion, states whose corresponding component values in this vector are relatively close together form a cluster of states.

This mathematical proof demonstrates that the right-hand eigenvectors of a stochastic matrix provide information based upon which its states can be clustered. Therefore, the spectral approach can be applied not only in the symmetric case, e.g., the Laplacian and signless Laplacian matrices, but also in nonsymmetric cases under the framework of Markov chains.

4.3 State Clustering on Markov Chains and Graphs

In this section, we show the relationship between the solution of spectral clustering and the clustering result from the dominant right-hand eigenvectors of Markov chains.

4.3.1 Markov Random Walks on Graphs

The clusters of graph nodes in spectral clustering, described in Chapter 3, are based on minimum cut, while the clusters of Markov chains states are clustered according to distance from each state to the steady state. These two measures of clusters can be connected in the framework of a random walk on a graph topology. An interesting observation from Meila et al. [24] is that spectral clustering/partitioning can be depicted in the framework of Markov random walks on a graph. Solving the eigenvalue problem of the transition probability matrix of a Markov random walk can determine the normalized cut on this graph. In this section, spectral clustering is viewed as states clustering of Markov chains.

Definition 4.3.1 (**A Markov random walk on a graph** G)**.** *The transition probability* p_{ij} *from node* i *to node* j *is defined as*

$$p_{ij} = \frac{w_{ij}}{deg(i)}, \tag{4.8}$$

where w_{ij} *is the weight of the edge* $\{i, j\}$ *and* $deg(i) = \sum_j w_{ij}$ *is the vertex degree.*

Observe that, by Definition 4.3.1, the adjacency matrix A of a graph can be converted to a transition probability matrix P of a Markov chain by setting

$$P = D^{-1}A, \tag{4.9}$$

where D is a diagonal matrix whose diagonal elements are the vertex degrees. Hence P represents a random walk on a graph.

Using the Laplacian matrix $L = D - A$ and the signless Laplacian matrix $M = D + A$, we obtain the following relationships to P:

$$I - P = D^{-1}L \quad \text{and} \quad I + P = D^{-1}M. \tag{4.10}$$

Theorem 4.3.1. *If an eigenpair* (λ, v) *is a solution to the eigenvalue problem* $Pv = \lambda v$*, where* P *is given by Equation* (4.9)*, then the pair* (λ, v) *is also a solution to the generalized eigenvalue problems* $(1 - \lambda)Dv = Lv$ *and* $(1 + \lambda)Dv = Mv$*.*

Proof. Given a pair (λ, v) which satisfies the eigenvalue problem $Pv = \lambda v$, then, since $P = D^{-1}A$ and $L = D - A$, we have [24]

$$\begin{aligned} Pv = \lambda v \quad &\Rightarrow \quad D^{-1}Av = \lambda v \quad \Rightarrow \quad D^{-1}(D - L)v = \lambda v \\ &\Rightarrow \quad Iv - D^{-1}Lv = \lambda v \\ &\Rightarrow \quad (1 - \lambda)Dv = Lv. \end{aligned} \tag{4.11}$$

It is the same for the signless Laplacian matrix $M = D + A$,

$$\begin{aligned} Pv = \lambda v \quad &\Rightarrow \quad D^{-1}Av = \lambda v \quad \Rightarrow \quad D^{-1}(M - D)v = \lambda v \\ &\Rightarrow \quad D^{-1}Mv - Iv = \lambda v \\ &\Rightarrow \quad (1 + \lambda)Dv = Mv. \end{aligned} \tag{4.12}$$

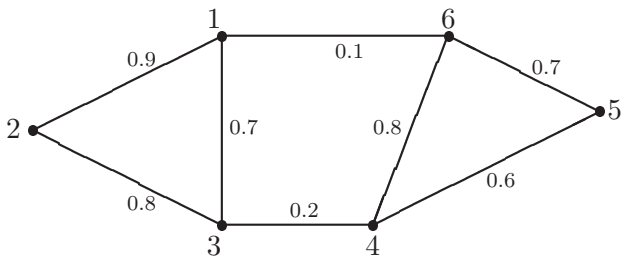

Figure 4.1: A sample graph with six vertices.

□

It is already known that the eigenvectors of the generalized eigenvalue problem $(1-\lambda)Dv = Lv$ give a heuristic solution to the balanced minimum cut problem of a graph [34, 7]. Therefore, from Theorem 4.3.1, we see that the right-hand eigenvectors of P also provide a balanced cut solution, the same as the eigenvectors of the generalized eigenvalue problem on M.

Example 4.3.1. Example 3.3.1 displays a graph with six vertices.

After constructing the matrix P of a random walk on this graph, we have

$$\mathbf{P} = \begin{pmatrix} 0 & 0.5294 & 0.4118 & 0 & 0 & 0.0588 \\ 0.5294 & 0 & 0.4706 & 0 & 0 & 0 \\ 0.4118 & 0.4706 & 0 & 0.1176 & 0 & 0 \\ 0 & 0 & 0.1250 & 0 & 0.3750 & 0.5000 \\ 0 & 0 & 0 & 0.4615 & 0 & 0.5385 \\ 0.0625 & 0 & 0 & 0.5000 & 0.4375 & 0 \end{pmatrix}.$$

The eigenvalues λ of P, in decreasing order, are

$$\lambda = \begin{pmatrix} 1 & 0.8815 & -0.3300 & -0.4435 & -0.5651 & -0.5429 \end{pmatrix},$$

and the corresponding right-hand eigenvectors v are the columns

$$\mathbf{v} = \begin{pmatrix} 0.4082 & 0.3807 & -0.4196 & -0.3253 & 0.4452 & 0.3832 \\ 0.4082 & 0.4131 & -0.1793 & 0.1126 & -0.2099 & -0.7335 \\ 0.4082 & 0.3455 & 0.5978 & 0.2598 & -0.2487 & 0.4150 \\ 0.4082 & -0.3962 & 0.5090 & -0.2914 & 0.4763 & -0.3226 \\ 0.4082 & -0.4711 & -0.3583 & 0.7647 & 0.2267 & 0.1162 \\ 0.4082 & -0.4315 & -0.2167 & -0.3800 & -0.6462 & 0.1593 \end{pmatrix}.$$

The K-means clustering method on vector v_2 corresponding to the second largest eigenvalue $\lambda_2 = 0.8815$ provides the balanced minimum cut solution $V_1 = \{1, 2, 3\}$ and $V_2 = \{4, 5, 6\}$.

Now let us consider the problem of determining clusters from P, the probability matrix of the random walk on a graph. Spectral clustering using P is based on the balanced minimum cut, while state clustering on a Markov chain of P is based on a distance measure from each state to the steady state. Now a question rises. Do these two kinds of clustering results coincide in the right eigenvectors of P? The answer is not a simple "yes" or "no"; instead it depends on the eigenvalues of P. State clustering on Markov chains proposed by Stewart uses the right eigenvector of P corresponding to the eigenvalue with modulus closest to 1. If the eigenvalue closest to the unit circle is positive, then it will be shifted to the eigenvalue closest to 0 in the generalized eigenvalue problem of L in Equation (4.11) and the corresponding eigenvector gives the balanced minimum cut on the graph. In this case the two clustering results coincide. However if a negative eigenvalue of P is closest to the unit circle, then it will be shifted to the eigenvalue closest to 2 in Equation (4.11), so the corresponding eigenvector will not give the clustering information based on the balanced minimum cut of a graph. Now the two clustering results can be different.

4.3.2 Some Interesting Facts

Before we discuss the details of clustering information given by the right-hand eigenvectors of P, let us first look at some interesting facts concerning P. These properties are true only for matrices P arising in the context of random walks on graphs as in Equation (4.9), and not for all transition probability matrices.

Theorem 4.3.2. *All the eigenvalues of a probability matrix $P = D^{-1}A$ arising in the context of a random walk on a graph are real.*

Proof. $P = D^{-1}A$ has a symmetric structure since D is diagonal and A, the adjacency matrix, is symmetric. Also, since

$$D^{1/2}PD^{-1/2} = D^{-1/2}AD^{-1/2},$$

P and the symmetric matrix $D^{-1/2}AD^{-1/2}$ are similar. Similar matrices share the same eigenvalues. Therefore all eigenvalues of P are real. □

The Theorem 4.3.2 suggests an alternative way to calculate the eigenvalues of P, given that it is easier to compute the eigenvalues of a symmetric matrix $D^{-1/2}AD^{-1/2}$ than a nonsymmetric P.

Result 4.3.1. *Let λ be an eigenvalue of P generated from a random walk on a graph, and let x_R and x_L be its corresponding right and left eigenvectors respectively, then*

$$x_R = D^{-1/2}v \quad \textit{and} \quad x_L = D^{1/2}v, \tag{4.13}$$

where v is the eigenvector of the symmetric matrix $D^{-1/2}AD^{-1/2}$ corresponding to the eigenvalue λ.

Proof. We have

$$Px_R = \lambda x_R \quad \text{and} \quad P^T x_L = \lambda x_L.$$

First, let us focus on x_R. Observe that

$$Px_R = \lambda x_R \quad \Rightarrow \quad D^{-1}Ax_R = \lambda x_R.$$

Now, if we premultiply by $D^{1/2}$ on both sides, we have

$$D^{-1/2}AD^{-1/2}(D^{1/2}x_R) = \lambda(D^{1/2}x_R).$$

Therefore, the value λ and the vector $D^{1/2}x_R$ are an eigenvalue/vector pair of $D^{-1/2}AD^{-1/2}$. Second, let us move to x_L. Observe that, since A is symmetric,

$$P^T x_L = \lambda x_L \quad \Rightarrow \quad AD^{-1}x_L = \lambda x_L.$$

Now if we premultiply $D^{-1/2}$ on both sides, we have

$$D^{-1/2}AD^{-1/2}(D^{-1/2}x_L) = \lambda(D^{-1/2}x_L).$$

Therefore, the value λ and the vector $D^{-1/2}x_L$ is an eigenvalue and corresponding eigenvector of $D^{-1/2}AD^{-1/2}$ respectively.

Thus x_R and x_L can be obtained from calculating the eigenvectors of $D^{-1/2}AD^{-1/2}$. Let v be the eigenvector of $D^{-1/2}AD^{-1/2}$ corresponding to the eigenvalue λ. Then

$$x_R = D^{-1/2}v \quad \text{and} \quad x_L = D^{1/2}v.$$

□

Definition 4.3.2. *The multiplication symbol $\times$ of two same dimensional vectors $a \times b$ denotes element-by-element multiplication of a and b; thus $a \times b$ is a vector with i^{th} element equal to $a_i \cdot b_i$ for $i = 1, \ldots, n$.*

Result 4.3.2. *Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the n eigenvalues of P, and the corresponding $x_{R_1}, x_{R_2} \ldots x_{R_n}$ and $x_{L_1}, x_{L_2} \ldots x_{L_n}$ be the n right and left eigenvectors of P respectively, then*

$$x_{L_i} = x_{R_i} \times \mu \quad \textit{for} \quad i = 1 \ldots n, \tag{4.14}$$

where μ is the stationary distribution of P. A more generalized form is given by

$$x_{R_i} \times x_{L_j} = x_{R_j} \times x_{L_i} \quad \textit{for} \quad i = 1 \ldots n. \tag{4.15}$$

Proof. Let d_i be the vertex degree on a diagonal position of the diagonal matrix D associated with vertex i, then we have $d = [d_1, d_2 \dots d_n]^T$. Also $x_{R_1} = e = [1, 1 \dots 1]^T$ and $x_{L_1} = \mu$, therefore by Result 4.3.1:

$$x_{R_1} = D^{-1/2} v_1 \quad \text{and} \quad x_{L_1} = D^{1/2} v_1,$$

where v_1 is the eigenvector of $D^{-1/2} A D^{-1/2}$ associated with λ_1. It is easy to show that $D^{-1/2} v_1 = d^{-1/2} \times v_1$ and $D^{1/2} v_1 = d^{1/2} \times v_1$. Similarly,

$$x_{R_2} = D^{-1/2} v_2 = d^{-1/2} \times v_2 \quad \text{and} \quad x_{L_2} = D^{1/2} v_2 = d^{1/2} \times v_2.$$

Then

$$x_{R_1} \times x_{L_2} = d^{-1/2} \times v_1 \times d^{1/2} \times v_2 = d^{-1/2} \times v_2 \times d^{1/2} \times v_1 = x_{R_2} \times x_{L_1}.$$

Since $x_{R_1} = e$ and $x_{L_1} = \mu$, we have

$$x_{L_2} = x_{R_2} \times \mu.$$

This statement can be generalized to

$$x_{L_i} = x_{R_i} \times \mu \quad \text{for} \quad i = 1 \dots n.$$

In fact, it is not hard to justify the more generalized form

$$x_{R_i} \times x_{L_j} = x_{R_j} \times x_{L_i},$$

since

$$x_{R_i} \times x_{L_j} = d^{-1/2} \times v_i \times d^{1/2} \times v_j = d^{-1/2} \times v_j \times d^{1/2} \times v_i = x_{R_j} \times x_{L_i}$$

□

Example 4.3.2. An example of the adjacency matrix A is given by

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Using $P = D^{-1}A$ we obtain the probability matrix P:

$$\mathbf{P} = \begin{pmatrix} 0 & 1/3 & 1/3 & 0 & 0 & 1/3 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/3 & 1/3 & 0 \end{pmatrix}.$$

Then the eigenvalues and right, left eigenvectors of P are

$$\lambda = \begin{pmatrix} 1 & 0.5774 & 0 & -0.3333 & -0.6667 & -0.5774 \end{pmatrix},$$

$$\mathbf{x_R} = \begin{pmatrix} 0.4082 & 0.3162 & 0.5000 & -0.2132 & 0.5000 & -0.3162 \\ 0.4082 & 0.5477 & 0.0000 & 0.6396 & -0.0000 & 0.5477 \\ 0.4082 & 0.3162 & -0.5000 & -0.2132 & -0.5000 & -0.3162 \\ 0.4082 & -0.3162 & -0.5000 & -0.2132 & 0.5000 & 0.3162 \\ 0.4082 & -0.5477 & -0.0000 & 0.6396 & 0.0000 & -0.5477 \\ 0.4082 & -0.3162 & 0.5000 & -0.2132 & -0.5000 & 0.3162 \end{pmatrix},$$

$$\mathbf{x_L} = \begin{pmatrix} 0.4523 & 0.3873 & -0.5000 & -0.2887 & 0.5000 & 0.3873 \\ 0.3015 & 0.4472 & 0.0000 & 0.5774 & -0.0000 & -0.4472 \\ 0.4523 & 0.3873 & 0.5000 & -0.2887 & -0.5000 & 0.3873 \\ 0.4523 & -0.3873 & 0.5000 & -0.2887 & 0.5000 & -0.3873 \\ 0.3015 & -0.4472 & -0.0000 & 0.5774 & 0.0000 & 0.4472 \\ 0.4523 & -0.3873 & -0.5000 & -0.2887 & -0.5000 & -0.3873 \end{pmatrix}.$$

First we check that $x_{L_i} = x_{R_i} \times \mu$. For example

$$x_{R_4} \times x_{L_1} = cx_{L_4} = \begin{pmatrix} -0.0964 \\ 0.1928 \\ -0.0964 \\ -0.0964 \\ 0.1928 \\ -0.0964 \end{pmatrix} \quad \text{where } c = 0.334 \text{ is a constant.}$$

Second we check that $x_{R_i} \times x_{L_j} = x_{R_j} \times x_{L_i}$. Again

$$x_{R_4} \times x_{L_6} = cx_{L_4} \times x_{R_6} = \begin{pmatrix} -0.0826 \\ -0.2860 \\ -0.0826 \\ 0.0826 \\ 0.2860 \\ 0.0826 \end{pmatrix} \quad \text{where } c = -0.9045 \text{ is a constant.}$$

4.4 Clustering Techniques for Markov Chains

After generating a random walk on a graph by setting $P = D^{-1}A$, we can perform a clustering analysis on P. If the Markov chain is ergodic, then spectral decomposition on the transition probability matrix P provides two clustering measures of graph nodes explained as below.

The first measure is based on the balanced minimum cut. If we sort the eigenvalues of the probability matrix P in descending order, the eigenvector associated with the second largest (positive) eigenvalue, which is shifted to become the second smallest one in the generalized eigenvalue problem Equation (4.11) and the second largest one in Equation (4.12), gives clusters of states based on the balanced minimum cut. There is much research already available using the eigenvectors of P corresponding to positive eigenvalues to perform spectral clustering [25].

The second measure is based on the "distance" from each state to the steady state. If we sort the *modulus* of the eigenvalues in descending order, the eigenvector associated

with the second largest one (not necessarily positive) gives clusters of states based on this distance measure. If the second largest one is positive, then both clustering measures coincide; otherwise they are different. Previous research paid more attention to the first and less with the second, because clustering using the first measure is very close to the structure of real world networks, while the second is not. However the second measure is also related to the graph structure and we cannot simply ignore its information.

Let us focus on the case of when the eigenvalue having the second largest modulus is negative. How should we identify clustering information from its corresponding eigenvector? Two aspects need to be considered. The first is the *modulus* of elements in this eigenvector. Elements with small modulus in the eigenvector indicate the corresponding states are close to the steady state; alternatively, they are in a group of states in which states are more closely linked with each other. On the other hand, elements with large modulus indicate the corresponding states are far from the steady state and belong to a group in which states are not closely linked with each other. The second aspect is the numeric value of elements in this eigenvector. If a value in one group of elements is closer to another group than other values in the same group, this means the corresponding state has more chance to link states in another group than other states in the same group.

A concrete example should help understanding these clustering techniques for Markov chains.

Example 4.4.1. The figure below is an unweighted graph with 11 vertices. After formulating the matrix P of a random walk on this graph, its second largest eigenvalue is found to be $\lambda_1 = 0.8852$ and the eigenvalue with second largest modulus is $\lambda_2 = -0.9336$. The corresponding eigenvectors are v_1 and v_2.

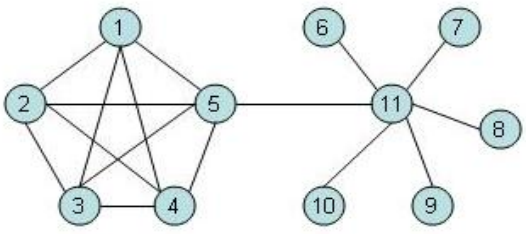

Figure 4.2: Joint complete and star graph.

$$\mathbf{v_1}, \mathbf{v_2} = \begin{pmatrix} -0.1423 & 0.0105 \\ -0.1423 & 0.0105 \\ -0.1423 & 0.0105 \\ -0.1423 & 0.0105 \\ -0.0770 & -0.0707 \\ 0.2581 & -0.3085 \\ 0.2581 & -0.3085 \\ 0.2581 & -0.3085 \\ 0.2581 & -0.3085 \\ 0.2581 & -0.3085 \\ 0.2285 & 0.2880 \end{pmatrix}.$$

Based on the first clustering measure from v_1, we obtain cluster 1 {vertices 1,...,5} and cluster 2 {vertices 6,...,11}. Moving to the second clustering measure, we make two observations:

(a) Eigenvector v_2 shows that the first five elements associated with cluster 1 have relatively small modulus while others associated with cluster 2 have relatively large modulus. This means that cluster 1 has a structure in which states are closely linked between each other, while cluster 2 has the opposite structure. This observation can be confirmed by the figure. The left part of figure is a clique and right part is a star.

(b) If we sort the values of elements in v_2, then the value of vertex 5 is closer to cluster 2 than other vertices in cluster 1 while the value of vertex 11 is closer to cluster 1 than other vertices in cluster 2. This is reasonable, because vertex 5 and 11 have the possibility of transitioning to the other cluster in a single step; they are the connecting vertices between clusters. This can also be confirmed by the observation that vertex 5 in v_2 has a modulus which is relatively large compared with other vertices in cluster 1; vertex 11 has relatively smaller modulus in cluster 2.

Therefore, the eigenvector associated with a negative eigenvalue whose modulus is very close to 1 indicates firstly, the cluster structure and secondly, the role or position of states (graph nodes) in each cluster. When we perform a clustering analysis on P, it

is appropriate to combine both clustering techniques, using not only the eigenvectors with positive eigenvalues but also those with negative eigenvalues to obtain more comprehensive information concerning each cluster of states.

There are two applications on Markov chains associated with these two types of clustering techniques. One is the set of nearly completely decomposable (NCD) Markov chains, and the other is the set of nearly periodic Markov chains. In NCD Markov chains, there may be one or more positive eigenvalues very close to 1. In this case, the eigenvectors corresponding to such eigenvalues provide the first measure of clustering information. In nearly periodic Markov chains, there may be one or more negative eigenvalues with modulus very close to 1. In this case, the corresponding eigenvectors provide the second measure of clustering information. The advantage of using the transition probability matrix P of the Markov chain to perform clustering analysis on graph nodes is that the spectral decomposition of P not only provides the clusters of nodes in balanced minimum cut which are the same from the Laplacian and the signless Laplacian matrix, but also provides additional clustering information concerning each cluster's structure and the role or position of graph nodes in each cluster.

4.5 More Examples

Example 4.5.1. For some basic graphs, it does not make sense to cluster graph nodes, e.g. cycle and complete graph, although the spectral clustering method on them also gives the solution of normalized cut (balanced minimum cut). Let us first examine the spectrum of the probability matrix P of Markov random walks on some basic graphs. Studying their spectrums can help us to recognize patterns.

1. **The cycle with n vertices (n-cycle).** The eigenvalues of P formulated on n-cycle are

$$\lambda_k = \cos\frac{2(k-1)\pi}{n} \qquad k = 1, \ldots, n.$$

   No matter whether n is odd or even, it makes no sense to cluster vertices on a cycle. When n is even, the P has eigenvalue -1, which means that the Markov chain is

periodic, so the second clustering measure is not available; when n is odd, it is not periodic.

2. **The path with n vertices (n-path).** The eigenvalues of P are

$$\lambda_k = \cos\frac{(k-1)\pi}{n-1} \qquad k = 1, \ldots, n.$$

P has eigenvalue -1 and the Markov chain is periodic. The second clustering measure is not available.

3. **The star graph with n vertices.** The eigenvalues of P are

$$\lambda_1 = 1, \quad \lambda_n = -1, \quad \lambda_k = 0 \ \text{ for } \ k = 2, \ldots, n-1.$$

The star graph possesses a structure that should not be divided. The P here is periodic.

4. **The complete graph with n vertices.** The eigenvalues of P

$$\lambda_1 = 1, \quad \lambda_k = -\frac{1}{n-1} \ \text{ for } \ k = 2, \ldots, n.$$

The complete graph also possesses a structure that should not be divided, so it makes no sense to cluster the states of such a graph.

5. **The Petersen graph.** It is a famous graph with 10 vertices in graph theory and is illustrated in Figure 4.3. The eigenvalues of P are

$$\lambda_1 = 1, \quad \lambda_i = \frac{1}{3} \ \text{ for } \ i = 2, \ldots, 6, \quad \lambda_j = -\frac{2}{3} \ \text{ for } \ j = 7, \ldots, 10.$$

The Petersen graph also possesses a structure that should not be divided.

6. **Two cycles with m and n vertices connected by one edge.** The first clustering measure works well on this kind of graph and we obtain two clusters (cycles), no matter what values of m and n are chosen. When both m and n are even, the Markov chain is periodic, and the clustering result is based on the normalized cut.

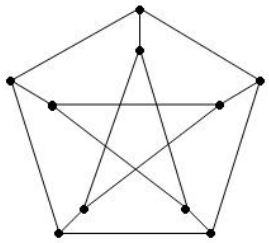

Figure 4.3: Peterson graph

7. **Two cycles with m and n vertices connected by one center.** If m and n are both even, then the Markov chain is periodic. In this case, the eigenvalues of P are in $\pm$ pair and at least one is zero. No matter what m and n are, the eigenvector corresponding to the second largest eigenvalue provides two clusters (cycles). If $m = n$, then the element of this eigenvector corresponding to the center is zero, else if $m \neq n$, then it is not zero but very small because of its central position in the network.

8. **Two stars with m and n vertices connected by one center.** No matter what m and n are, the Markov chain is periodic. The eigenvalues of P are

$$\lambda_1 = 1, \quad \lambda_2 = a, \quad \lambda_{m+n} = -a, \quad \lambda_{m+n+1} = -1,$$
$$\lambda_k = 0 \quad \text{for} \quad k = 3, \ldots, m+n-1,$$

   where a is a positive value close to 1. The eigenvector corresponding to λ_2 provides two clusters (stars). If $m = n$, then the element of this eigenvector corresponding to the center is zero, otherwise it is not zero but very small.

9. **Two complete graphs with m and n vertices connected by one center.** No matter what m and n are, the Markov chain is not periodic. The eigenvector corresponding to the second largest eigenvalue of P provides two clusters (cliques). If $m = n$, then the element of this eigenvector corresponding to the center is zero, otherwise it is not zero but very small.

10. **An m-complete graph connected with an n-cycle by a center or connected with an n-star by a center.** The Markov chain is not periodic. The eigenvector

corresponding to the second largest eigenvalue of P provides two clusters. The element of this eigenvector corresponding to the center is small. If the eigenvalue with second largest modulus is negative, then the elements of the corresponding eigenvector that are associated with nodes in m-complete graph have small modulus, and the elements associated with nodes in n-cycle or star have large modulus, and the element associated with the center node is in the middle.

Example 4.5.2. This is a graph with eight vertices in Figure 4.4. After formulating the P of this graph, its second largest eigenvalue is $\lambda_2 = 0.8206$. Therefore we use v_2 to cluster graph vertices into two groups. Since $\lambda_8 = -0.9455$ has the second largest modulus, its corresponding eigenvector v_8 should be analyzed.

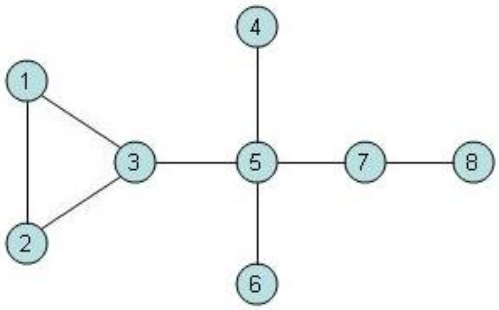

Figure 4.4: An example graph with eight vertices.

$$\mathbf{v_2}, \mathbf{v_8} = \begin{pmatrix} 0.4230 & -0.0630 \\ 0.4230 & -0.0630 \\ 0.2712 & 0.1821 \\ -0.2173 & 0.4130 \\ -0.1783 & -0.3905 \\ -0.2173 & 0.4130 \\ -0.4219 & 0.4686 \\ -0.5142 & -0.4957 \end{pmatrix}.$$

The K-means method on v_2 gives cluster 1 {1,2,3} and cluster 2 {4,5,6,7,8}. The first three elements in v_8 have small modulus and last five have large modulus. This means cluster 1 has a structure in which vertices are closely linked with each other, but not for cluster

2. Vertex 3 in v_8 has relative larger modulus than those in cluster 1; and vertex 5 in v_8 has relative small modulus relative to others in cluster 2, because vertex 3 and 5 are the connecting points.

Example 4.5.3. This is a graph with eleven vertices in Figure 4.5. After formulating the P of this graph, $\lambda_2 = 0.9328$ is its second largest eigenvalue. Therefore we use v_2 to cluster vertices into two groups. $\lambda_{11} = -0.9653$ has the second largest modulus and its corresponding eigenvector is v_{11}.

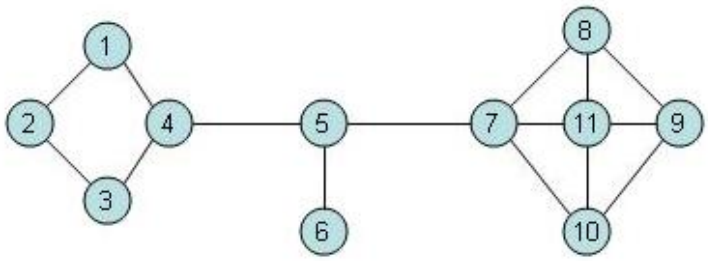

Figure 4.5: An example graph with eleven vertices.

$$\mathbf{v_2}, \mathbf{v_{11}} = \begin{pmatrix} -0.2700 & 0.3132 \\ -0.2895 & -0.3245 \\ -0.2700 & 0.3132 \\ -0.2142 & -0.2802 \\ -0.0595 & 0.1849 \\ -0.0638 & -0.1916 \\ 0.1116 & -0.0637 \\ 0.1578 & 0.0264 \\ 0.1700 & -0.0211 \\ 0.1578 & 0.0264 \\ 0.1601 & 0.0083 \end{pmatrix}.$$

The K-means method on v_2 gives cluster 1 {1,2,3,4,5,6} and cluster 2 {7,8,9,10,11}. The first six elements in v_{11} corresponding to cluster 1 have large modulus and last five elements have small modulus. This makes sense because cluster 1 has a relative sparse structure while cluster 2 has a relative dense structure.

In this chapter, we present two clustering techniques of Markov chains. One technique is spectral clustering in the framework of a Markov random walk generated on a graph. The clustering of this technique is based on a measure of normalized cut. We examine some interesting facts that are true only for transition probability matrices arising in the context of random walks on graphs. The other technique is using dominant eigenvectors of a probability matrix to cluster states into meaningful groups based on a distance measure of states from the steady state.

Chapter 5

Spectral and Statistical Clustering

The data with which we deal in spectral clustering is an $n \times m$ matrix corresponding to a graph with n nodes and m edges. The spectral decomposition of this matrix is a preprocessing step to project the high-dimensional representation of nodes into a lower dimension. The lower dimensional representation of graph nodes may be an eigenvector or a singular vector.

In multivariate statistical analysis, we analyze the multivariate random variable X, which is composed of p one-dimensional random variables:

$$X = (X_1, X_2, \ldots, X_p).$$

Given n observations of X, each observation x_i has p dimensions:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}),$$

and is a vector in $\mathcal{R}^p$. Therefore the dataset of X in high dimensions can be described in an $n \times p$ matrix $\mathcal{X}$. There are some dimension reduction techniques available for analyzing $\mathcal{X}$, such as the factorial method and principal components analysis. These techniques also involve the eigendecomposition of a matrix. Hence we have some justification in thinking that spectral clustering may be related to these techniques in statistical clustering. Later in this chapter, we shall see that spectral clustering has a connection to the factorial method and principal components analysis.

In Section 5.1, we first go through the details of the factorial method of data

matrices, then we discuss the commonalities and differences between spectral clustering and the factorial method in Section 5.2. The idea of principal components analysis will be reviewed in Section 5.3. Finally, in Section 5.4, we relate the clustering result from principal components analysis to that from spectral clustering.

5.1 Factorial Method of Data Matrices

The idea of the factorial method is to reduce the dimension of a multivariate data set in the sense of a least-squares criterion. This involves the decomposition of data matrices by factors. We can then perform the clustering analysis on the lower-dimensional representation of the data set.

The data matrix $\mathcal{X}$ composed of n observations of p variables can be viewed in two ways:

1. Each observation (row by row) is a vector ${x_i}^T = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathcal{R}^p$. Therefore matrix $\mathcal{X}$ is represented by n points in p dimensions.

2. Each variable (column by column) is a vector $x_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^T \in \mathcal{R}^n$. Therefore matrix $\mathcal{X}$ is represented by p points in n dimensions.

5.1.1 Analysis of n Observations in $\mathcal{R}^p$

We first consider each row of $\mathcal{X}$, i.e., $\mathcal{X}$ is represented by n points in $\mathcal{R}^p$. The factorial method is to project each point $x_i, i = 1, \ldots, n$, into a space of lower dimension.

Let us begin at finding a subspace of dimension 1. This means we want to find a straight line L_1 through the origin with all points are projected onto this line in the least-squares sense. To find this line, we define its direction as a vector $d_1 \in \mathcal{R}^p$ and $\|d_1\| = 1$. The projection point of a point $x_i \in \mathcal{R}^p$ on L_1 is defined as $p_{x_i} \in \mathcal{R}^p$. Hence, this problem is translated into finding a d_1 that minimizes

$$\sum_{i=1}^{n} \|x_i - p_{x_i}\|^2. \tag{5.1}$$

Since $\|x_i - p_{x_i}\|$ is the distance between points x_i and p_{x_i}, and p_{x_i} is the projection point of x_i on L_1, then by the Pythagorean theorem,

$$\|x_i - p_{x_i}\|^2 = \|x_i\|^2 - \|p_{x_i}\|^2. \tag{5.2}$$

This equation is illustrated in Figure 5.1 with a space of dimension 3. Now the minimization

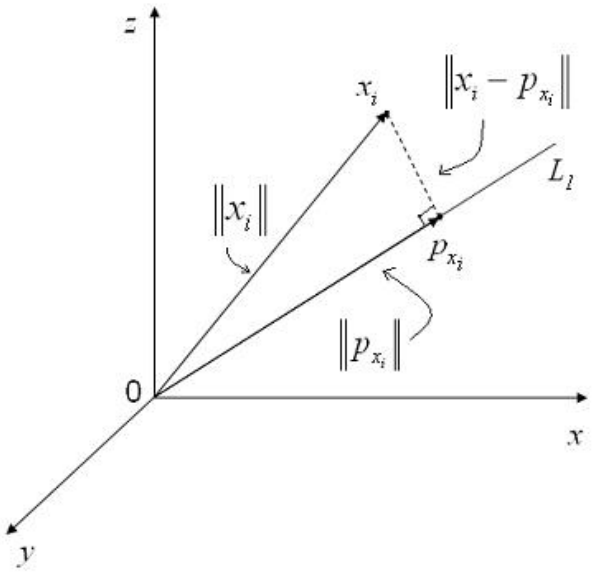

Figure 5.1: Illustration in 3-dimensional space.

problem of (5.1) is equivalent to the maximization of $\sum_{i=1}^{n} \|p_{x_i}\|^2$. If the angle between vector x_i and line L_1 is θ, then we have

$$\cos\theta = \frac{{x_i}^T d_1}{\|x_i\|\|d_1\|}.$$

Since $\|d_1\| = 1$,

$$\|p_{x_i}\| = \|x_i\||\cos\theta| = |{x_i}^T d_1|.$$

For n observations, we write

$$\begin{pmatrix} \|p_{x_1}\| \\ \|p_{x_2}\| \\ \vdots \\ \|p_{x_n}\| \end{pmatrix} = \begin{pmatrix} |{x_1}^T d_1| \\ |{x_2}^T d_1| \\ \vdots \\ |{x_n}^T d_1| \end{pmatrix}.$$

Since

$$\begin{pmatrix} {x_1}^T d_1 \\ {x_2}^T d_1 \\ \vdots \\ {x_n}^T d_1 \end{pmatrix} = \mathcal{X} d_1,$$

the maximization of $\sum_{i=1}^{n} \|p_{x_i}\|^2$ is reformulated to maximize the quadratic form $(\mathcal{X}d_1)^T(\mathcal{X}d_1)$:

$$\max_{\|d_1\|=1} {d_1}^T \mathcal{X}^T \mathcal{X} d_1. \tag{5.3}$$

$\mathcal{X}^T\mathcal{X}$ is a symmetric matrix. Theorem 2.2.5 provides the solution of this maximization problem.

Theorem 5.1.1 ([17]). *The maximum of Equation* (5.3) *is given by the largest eigenvalue* λ_1 *of the matrix* $\mathcal{X}^T\mathcal{X}$*, and the associated vector* d_1 *is the eigenvector* v_1 *of* $\mathcal{X}^T\mathcal{X}$ *corresponding to* λ_1*.*

The straight line L_1 given by the direction $v_1 \in \mathcal{R}^p$ (the first eigenvector of $\mathcal{X}^T\mathcal{X}$) is what is needed to project the data set $\mathcal{X}$ into dimension 1 using the least-square criterion (5.1). The projection coordinates of n points onto L_1 are $\mathcal{X}v_1$. In the factorial method, $\mathcal{X}v_1$ and v_1 are called the first factorial variable (or the first factor) and the first factorial axis respectively [17]. For each observation x_i, and $v_1 = (v_{11}, v_{21}, v_{31}, \ldots, v_{p1})^T$ the first factor is a linear combination of the original p variables

$$x_{i1}v_{11} + x_{i2}v_{21} + x_{i3}v_{31} + \cdots + x_{ip}v_{p1}.$$

The first factor contains the most important information of data. We can perform a clustering analysis, e.g., K-means, on the first factor to cluster n points (observations).

Suppose now we want to find a subspace of dimension 2, i.e. a plane through the origin, to fit the n points in $\mathcal{R}^p$. A plane through the origin can be determined by two orthogonal unit vector d_1 and d_2 with $\|d_1\| = 1$ and $\|d_2\| = 1$ (d_1 and d_2 here represent the directions of two straight lines). If we set the coordinates of the projection point p_{x_i} of each point x_i onto this plane (onto directions d_1 and d_2) are p_{i1} and p_{i2}, then we have

$$p_{i1} = {x_i}^T d_1 \quad \text{and} \quad p_{i2} = {x_i}^T d_2. \tag{5.4}$$

Based on the previous discussion of least-square criterion, we want to find d_1 and d_2 so as to minimize the function (5.1). This is equivalent to maximizing $\sum_{i=1}^{n} \|p_{x_i}\|^2$. From Equation (5.4) and

$$\|p_{x_i}\|^2 = p_{i1}^2 + p_{i2}^2,$$

from the Pythagorean theorem, we may write in matrix form

$$\begin{aligned} \sum_{i=1}^{n} \|p_{x_i}\|^2 &= \sum_{i=1}^{n} p_{i1}^2 + \sum_{i=1}^{n} p_{i2}^2 \\ &= (\mathcal{X}d_1)^T(\mathcal{X}d_1) + (\mathcal{X}d_2)^T(\mathcal{X}d_2). \end{aligned} \tag{5.5}$$

The problem can now be formulated as

$$\begin{aligned} &\text{Maximize} && {d_1}^T(\mathcal{X}^T\mathcal{X})d_1 + {d_2}^T(\mathcal{X}^T\mathcal{X})d_2 \\ &\text{such that} && \|d_1\| = \|d_2\| = 1 \quad \text{and} \quad {d_1}^T d_2 = 0. \end{aligned} \tag{5.6}$$

Theorem 5.1.2 ([17]). *The vectors d_1 and d_2 that maximize problem* (5.6) *are the eigenvectors v_1 and v_2 of $\mathcal{X}^T\mathcal{X}$ corresponding to the largest and second largest eigenvalues λ_1 and λ_2 of $\mathcal{X}^T\mathcal{X}$ respectively. The maximum of* (5.6) *is $\lambda_1 + \lambda_2$.*

The unit vector v_2 represents a second line L_2. The projection coordinates of n points onto a 2-dimensional plane (onto L_1 and L_2) are $\mathcal{X}v_1$ and $\mathcal{X}v_2$. Eigenvector v_2 is called the second factorial axis, and $\mathcal{X}v_2$ is called the second factor. The data of n observations in $\mathcal{R}^p$ is reduced to $\mathcal{R}^2$. The clustering algorithms can be applied on the first and the second factors together to cluster the n observations.

To project n observations of data into a q-dimensional subspace ($q < p$) in the sense of least-square criterion, the best subspace we want is obtained by the first q orthogonal eigenvectors $v_1, v_2, \ldots, v_q$ of $\mathcal{X}^T\mathcal{X}$ corresponding to the first q largest eigenvalues

$\lambda_1, \lambda_2, \ldots, \lambda_q$. This follows the same argument as before. The projection coordinates of n points onto q-dimensional subspace are $\mathcal{X}v_1, \mathcal{X}v_2, \ldots, \mathcal{X}v_q$. Eigenvector v_i is the ith factorial axis and $\mathcal{X}v_i$ is called the ith factor for $i = 1, \ldots, q$.

5.1.2 Analysis of p Variables in $\mathcal{R}^n$

The data matrix $\mathcal{X}$ can also be viewed as p points (variables) in $\mathcal{R}^n$. Each point is $x_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^T$. To analyze these p variables or carry out the clustering on them, we want to reduce the dimension of each point, i.e., to project each point into a subspace of lower dimension in the least-square sense. The method here is the same as above.

If we want to find a straight line (dimension 1) to fit p points, we set the direction of this line as $d_1 \in \mathcal{R}^n$ with $\|d_1\| = 1$. Then the coordinate of the projection p_{x_j} of point x_j on this line is ${x_j}^T d_1$. To obtain the least square of (5.1) is equivalent to finding a d_1 such that

$$\sum_{j=1}^{p} \|p_{x_j}\|^2 = \sum_{j=1}^{p} ({x_j}^T d_1)^2$$

is maximized. Since

$$\sum_{j=1}^{p} ({x_j}^T d_1)^2 = (\mathcal{X}^T d_1)^T (\mathcal{X}^T d_1),$$

this is equivalent to finding a unit vector d_1 so as to maximize ${d_1}^T (\mathcal{X}\mathcal{X}^T) d_1$.

Theorem 5.1.3 ([26]). *The unit vector d_1 maximizing ${d_1}^T (\mathcal{X}\mathcal{X}^T) d_1$ is the eigenvector u_1 of $\mathcal{X}\mathcal{X}^T$ corresponding to the largest eigenvalue of $\mathcal{X}\mathcal{X}^T$.*

The coordinates of p points on this line are $\mathcal{X}^T u_1$, called the first factor. The first factor of each variable x_j in $\mathcal{R}^n$ is a linear combination of the original coordinate of x_j with coefficients by the vector $u_1 = (u_{11}, u_{21}, u_{31}, \ldots, u_{n1})^T$:

$$x_{1j}u_{11} + x_{2j}u_{21} + x_{3j}u_{31} + \cdots + x_{nj}u_{n1}.$$

Finally the first factor can be the input to clustering algorithms.

To fit p variables into a q-dimensional subspace ($q < n$), following the same argument, the best subspace in the least square sense is given by the first q eigenvectors $u_1, u_2, \ldots, u_q$ of $\mathcal{X}\mathcal{X}^T$ corresponding to the first q largest eigenvalues. $\mathcal{X}^T u_k$ provides the kth factor of p variables, $k = 1, 2, \ldots, q$.

5.1.3 Duality Relations Between $\mathcal{X}^T\mathcal{X}$ and $\mathcal{X}\mathcal{X}^T$

The data $\mathcal{X}$ is an $n \times p$ matrix. We set $r = \text{rank}(\mathcal{X}) = \text{rank}(\mathcal{X}\mathcal{X}^T) = \text{rank}(\mathcal{X}^T\mathcal{X}) \leq \min(n, p)$. For $k < r$, we have the eigenvalue problem

$$\mathcal{X}^T\mathcal{X}v_k = \lambda_k v_k,$$

where v_k is an eigenvector of $\mathcal{X}^T\mathcal{X}$ corresponding to the eigenvalue λ_k. Multiplying by $\mathcal{X}$ on both side gives

$$\mathcal{X}\mathcal{X}^T(\mathcal{X}v_k) = \lambda_k(\mathcal{X}v_k).$$

Now $\mathcal{X}v_k$ is an eigenvector of $\mathcal{X}\mathcal{X}^T$ corresponding to the eigenvalue λ_k.

Similarly, for the eigenvalue problem of $\mathcal{X}\mathcal{X}^T$

$$\mathcal{X}\mathcal{X}^T u_k = \lambda_k u_k,$$

and by multiplying by $\mathcal{X}^T$ on both side,

$$\mathcal{X}^T\mathcal{X}(\mathcal{X}^T u_k) = \lambda_k(\mathcal{X}^T u_k).$$

Now $\mathcal{X}^T u_k$ is an eigenvector of $\mathcal{X}^T\mathcal{X}$ corresponding to the eigenvalue λ_k.

Hence we obtain the relationship

$$v_k = c_{k1}\mathcal{X}^T u_k \quad \text{and} \quad u_k = c_{k2}\mathcal{X}v_k,$$

where c_{k1} and c_{k2} are the constants. Since $\|v_k\| = \|u_k\| = 1$, then

$$c_{k1} = c_{k2} = \frac{1}{\sqrt{\lambda_k}}.$$

Theorem 5.1.4 (Duality Relations [17]). *For $k < r = rank(\mathcal{X})$, the matrices $\mathcal{X}^T\mathcal{X}$ and $\mathcal{X}\mathcal{X}^T$ share the same nonzero eigenvalues λ_k, and the corresponding eigenvectors v_k and u_k have the relationship*

$$v_k = \frac{1}{\sqrt{\lambda_k}}\mathcal{X}^T u_k \tag{5.7}$$

$$u_k = \frac{1}{\sqrt{\lambda_k}}\mathcal{X} v_k. \tag{5.8}$$

To analyze n observations of $\mathcal{X}$ using the factorial method, we need to obtain the kth factor that is given by $\mathcal{X}v_k$. From equation (5.8), we have the kth factor

$$\mathcal{X}v_k = \sqrt{\lambda_k}u_k. \tag{5.9}$$

Result 5.1.1. *Equation* (5.9) *means that the eigenvector u_k of the $n \times n$ matrix $\mathcal{X}\mathcal{X}^T$ can be used directly as the kth factor to analyze n observations. For instance, the first eigenvector u_1 of $\mathcal{X}\mathcal{X}^T$ can be treated as the first factor of n observations.*

To analyze p variables of $\mathcal{X}$ using the factorial method, we need to obtain the kth factor that is given by $\mathcal{X}^T u_k$. From equation (5.7), we have the kth factor

$$\mathcal{X}^T u_k = \sqrt{\lambda_k}v_k. \tag{5.10}$$

Result 5.1.2. *Equation* (5.10) *means that the eigenvector v_k of $p \times p$ matrix $\mathcal{X}^T\mathcal{X}$ can also be used directly as the kth factor to analyze p variables. For instance, the first eigenvector v_1 of $\mathcal{X}^T\mathcal{X}$ can be treated as the first factor of p variables.*

Note that v_k and u_k are also the singular vectors given by the SVD of $\mathcal{X}$. Therefore the SVD can be applied on the data set $n \times p$ matrix $\mathcal{X}$ to obtain the factors of n observations and p variables.

5.1.4 Evaluating a Fitting

In the factorial method, we usually project n observations in $\mathcal{R}^p$ of $\mathcal{X}$ into a 2-dimensional plane by obtaining the first factor $z_1 = \mathcal{X}v_1$ and the second factor $z_2 = \mathcal{X}v_2$, where v_1 and v_2 are the eigenvectors of $\mathcal{X}^T\mathcal{X}$ corresponding to the largest and the second largest eigenvalues λ_1 and λ_2. Then the representation of n observations on the plane can be plotted by z_1 versus z_2. We can cluster the n observations in two dimensions.

It is natural to ask if the representation of n observations in a plane captures the major information of data, and how should it be evaluated? Suppose we use a q-dimensional subspace ($q < p$) to fit n observations, we have to find a way to evaluate the quality of fitting in the q-dimensional subspace. It is obvious that the closer q is to p, the more accurate the fitting.

Let us first consider the case of $p = 3$ and $q = 2$, i.e., using a plane to fit n points in $\mathcal{R}^3$. By Theorem 5.1.2, the best plane P_1 in the least-squares sense is given by the eigenvectors v_1 and v_2 of $\mathcal{X}^T\mathcal{X}$ corresponding to eigenvalues λ_1 and λ_2, where $\lambda_1 \geq \lambda_2 \geq \lambda_3$ are the eigenvalues of $\mathcal{X}^T\mathcal{X}$. The projection point of an individual $x_i \in \mathcal{R}^3$ on P_1 is $p_{x_i} \in \mathcal{R}^3$. Then the least-squares function (5.1) is minimized. From the Pythagorean theorem and Equation (5.5), we have

$$\begin{aligned}
\sum_{i=1}^{n} \|x_i - p_{x_i}\|^2 &= \sum_{i=1}^{n} \|x_i\|^2 - \sum_{i=1}^{n} \|p_{x_i}\|^2 \\
&= trace(\mathcal{X}\mathcal{X}^T) - (v_1^T \mathcal{X}^T \mathcal{X} v_1 + v_2^T \mathcal{X}^T \mathcal{X} v_2) \\
&= trace(\mathcal{X}^T\mathcal{X}) - (\lambda_1 + \lambda_2) \\
&= \sum_{i=1}^{3} \lambda_i - (\lambda_1 + \lambda_2) \\
&= \lambda_3.
\end{aligned}$$

The smallest eigenvalue λ_3 of $\mathcal{X}^T\mathcal{X}$ provides the least-squares value of fitting n points on the plane P_1. In the general case, using a q-dimensional subspace to fit n points $x_i \in \mathcal{R}^p$, we have

$$\sum_{i=1}^{n} \|x_i - p_{x_i}\|^2 = \sum_{i=q+1}^{p} \lambda_i. \tag{5.11}$$

Therefore, a standard way to evaluating the fitting in a subspace of dimension q is to use the percentage ratio δ:

$$\delta = \frac{\lambda_1 + \cdots + \lambda_q}{\lambda_1 + \cdots + \lambda_p} = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i}. \tag{5.12}$$

The ratio δ with $0 < \delta < 1$ shows the percentage of data information explained by the first q factors.

5.2 Spectral Clustering and Factorial Method

In spectral clustering, the Laplacian matrix L and the signless Laplacian matrix M of a graph G are the basic tools to model the graph minimum cut and maximum association. We have

$$L = B_G {B_G}^T \quad \text{and} \quad M = I_G {I_G}^T,$$

where B_G is the oriented incidence matrix of G and I_G is the unoriented incidence matrix.

5.2.1 Commonalities

The matrix I_G can be seen as the data set matrix $\mathcal{X}$ in the factorial method. The oriented incidence matrix B_G can not be seen as $\mathcal{X}$, because the negative value in B_G corresponding to each edge is arbitrary so that it could not represent the intrinsic nature of the data. The I_G of G is an $n \times m$ matrix having one row per node and one column per edge. Following the idea of the factorial method, the I_G can be represented by n points (vertices) in $\mathcal{R}^m$. Thus we can analyze the factors of n vertices decomposed from I_G in the least-square sense.

As discussed in Section 5.1.1, the kth factor axis of vertices is the eigenvector v_k of ${I_G}^T I_G$ corresponding to the kth largest eigenvalue λ_k, and the kth factor is $I_G v_k$. From Equation (5.9), the kth factor is also equal to $\sqrt{\lambda_k} u_k$, where u_k is the eigenvector of $M = I_G {I_G}^T$ corresponding to the kth largest eigenvalue λ_k. Therefore the factors of n vertices in the factorial method can be obtained by the eigenvectors of the signless Laplacian M. For example, the first factor is the eigenvector of M associated with the largest eigenvalue. In spectral clustering, the second eigenvector u_2 of M associated with the second largest eigenvalue λ_2 provides the nodes clustering result and here u_2 is also associated with the second factor of n vertices. We do not use the first factor to cluster nodes, because all elements in it have the same sign, which does not comply with the definition of a partition vector in the context of graph partitioning. However the first factor contains important information concerning the n vertices. In the context of the factorial method, the first factor can also be used to cluster nodes if we cluster its elements based on close values, but not based on different signs, in which case we get the same clustering result as in spectral clustering. Remember the factors are the coordinates of the projection of n vertices in a lower dimensional subspace in the least-square sense. Since I_G is nonnegative, it

is appropriate that the projection coordinates of n vertices on the first factor axis (first factor) are positive. The projection coordinates on the second factor axis (second factor) are distributed in both sides of the origin, which are used to partition graph nodes. This is also another explanation as to why multiple eigenvectors of M can be used to obtain multiple clustering, because the first q eigenvectors of M provide the coordinates of n vertices projection in the best fitting of a q-dimensional subspace. Here we may say spectral clustering is one special case of the factorial method of data matrices, i.e., elements in the data set I_G are nonnegative with special structure. The factors of n vertices obtained from the incidence matrix I_G have a connection to graph partitioning.

If we consider that I_G is represented by m points (edges) in $\mathcal{R}^n$, then from Equation (5.10) the kth factor of m edges ${I_G}^T u_k$ is equal to $\sqrt{\lambda_k} v_k$, where v_k is the eigenvector of $E = {I_G}^T I_G$ corresponding to the kth largest eigenvalue. Therefore the eigenvectors of E treated as factors of m edges can be used to cluster graph edges. In spectral clustering, the eigenvectors of E provide edge clustering based on the vertices in the line graph of the original graph G. Therefore, the results from factorial method and spectral clustering for clustering edges coincide in the eigenvectors of $E = {I_G}^T I_G$.

Since the eigenvector u_k of $M = I_G {I_G}^T$ and the eigenvector v_k of $E = {I_G}^T I_G$ are the left-hand and right-hand singular vector of I_G, the SVD of I_G contains the clustering information of vertices and edges from the point of view of the factorial method.

Example 5.2.1. Let us return to Example 3.3.1, the graph with six vertices. The eigenvectors u_k and v_k of the matrices M and E are given in the orthogonal matrices U and V, respectively.

$$\mathbf{U} = \begin{pmatrix} 0.5597 & -0.1426 & 0.4930 & -0.5223 & 0.2012 & -0.3318 \\ 0.5752 & -0.1764 & 0.1750 & 0.5136 & -0.5654 & 0.1546 \\ 0.5429 & -0.1041 & -0.6850 & 0.0195 & 0.4393 & 0.1784 \\ 0.1677 & 0.5802 & -0.3852 & -0.3130 & -0.5115 & -0.3566 \\ 0.1015 & 0.4830 & 0.1696 & 0.5880 & 0.4137 & -0.4591 \\ 0.1509 & 0.6064 & 0.2828 & -0.1389 & 0.1182 & 0.7045 \end{pmatrix}$$

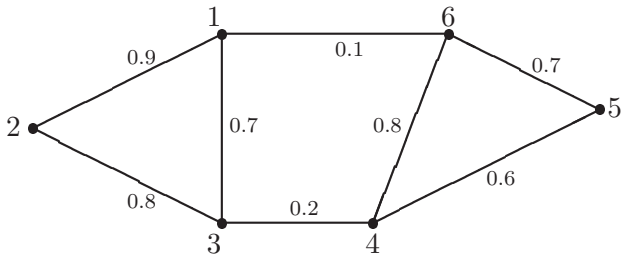

Figure 5.2: A sample graph with six vertices.

$$\mathbf{V} = \begin{pmatrix} 0.5899 & -0.1778 & 0.6031 & -0.0091 & -0.3968 & -0.2021 & 0.1198 & 0.2097 \\ 0.5479 & -0.1474 & -0.4341 & 0.5282 & -0.1295 & 0.3581 & -0.1271 & -0.2225 \\ 0.5055 & -0.1212 & -0.1528 & -0.4658 & 0.6155 & -0.1542 & -0.2853 & 0.0738 \\ 0.1231 & 0.0861 & 0.2335 & -0.2315 & 0.1160 & 0.1417 & 0.3952 & -0.8244 \\ 0.1741 & 0.1250 & -0.4555 & -0.1454 & -0.0371 & -0.0958 & 0.7879 & 0.3069 \\ 0.1561 & 0.6233 & -0.0872 & -0.4476 & -0.4040 & 0.3741 & -0.2668 & 0.0690 \\ 0.1143 & 0.4837 & -0.1590 & 0.2359 & -0.0871 & -0.7596 & -0.1468 & -0.2569 \\ 0.1157 & 0.5353 & 0.3602 & 0.4162 & 0.5111 & 0.2468 & 0.1359 & 0.2378 \end{pmatrix}$$

The vectors u_1 and v_1, i.e., the first factors of graph nodes and edges respectively, can also be analyzed to cluster nodes and edges; meanwhile, values of elements in u_2 and v_2, i.e., the second factors, are appropriate within the context of graph partitioning.

5.2.2 Differences

There are two main differences between spectral clustering and the factorial method:

1. In multivariate analysis, the data set is an $n \times p$ matrix $\mathcal{X}$. In most cases, the number of variables p is much smaller than the number of observations n, e.g., $p \sim 10$ and $n \sim 10^3$ or more. In the factorial method, for obtaining the first several factors of observations, we can compute the eigenvectors of the small $p \times p$ matrix $\mathcal{X}^T\mathcal{X}$ to get the factor axes. Then the factors of observations are linear combinations of the original p variables with coefficients given by these factor axes. In other words, because of the characteristics of multivariate data, the factorial method solves the eigenvalue problem of a small matrix, instead of a large matrix, and thereby avoids a large amount of computation.

However, in spectral clustering with an $n \times m$ incidence matrix I_G, the number of edges m is usually much larger than the number of nodes n. Therefore, in the case of a graph with a large number of nodes, we cannot find a small matrix to reduce the computational effort to cluster graph nodes.

2. In the factorial method, n observations in $\mathcal{R}^p$ are basically projected onto a subspace of dimension two or three. Since p is small, the ratio δ in Equation (5.12) is introduced to evaluate the fitting in a subspace of dimension $q < p$,

$$\delta = \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i},$$

where $\lambda_i, i = 1, \ldots, p$, are eigenvalues of the $p \times p$ matrix $\mathcal{X}^T\mathcal{X}$ in descending order. The ratio δ shows the percentage of data information explained by the first q factors.

However, in spectral clustering with large-scale data, this ratio δ cannot correctly evaluate a partition given by the first several eigenvectors of the signless Laplacian M. For example, in general, it is enough to use the first two eigenvectors of M to cluster n nodes, then the ratio is

$$\delta = \frac{\sum_{i=1}^{2} \lambda_i}{\sum_{i=1}^{n} \lambda_i},$$

where $\lambda_i, i = 1 \ldots n$, are eigenvalues of M is descending order. If n is quite large, then δ will be small. We cannot use this ratio to explain how many eigenvectors are needed for partitioning graph nodes.

5.3 Principal Components Analysis

The aim of principal components analysis is also to reduce the dimension of the multivariate data. It produces a lower dimensional representation of the observations or variables of a multivariate data matrix. Then we can analyze this lower dimensional representation to cluster the observations or variables. A multivariate variable X with p variables is a vector

$$X = (X_1, X_2, \ldots, X_p)^T.$$

To reduce the dimension of each observation, we take a weighting coefficients vector $\mu = (\mu_1, \mu_2, \ldots, \mu_p)^T$ to obtain a projection of X, so that

$$\mu^T X = \sum_{i=1}^{p} \mu_i X_i \quad \text{and} \quad \sum_{i=1}^{p} {\mu_i}^2 = 1.$$

The weighting coefficients μ we choose are used to investigate and detect specific features of the multivariate variable X. One strategy is to maximize the variance of the projection $\mu^T X$. Since

$$\mathrm{Var}(\mu^T X) = \mu^T \mathrm{Var}(X)\mu,$$

the problem is formulated so as to find a coefficient vector μ such that

$$\max_{\|\mu\|=1} \mu^T \mathrm{Var}(X)\mu. \tag{5.13}$$

Here $\mathrm{Var}(X)$ is the covariance matrix of multivariate variable X where definition of the covariance matrix Σ is

$$\Sigma = \begin{pmatrix} \sigma_{X_1}^2 & \sigma_{X_1X_2} & \dots & \sigma_{X_1X_p} \\ \sigma_{X_2X_1} & \sigma_{X_2}^2 & \dots & \sigma_{X_2X_p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{X_pX_1} & \sigma_{X_pX_2} & \dots & \sigma_{X_p}^2 \end{pmatrix}, \tag{5.14}$$

with

$$\sigma_{X_iX_j} = \mathrm{Cov}(X_i, X_j) = E(X_iX_j) - E(X_i)E(X_j) \qquad i \neq j,$$

and

$$\sigma_{X_i}^2 = \mathrm{Cov}(X_i, X_i) = \mathrm{Var}(X_i).$$

$\mathrm{Cov}(X_i, X_j)$ is the covariance of two random variables X_i and X_j.

Theorem 5.3.1. *From Theorem 2.2.5, the solution vector μ of problem* (5.13) *is given by the eigenvector v_1 corresponding to the largest eigenvalue λ_1 of covariance matrix Σ.*

The vector v_1 gives the direction on which the variance of the projection of observations is maximized, or alternatively this direction captures the majority of the variance of the data. Therefore the linear combination

$$Y_1 = {v_1}^T X = v_{11}X_1 + v_{21}X_2 + v_{31}X_3 + \cdots + v_{p1}X_p,$$

is called the first principal component [17]. The second highest variance of the projection is given by $Y_2 = {v_2}^T X$, called the second principal component, where vector v_2 is orthogonal to v_1 and is the eigenvector corresponding to the second largest eigenvalue λ_2 of Σ. In matrix notation, principal components are $Y = V^T X$, where V is the matrix form of eigenvectors of Σ. These first several principal components are what we want for describing the multivariate data in a lower dimension.

In practice, given the multivariate statistical data $\mathcal{X}$ ($n \times p$ matrix) with n observations and p variables, let x_{ij} be the entry of $\mathcal{X}$ corresponding to the ith row and jth column, then the elements in the covariance matrix of $\mathcal{X}$ in Equation (5.14) are

$$\sigma_{X_i X_j} = \frac{1}{n}\sum_{k=1}^{n}(x_{ki} - \overline{x_i})(x_{kj} - \overline{x_j}),$$

and

$$\sigma_{X_i}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_{ki} - \overline{x_i})^2,$$

where $\overline{x_i} = \frac{1}{n}\sum_{k=1}^{n} x_{ki}$.

Here we can see that it is better to first center every variable of the data matrix $\mathcal{X}$. We denote the centered data matrix as $\mathcal{X}_c$. It is given by

$$\mathcal{X}_c = \mathcal{X} - 1_n \overline{x}^T,$$

where $1_n = (1, 1, \ldots, 1)^T$ in n dimension and $\overline{x} = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_p})^T$. If we let $\mathcal{H} = I - (n^{-1} 1_n {1_n}^T)$ be the centering matrix, then $\mathcal{X}_c = \mathcal{H}\mathcal{X}$. After forming the centered data matrix $\mathcal{X}_c$, it is not difficult to calculate the covariance matrix

$$\Sigma = \frac{1}{n}{\mathcal{X}_c}^T \mathcal{X}_c. \tag{5.15}$$

Theorem 5.3.2 ([17]). *Let $v_1, \ldots, v_p$ be the eigenvectors of the $p \times p$ covariance matrix Σ corresponding to the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_p$, then the principal components are given by $y_k = \mathcal{X}_c v_k$ with the following*

$$E(y_k) = 0, \quad k = 1, \ldots, p \tag{5.16}$$

$$\mathrm{Var}(y_k) = \lambda_k, \quad k = 1, \ldots, p \tag{5.17}$$

$$\sum_{k=1}^{p} \mathrm{Var}(y_k) = \sum_{k=1}^{p} \lambda_k = \mathrm{trace}(\Sigma). \tag{5.18}$$

The principal component $y_k = \mathcal{X}_c v_k$ can be represented by an eigenvector of the $n \times n$ matrix $\frac{1}{n}\mathcal{X}_c{\mathcal{X}_c}^T$. Recall our discussion of the duality relations between $\frac{1}{n}{\mathcal{X}_c}^T\mathcal{X}_c$ and $\frac{1}{n}\mathcal{X}_c{\mathcal{X}_c}^T$ in Section 5.1.3, if we let u_k be the kth eigenvector of $\frac{1}{n}\mathcal{X}_c{\mathcal{X}_c}^T$ associated with the kth largest eigenvalue λ_k, then from Equation (5.9), we have

$$\frac{1}{\sqrt{n}}\mathcal{X}_c v_k = \sqrt{\lambda_k} u_k.$$

This is equivalent to

$$y_k = \mathcal{X}_c v_k = \sqrt{n\lambda_k} u_k. \tag{5.19}$$

Therefore, the kth principal component y_k can be obtained by the kth eigenvector of $\frac{1}{n}\mathcal{X}_c{\mathcal{X}_c}^T$. The most important one is the first principal component $y_1 = \sqrt{n\lambda_1}u_1$, where u_1 is the first eigenvector of $\frac{1}{n}\mathcal{X}_c{\mathcal{X}_c}^T$. Actually the vector u_1 also provides a partition solution of a graph composed by n observations of the multivariate variable X. We shall explain this partition solution in next section.

5.4 Graph Partitioning in Multivariate Statistical Analysis

In this section, we shall consider the possibility of using graph partitioning techniques to analyze a multivariate statistical variable X.

Our task is to analyze n observations of X in a lower dimensional representation. If we want to apply graph partitioning techniques, the first problem we need to solve is to create a graph of n nodes associated with n observations. This is equivalent to the problem of deciding which method to use to determine the weight of an edge connecting

two observations. It is necessary to choose an edge weight that appropriately depicts the relationship between two observations.

Given a centered data matrix $\mathcal{X}_c$ with n observations and p variables, from Equation (5.19), we know the kth principal component is $y_k = \sqrt{n\lambda_k}u_k$, where u_k is the eigenvector of $\frac{1}{n}\mathcal{X}_c{\mathcal{X}_c}^T$ corresponding to the kth largest eigenvalue λ_k. Now let us focus on the $n \times n$ matrix $\mathcal{X}_c{\mathcal{X}_c}^T$. If we set $W = \mathcal{X}_c{\mathcal{X}_c}^T$, and let a vector ${x_i}^T = (x_{i1}, x_{i2}, \ldots, x_{ip})$ be the ith row of $\mathcal{X}_c$ corresponding to the ith observation, then the element w_{ij} of W is

$$w_{ij} = {x_i}^T x_j = x_{i1}x_{j1} + x_{i2}x_{j2} + \cdots + x_{ip}x_{jp}. \tag{5.20}$$

If $w_{ij} < 0$, then the kth element $x_{ik}x_{jk}$ of Equation (5.20) is more likely to be negative or we can say w_{ij} is dominated by negative $x_{ik}x_{jk}$. This means, for the kth variable of observations i and j, one value is above the mean of this variable and another is below the mean. If $w_{ij} > 0$, then we can say w_{ij} is dominated by positive $x_{ik}x_{jk}$. This means, for the kth variable of observations i and j, both values are above the mean or both are below the mean. Therefore a negative value of w_{ij} depicts the difference between observations i and j, and a positive value depicts the similarity. The value w_{ij} can be assigned to the edge connecting i and j as a weight.

Definition 5.4.1. *Given a multivariate centered data matrix $\mathcal{X}_c$ with n observations and p variables, an edge connecting observations i and j ($i \neq j$) in a graph G of n nodes (observations) without self-loops has a weight w_{ij} such that*

$$w_{ij} = {x_i}^T x_j$$

where ${x_i}^T = (x_{i1}, x_{i2}, \ldots, x_{ip})$ is the ith row of $\mathcal{X}_c$ associated with the observation i.

Based on this graph G, two observations connected by a large positive edge should be in the same cluster, and two observations connected by a negative edge having large modulus should be in different clusters. Therefore, we can use the spectral clustering technique discussed in Section 3.4 to cut negative edges so as to partition graph nodes.

To perform the spectral clustering, we shall first get the adjacency matrix A of G whose elements are edge weights. The diagonal elements of A are zeros. Then an $n \times n$ matrix M is obtained as $M = D + A$, where D is a diagonal matrix whose diagonal element is $d_{ii} = \sum_k |w_{ik}|$. A partition result with more positive edges within clusters and more

negative edges cut between clusters can be achieved by maximizing the Rayleigh quotient of $p^T Mp/p^T p$, where p is a partition vector defined in Equation (3.5). Since the matrix $W = \mathcal{X}_c \mathcal{X}_c{}^T$, and the only differences between W and the matrix $M = D + A$ of G are diagonal elements, then we can get a diagonal matrix $D_1 = W - M$. Now

$$\frac{p^T Mp}{p^T p} \equiv \frac{p^T Wp}{p^T p} - \frac{p^T D_1 p}{p^T p}.$$

It is obvious that $p^T D_1 p/p^T p$ is constant, so maximizing $p^T Mp/p^T p$ is equivalent to maximizing $p^T Wp/p^T p$. From Theorem 2.2.5, an eigenvector of $W = \mathcal{X}_c \mathcal{X}_c{}^T$ corresponding to the largest eigenvalue maximizes the Rayleigh quotient of W. Therefore we can assert that the first eigenvector of $\mathcal{X}_c \mathcal{X}_c{}^T$ associated with the largest eigenvalue provides a heuristic solution for clustering graph nodes (observations) of G.

Because matrices $\mathcal{X}_c \mathcal{X}_c{}^T$ and $\frac{1}{n}\mathcal{X}_c \mathcal{X}_c{}^T$ share the same eigenvectors, this heuristic solution can be obtained from the first eigenvector u_1 of $\frac{1}{n}\mathcal{X}_c \mathcal{X}_c{}^T$. We already know the first principal component of n observations is $y_1 = \sqrt{n\lambda_1}u_1$, where λ_1 is the largest eigenvalue of $\frac{1}{n}\mathcal{X}_c \mathcal{X}_c{}^T$. In other words, the result from principal components analysis has been connected to the heuristic result from spectral clustering by a constant $\sqrt{n\lambda_1}$. Let us see an example of world university rank data.

Example 5.4.1. The Academic Ranking of World Universities data consists of information of the world's top 100 universities in 2005 (see appendix for data). All universities have been evaluated in six aspects: 'Score on Alumni', 'Score on Award', 'Score on Citation', 'Score on N&S', 'Score on SCI' and 'Score on Size'. These scores are given in a scale of 100. The multivariate data is a 100×6 matrix with 100 observations and 6 variables. This matrix should be first centered according to each variable.

Using the method of principal components analysis discussed in Section 5.3 to calculate the first principal component y_1, and then applying the K-means clustering algorithm on y_1, we obtain two groups of universities. The top 12 universities form one group, and the rest form the other group.

Using Definition 5.4.1 to create a graph with 100 nodes associated with 100 universities, and then computing the first eigenvector u_1 of W provides a heuristic solution for clustering nodes. The K-means clustering algorithm on u_1 gives two groups of universities. This result is the same as that obtained by the principal components analysis.

In this chapter, we discussed the relation between spectral clustering and statistical clustering, and especially two techniques in statistical clustering, i.e., the factorial method and principal components analysis. We saw that

1. Spectral clustering can be seen as a special case of the factorial method on node-edge incidence matrix.

2. Based on a graph of observations created from a multivariate data matrix, the result obtained from spectral clustering is related to the result obtained from principal components analysis.

Chapter 6

Heuristic Methods for Decomposing Real Unsymmetric Matrices

6.1 Introduction

Classical block iterative methods for solving large sparse linear systems of equations in the form

$$Ax = b,$$

where A is an $n \times n$ real matrix and x, b are vectors, requires that the diagonal blocks be more dense than the off-diagonal blocks. This is also very common in the problems of computing the stationary distribution of a large Markov chain. A Markov chain is called nearly completely decomposable (NCD), if its state space can be partitioned into disjoint subsets, with strong interaction between states within subsets but with weak interaction between subsets themselves. Given an NCD Markov chain, we want to obtain a block form as

$$\mathbf{P_{n\times n}} = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{pmatrix}$$

by a symmetric permutation so that the elements in the off-diagonal blocks P_{ij} $(i \neq j)$ are much smaller than those in the diagonal blocks P_{ii}.

In this chapter, we discuss applying spectral clustering procedures to decomposing a real unsymmetric matrix into blocks and especially to find the nearly-completely-decomposable components of NCD Markov chains. There are already several effective methods in this field, such as MARCA [35] and TPABLO [3], which will be introduced first in Section 6.2 and Section 6.3 respectively. In Section 6.4, we shall develop a new spectral clustering procedure for obtaining partitions of matrices. A modified clustering procedure suitable for the case of large-scale data matrices is also presented.

6.2 MARCA

MARkov Chain Analyzer (MARCA) [36, 37] is a software package designed for generating and analyzing large Markov chains. In this section, we focus on the near-complete decomposability test (*ncdtest*) of the package MARCA, i.e., a block partitioning algorithm for matrices. We shall call this algorithm MARCA for short.

To do the near-complete-decomposability test, MARCA provides a parameter for controlling the blocks solution obtained. This parameter γ is called the decomposability factor, which can be varied from 10^{-10} to 10^{-1}. For each different value, all nonzero elements of a probability matrix P less than this specified value are discarded. Then strongly connected components (SCC) are searched in a directed graph corresponding to the remaining matrix. Each strongly connected component represents a partition block. Finally, for different values of γ, MARCA lists the number of partition blocks found and the sizes of the biggest and smallest block obtained. If given an infinitesimal generator Q, MARCA will test the matrix $Q\Delta t + I$ instead. We shall go through an example to illustrate this procedure of MARCA.

Example 6.2.1. An artificially constructed 10×10 irreducible stochastic matrix P [35] is

$$\mathbf{P} = \begin{pmatrix}
. & .5 & . & .5 & . & . & & & & \\
. & . & .999994 & . & . & . & & & & \\
.999995 & . & . & . & . & . & & & & \\
. & . & . & .11999 & .44 & .44 & & & & \\
.0001 & . & . & .99 & .0099 & . & & & & \\
. & .0002 & .0002 & .99 & .00959 & . & & & & \\
. & .000001 & . & . & . & . & & & & \\
. & .000003 & . & . & . & .000007 & & & & \\
. & . & . & . & .000001 & . & & & & \\
. & . & .000005 & .000005 & . & . & & & &
\end{pmatrix} \cdots$$

$$\begin{pmatrix}
. & . & . & . \\
.000006 & . & . & . \\
. & . & . & .000005 \\
.000003 & .000002 & .000003 & .000002 \\
. & . & . & . \\
. & .000005 & .000005 & . \\
.499 & .499 & .001 & .000999 \\
.49999 & .49 & .006 & .004 \\
. & .009999 & . & .99 \\
.00999 & .01 & .98 & .
\end{pmatrix}.$$

In MARCA, if the parameter γ increases from 10^{-10} to 10^{-6}, there is no change on the matrix P, because none of the nonzero elements of P is less than 10^{-6}. This irreducible matrix P represents only one strongly connected component that contains all 10 states.

If γ is set to be 10^{-5}, then we obtain a modified matrix below:

$$\left(\begin{array}{cccccc|cccc}
. & .5 & . & .5 & . & . & . & . & . & . \\
. & . & .999994 & . & . & . & . & . & . & . \\
.999995 & . & . & . & . & . & . & . & . & . \\
. & . & . & .11999 & .44 & .44 & . & . & . & . \\
.0001 & . & . & .99 & .0099 & . & . & . & . & . \\
. & .0002 & .0002 & .99 & .00959 & . & . & . & . & . \\
\hline
. & . & . & . & . & . & .499 & .499 & .001 & .000999 \\
. & . & . & . & . & . & .49999 & .49 & .006 & .004 \\
. & . & . & . & . & . & . & .009999 & . & .99 \\
. & . & . & . & . & . & .00999 & .01 & .98 & .
\end{array}\right).$$

There are two SCCs corresponding to two blocks: the first consists of states 1 through 6 and the second consists of states 7 through 10. If γ increases to 10^{-4}, no change is made on this matrix.

Now if γ is continuously increased to 10^{-3}, then the resulting matrix has the following structure:

$$\left(\begin{array}{ccc|ccc} . & .5 & . & .5 & . & . \\ . & . & .999994 & . & . & . \\ .999995 & . & . & . & . & . \\ \hline . & . & . & .11999 & .44 & .44 \\ . & . & . & .99 & .0099 & . \\ . & . & . & .99 & .00959 & . \end{array}\right) \text{ and } \left(\begin{array}{cccc} .499 & .499 & . & . \\ .49999 & .49 & .006 & .004 \\ . & .009999 & . & .99 \\ .00999 & .01 & .98 & . \end{array}\right).$$

The second block consisting of states 7 through 10 still has only one SCC contains all four states. However, in the first block of six states, we can find two SCCs. Therefore this block is divided into two components: states 1 through 3 and states 4 through 6.

If γ is 10^{-2}, then we obtain the resulting matrix as

$$\left(\begin{array}{ccc|ccc} . & .5 & . & .5 & . & . \\ . & . & .999994 & . & . & . \\ .999995 & . & . & . & . & . \\ \hline . & . & . & .11999 & .44 & .44 \\ . & . & . & .99 & . & . \\ . & . & . & .99 & . & . \end{array}\right) \text{ and } \left(\begin{array}{cc|cc} .499 & .499 & . & . \\ .49999 & .49 & . & . \\ \hline . & . & . & .99 \\ . & . & .98 & . \end{array}\right).$$

Based on the search of SCCs, the block consisting of last four states is broken into two components. Now there are four groups of states obtained: {1,2,3},{4,5,6},{7,8} and {9,10}. This partition result will not change if γ is increased to 10^{-1}.

In next section, we shall review another algorithm, called TPABLO, for finding blocks of NCD Markov chains.

6.3 TPABLO Algorithms

The TPABLO (Threshold PArameterized BLock Ordering) partitioning algorithms [3] are variants of the PABLO algorithm. The original PABLO algorithm presented by O'Neil and Szyld [29] aims to obtain a diagonal dense blocks by permuting sparse matrices.

The new permuted matrices can be used to improve block iterative methods for solving linear systems. We first quickly review the PABLO algorithm, and then describe the TPABLO algorithms.

6.3.1 The Original PABLO Algorithm

Given a matrix A, let $G = (V, E)$ be its associated graph, i.e., G is a matrix-induced graph of the matrix A, then PABLO will produce k subgraphs $G_i = (V_i, E_i)$, $i = 1, \ldots, k$, corresponding to k permuted diagonal blocks of the matrix A, where $E_i \subset E$, and the sets V_i are disjoint and $\cup_{i=1}^{k} V_i = V$. The number of blocks is not given a priori, but it is determined by the algorithm with input parameters. The graph G is supposed to be connected. There are three sets named P, Q and C in PABLO. The set P contains those marked vertices that will be organized into the current block, and finally becomes a V_i. The set Q contains those vertices that are adjacent to at least one vertex in P. The set C contains the rest of the unmarked vertices of V that are neither in P nor in Q. The PABLO algorithm is a series of operations on these three sets until no further operation can be applied. At the beginning, we set $P = Q = \emptyset$ and $C = V$.

Let us define the fullness of a graph $G(V, E)$, denoted as ϕ_V, as the ratio of the number of edges in E to the number of edges in the clique of the vertex set V. For the vertex i, let $deg(i)$ be the number of edges in the form $\{i, j\}$ or $\{j, i\}$, for a different vertex j; and let $in(i)$ be the number of edges $\{i, j\}$ or $\{j, i\}$ with $j \in P$. PABLO will choose the first vertex from C which has the minimum *degree*, mark it, and put it into set P. Here we initiate $\phi_P = 0$. All vertices in C that are adjacent to the first vertex chosen into P are moved to the rear of the set Q. After this operation, PABLO updates the in degree of all vertices in Q. Now we pick up a node p from the front of Q, and we have to decide whether or not p should be added to P. PABLO uses two criteria with two input parameters α and β to make the decision.

1. To ensure that the fullness of $G(P \cup \{p\})$ keeps at least some fraction of the fullness of $G(P)$:

$$\phi_{P \cup \{p\}} \geq \alpha \phi_P, \quad \text{for} \quad \alpha \geq 0.$$

2. To ensure that the node p is adjacent to more nodes in P:

$$\frac{in(p)}{deg(p)} \geq \beta, \quad \text{for} \;\; 0 \leq \beta \leq 1.$$

The node p will be added to P if either of these two criteria is satisfied. If neither of these two criteria is satisfied, then the node p will be put back into C. After a new node is added to P, we need to update the Q and repeat previous operations until Q is empty. Thus we obtain a subgraph $G(V_i, E_i)$ with $V_i = P$. If C is also empty, then all the vertices in V have been marked and the blocks have been obtained and the PABLO algorithm stops.

6.3.2 TPABLO1 and TPABLO2

There are two variants of the PABLO algorithm, called TPABLO1 and TPABLO2. In these two versions of TPABLO, a third additional criteria is introduced to decide if the vertex p from Q is added to the set P that contains those marked vertices. Let a_{ij} be an entry of A corresponding to ith row (vertex) and jth column (vertex), and $\gamma \geq 0$ be the given threshold which is introduced as the third parameter of TPABLO. Then the node p will be added to P if, besides either of these two criteria of PABLO mentioned above, the third criteria must be satisfied.

1. TPABLO1 algorithm:

$$|a_{ip}| > \gamma \;\; \text{or} \;\; |a_{pi}| > \gamma \;\; \text{for at least one} \;\; i \in P;$$

2. TPABLO2 algorithm:

$$|a_{ip}| > \gamma \;\; \text{and} \;\; |a_{pi}| > \gamma \;\; \text{for all} \;\; i \in P.$$

Here we can see that the third criteria of TPABLO1 makes the absolute value of every entry in off-diagonal blocks smaller than the given threshold γ, meanwhile the criteria of TPABLO2 makes the absolute value of every entry in diagonal blocks larger than the threshold γ. The TPABLO1 algorithm can be used to find blocks in nearly completely decomposable (NCD) Markov chains [3].

There are two other parameters for these two versions of TPABLO, called *minbs* and *maxbs*, which are used to control the minimum and maximum size of diagonal blocks obtained from TPABLO respectively.

6.3.3 Implementation

The PABLO, TPABLO1 and TPABLO2 algorithms were implemented in the C programming language. Running results of the code provide the number of blocks obtained, the size of each block and also list all states in each block. Since TPABLO1 is applied to find blocks in NCD Markov chains, it was tested with the 10×10 irreducible stochastic matrix P of Example 6.2.1.

In the TPABLO1 algorithm, if we set the parameters $\alpha = 0.6$, $\beta = 0.6$ and $\gamma = 0.001$, then it will give a partition of P with two blocks: states 1 through 6 and states 7 through 10. Now if we increase γ from 0.001 to 0.01, then a partition with three blocks is obtained: states 1 through 6, states 7 and 8, and states 9 and 10. Finally, if the parameters are changed to $\alpha = 0.8$, $\beta = 0.6$ and $\gamma = 0.01$, then the TPABLO1 will give a partition with four blocks: states {1,2,3},{4,5,6},{7,8} and {9,10}. Here we can see TPABLO1 algorithm can provide the desired blocks of NCD Markov chains.

6.4 Spectral Clustering Procedures for Finding Blocks of Matrices

The objective of spectral clustering is to group graph nodes based on a partitioning. One popular partitioning objective is normalized cut (balanced minimum cut), which can be related to the state clustering of Markov chains in the framework of a random walk generated on a graph. This partitioning objective can be used to find blocks of a probability matrix arising in the context of a Markov random walk on a graph. Each block corresponds to a cluster of graph nodes. Extending this idea, we shall apply spectral clustering techniques to general stochastic matrices in order to partition them. Of special concern is the application of finding blocks of NCD Markov chains. In this section, such spectral clustering procedures are introduced.

To decompose matrices into blocks, we seek to obtain multiple clusters of graph nodes using spectral clustering, and just bi-partitioning, i.e. a pair of clusters of graph nodes. One solution is to use multiple eigenvectors to group nodes into multiple subsets without losing information of whole graph structure. We can do this because of Courant-Fischer theorem, which will be explained in next section. Based on the Rayleigh quotient

and Courant-Fischer theorem, we propose a heuristic method for determining the number of clusters in a graph.

6.4.1 The Rayleigh Quotient and Courant-Fischer Theorem

Given a symmetric matrix A and a nonzero real vector x, the Rayleigh quotient $R(A, x)$ is defined as:

$$R(A, x) = \frac{x^T Ax}{x^T x}. \tag{6.1}$$

A well known result from matrix computation [16] states that

$$\lambda_{max} = \max_{x \neq 0} R(A, x) \quad \text{and} \quad \lambda_{min} = \min_{x \neq 0} R(A, x), \tag{6.2}$$

where λ_{max} and λ_{min} are the largest and smallest eigenvalue of A respectively, and x is the corresponding eigenvector.

It is only natural to wonder if the intermediate eigenvalues of a symmetric matrix have representations similar to those of the extreme eigenvalues as described by Equation (6.2). Fischer answered this question for matrices in 1905 while Courant [26] provided extensions for infinite-dimensional operators in 1920. Their results are combined into the so-called Courant-Fischer theorem which states:

Let V^k denote a k dimensional subspace of R^n and $x \perp V^k$ mean that $x \perp y$ for all $y \in V^k$. Then the eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ of a symmetric matrix $A_{n \times n}$ are

$$\lambda_i = \max_{V^{i-1}} \left(\min_{\substack{x \perp V^{i-1} \\ x \neq 0}} R(A, x) \right) \quad \text{and} \quad \lambda_i = \min_{V^{n-i}} \left(\max_{\substack{x \perp V^{n-i} \\ x \neq 0}} R(A, x) \right) \tag{6.3}$$

When $i = 1$ in the max-min formula and when $i = n$ in the min-max formula, these results revert to Equation (6.2).

From the proof of the Courant-Fischer theorem, it will follow that λ_i is attained when V^{i-1} in the first expression is the span of the first $i - 1$ eigenvectors of A, and V^{n-i} in the second expression is the span of the last $n - i$ eigenvectors of A. This means the ith eigenvector of A minimizes $R(A, x)$ in a subspace that is the span of the last $n - i + 1$ eigenvectors.

6.4.2 Determining the Number of Clusters

It is difficult to determine the number of clusters in a graph, because the criteria is obscure. How do we identify three clusters rather than two or four? Generally, the number of clusters is given as an input parameter in the previously discussed algorithms of spectral clustering. Here we propose a heuristic method based on the Courant-Fischer theorem to determine the number of clusters of a graph. The explanation is below:

From Theorem 3.7.1, the minimum value of a normalized cut can be obtained by minimizing the Rayleigh quotient of the normalized Laplacian matrix $\mathcal{L}$, and the eigenvector corresponding to the second smallest eigenvalue of $\mathcal{L}$ provides a way of grouping nodes into two clusters based on this normalized cut objective. This can also be observed from Equation (3.32). Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of the normalized Laplacian $\mathcal{L}$ and let $v_1, v_2, \ldots, v_n$ be their corresponding eigenvectors. Each eigenvector of $\mathcal{L}$ represents a partition solution of the graph. The v_1 minimizes the Rayleigh quotient $R(\mathcal{L}, x)$ and $R(\mathcal{L}, v_1) = \lambda_1 = 0$. However v_1 indicates that all nodes are in one cluster since all elements in v_1 have the same sign. It does not provide the normalized cut solution. The Courant-Fischer theorem tells us that the second eigenvector v_2 minimizes $R(\mathcal{L}, x)$ in the subspace which is the span of the last $n-1$ eigenvectors. It is for this reason that v_2 has been chosen to be the heuristic solution of the normalized cut. If we do not choose the partition solution provided by v_2, then we can pick up v_3 to partition the graph, because v_3 minimizes $R(\mathcal{L}, x)$ in the subspace which is the span of the last $n-2$ eigenvectors. Recall that the Rayleigh quotient $R(\mathcal{L}, v_i) = \lambda_i$ evaluates the cut value of a partition given by v_i. If $R(\mathcal{L}, v_3) = \lambda_3$ is close to 0 and not much different with λ_2, then the partition solution given by v_3 also makes sense and provides clustering information. This description is illustrated in the example of Figure 6.1. The left part of Figure 6.1 is a partition on a graph given by v_2 and the right

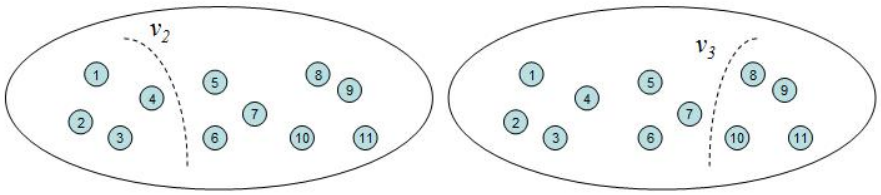

Figure 6.1: Using eigenvector v_2 and v_3 to partition a graph

part is a partition given by v_3. We can now obtain three clusters in this graph based on eigenvectors v_2 and v_3 together. This illustration gives us the intuition that the number of clusters in a graph can be determined by the number of eigenvectors with which we choose to partition the graph. If we choose m eigenvectors $v_2, v_3, \ldots, v_{m+1}$ to partition the graph, then the number of clusters obtained is $m + 1$.

Now this problem moves to the criterion of which eigenvectors should be chosen to partition the graph. Each eigenvalue λ_i of the normalized Laplacian $\mathcal{L}$ evaluates the cut value of a partition given by v_i, and the second smallest one λ_2 evaluates the value of the normalized cut. Therefore the eigenvalues of $\mathcal{L}$ can tell us which eigenvectors should be chosen. The details of how to decide upon the number of clusters is explained in our improved spectral clustering algorithm, which is in Section 6.4.4.

6.4.3 Spectral Clustering using Two Clustering Techniques on Markov Chains

As shown in Section 4.3, spectral clustering can be depicted in the framework of Markov random walks on the graph structure by $P = D^{-1}A$, where A is the adjacency matrix of a graph G and D is a diagonal matrix whose ith diagonal element is the ith vertex degree. P generated here is a transition probability matrix, and its eigenvalues are all real [22]. It is already known that the solution of the eigenvalue problem $Pv = \lambda v$ is also the solution of the generalized eigenvalue problem of the Laplacian $Lv = (1 - \lambda)Dv$ in Equation (3.31), which is used to minimize the normalized cut [24]. The eigenvalue λ of P is shifted to the eigenvalue $1 - \lambda$ of the normalized Laplacian $\mathcal{L}$. Therefore, the eigenvector corresponding to the second largest eigenvalue of P provides the heuristic solution for the normalized cut. Moreover, the stochastic matrix P and its eigenvectors are used in spectral clustering algorithms [25, 38].

The stochastic matrix P can represent a Markov chain, and there are two clustering techniques for Markov chains. The first technique focuses on the eigenvectors of P corresponding to the positive eigenvalues closest to 1, and the clustering information provided by these eigenvectors is based on the normalized cut. The second focuses on the eigenvectors of P corresponding to the eigenvalues with modulus closest to 1 (not necessarily positive), and the clustering information provided by these eigenvectors is based on a measure of the relative "distance" from each state to the stationary distribution. In previous

spectral clustering algorithms, only the first technique is used. In other words, only those eigenvectors with positive eigenvalues are used. Actually the eigenvector associated with a negative eigenvalue having the second largest modulus also provides important clustering information, i.e., cluster structure and the role or position of states (graph nodes) in each cluster, as discussed in Section 4.4. This is why here we introduce an improved spectral clustering algorithm that incorporates the two clustering techniques of Markov chains together. In Section 7.3, we shall see that our algorithm can be applied to detect communities in complex networks.

6.4.4 Multiple Clustering using Multiple Eigenvectors

The eigenvalues with large modulus of a stochastic matrix P show that the corresponding right eigenvectors, in which we are interested, contain significant information for clustering states. Since there are two clustering techniques available for a Markov chain, we analyze these right-hand eigenvectors from two aspects.

First, we pay attention to the eigenvectors associated with positive eigenvalues. Each eigenvector of P, likes each eigenvector of the normalized Laplacian $\mathcal{L}$, gives a partition solution of the graph G. The eigenvector having second largest eigenvalue provides the solution of the normalized cut. However, if the third or fourth largest eigenvalue of P is also close to 1, then the clustering information contained in these corresponding eigenvectors can not be ignored. Suppose the number of clusters existing in a graph is not given as a known, then, from the discussion in Section 6.4.2, this parameter may be determined by the number of eigenvectors with which we choose to partition the graph. Now we have to decide which eigenvectors are important.

An interesting result [9] is that a graph or a network is called well clustered if the P of a Markov random walk on this graph presents a spectral gap with a few nearly piece-wise constant eigenvectors with eigenvalues close to 1. A spectral gap means that if we sort the eigenvalues of P in descending order, there is an apparent value gap (value drop) between eigenvalues. This result may suggest a way of choosing important eigenvectors with which cluster nodes, based on such spectral gap of P. However, it is not easy to find out a spectral gap of P, especially when the state space is quite large and the eigenvalues of P are close and numerical-sensitive. An alternative way here is to examine a ratio of two adjacent eigenvalues of P, and then to decide the importance of the corresponding eigenvector for

clustering based on this ratio.

To decide which eigenvector is important, let $\lambda_1 \dots \lambda_m$ be the positive eigenvalues of P with the order $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$, and $v_1 \dots v_m$ be the corresponding right-hand eigenvectors, so $\lambda_1 = 1$ and $v_1 = e = [1, \dots, 1]^T$. We need to first check v_2, which provides the solution of normalized cut. If $\lambda_2 \approx 1$, then v_2 is significantly important; otherwise, v_2 is little important (the graph does not have well clustered structure). For the eigenvector v_i $(i = 3, \dots, m)$, if v_{i-1} is important for clustering and

$$\begin{cases} \lambda_i/\lambda_{i-1} \approx 1, & \text{then } v_i \text{ is significant;} \\ \\ \lambda_i/\lambda_{i-1} \approx 0, & \text{then } v_i \text{ is of little importance.} \end{cases}$$

Now let us introduce two input parameters α and β to control which eigenvectors should be chosen. Our algorithm begins by checking eigenvectors consecutively from v_2 to v_m. The first parameter α is used to check v_2. If $\lambda_2 \geq \alpha$, then v_2 is important; otherwise it is of little importance. The second parameter β is used to check $v_3 \dots v_m$. The eigenvector v_i $(i = 3 \dots m)$ is marked as important if $\lambda_i/\lambda_{i-1} \geq \beta$. The algorithm will not check any further eigenvectors once it discovers the first eigenvector v_j, for which $\lambda_j/\lambda_{j-1} < \beta$. Suppose v_2 is of little importance, then we do not need to check consecutive eigenvectors with positive eigenvalues because they are all of little importance. Finally, if the algorithm stops at v_j, then we obtain the important eigenvectors from v_2 to v_{j-1} for clustering, which imply that the number of clusters is $j - 1$.

We can see that the number of clusters obtained from our algorithm is controlled by the input parameters α and β, where $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$. The choice of α provides the user a control over how the graph is partitioned. It can be set as $\alpha = 0.5$, so if $\lambda_2 < 0.5$, then the graph contains only one cluster, itself. The value of β can vary to control whether multiple clusters greater than two are obtained or not. This β can also help us to find a spectral gap of P. Empirically, β between 0.7 and 0.9 can produce the desired clusters. One another choice is to set $\beta = \lambda_2$.

Second, we need to pay attention to a negative eigenvalue of P having the second largest modulus; if it exists, the corresponding eigenvector is also important and its elements give the relative "distance" from each state (graph node) to the steady state. Elements with small modulus in this eigenvector indicate that the corresponding states are close to the

steady state; in other words, these states are in a group in which states are more closely linked with each other. On the other hand, elements with large modulus indicate the corresponding states are far from the steady state and the states are not closely linked with each other. Also, element values that are close means that the corresponding states have more chance to be directly linked with each other than those whose values are not close. Therefore, the significant clustering information in this eigenvector should not be ignored.

In summary, we propose using two clustering techniques to group graph nodes. The eigenvectors with positive eigenvalues, which we choose by two input parameters α and β, can be used to determine the number of clusters in a graph, and the eigenvector with negative eigenvalue having second largest modulus indicates the cluster structure and the role or position of nodes in each cluster.

6.4.5 The Regular Clustering Algorithm

The spectral clustering algorithm is frequently used in the application of image segmentation. Later we shall see that it can also be applied to study the community structure in a complex network. However in these situation MARCA and TPABLO fail. We first present a regular spectral clustering algorithm that is suitable for the case of small-scale data, i.e., the number of graph nodes $n \approx 10^2$ or 10^3. Although the matrix P here is not large, we do not need to waste time and effort to calculate all n eigenvalues of P. The reason is that, in practice, the number of clusters k has $k \approx 10$ or 10^2, and only the first tens of eigenvalues and their corresponding eigenvectors are needed. Therefore, in the algorithm, we just calculate the first m ($m \approx n/10$) largest eigenvalues and their corresponding eigenvectors with which to do the clustering.

The procedure of our clustering algorithm consists of the following steps:

1. Given a graph G with n ($n \approx 10^2$ or 10^3) nodes, generate a Markov random walk on G as $P = D^{-1}A$, where A is the adjacency matrix of G and D is a diagonal matrix whose ith diagonal element is the ith vertex degree. P is a stochastic matrix.

2. Solve the eigenvalue problem $Pv = \lambda v$.

   If the number of clusters k is given as a known, then calculate $v_1, \ldots, v_k$, the right eigenvectors corresponding to the first k largest eigenvalues, and go to Step 4;

If k is not known a priori, then calculate $1 = \lambda_1 \geq \cdots \geq \lambda_m \geq 0$ the first m ($m \approx n/10$) largest eigenvalues and their corresponding right eigenvectors $v_1, \ldots, v_m$.

3. If $\lambda_2 < \alpha$, then v_2 is of little importance; there is no clustering and the algorithm stops; otherwise v_2 is marked as important.

   Next, compare v_3 to v_m. The algorithm marks v_i as important if $\lambda_i/\lambda_{i-1} \geq \beta$, until the first of little importance v_{k+1} is found; that is $\lambda_{k+1}/\lambda_k < \beta$. The important eigenvectors obtained are $v_2, \ldots, v_k$.

4. Form a matrix V whose column are $v_2, \ldots, v_k$ and the number of clusters is k.

5. Cluster the rows of V as points using the K-means algorithm with k clusters.

6. Calculate the eigenvalue λ_n of P.

   If λ_n is smallest but has the second largest modulus, then use the clustering result from Step 5 to partition v_n into k groups. Analyze the cluster structure and the role or position of nodes based on the elements in each group of v_n.

Let us first look at how this clustering algorithm works on some artificial sample graphs. The two parameters of our algorithm are set to be $\alpha = 0.8$ and $\beta = 0.85$.

Example 6.4.1. This is a graph with twelve vertices as shown in Figure 6.2. After formulating P on this graph, we obtain eigenvalues of P, $\lambda_2 = 0.9126$, $\lambda_3 = 0.8913$ and $\lambda_4 = 0.3919$. The eigenvector v_2 is important for clustering because $\lambda_2 > \alpha = 0.8$; v_3 is also important since $\lambda_3/\lambda_2 = 0.9767 > \beta = 0.85$. But v_4 is of little importance because $\lambda_4/\lambda_3 = 0.4397 < \beta$. Therefore we use v_2 and v_3 together to cluster vertices and the number of clusters $k = 3$. There is no negative eigenvalue having second largest modulus.

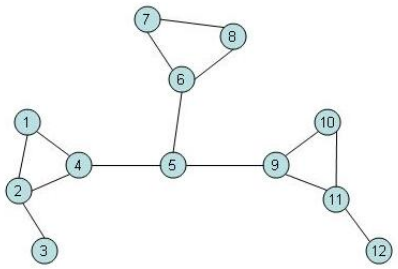

Figure 6.2: An sample graph with twelve vertices.

$$\mathbf{v_2}, \mathbf{v_3} = \begin{pmatrix} -0.3497 & 0.1966 \\ -0.3740 & 0.2144 \\ -0.4098 & 0.2405 \\ -0.2643 & 0.1361 \\ -0.0000 & -0.0471 \\ 0.0000 & -0.3981 \\ 0.0000 & -0.5086 \\ 0.0000 & -0.5086 \\ 0.2643 & 0.1361 \\ 0.3497 & 0.1966 \\ 0.3740 & 0.2144 \\ 0.4098 & 0.2405 \end{pmatrix}.$$

The K-means method gives clusters {1,2,3,4},{5,6,7,8} and {9,10,11,12}. Actually from v_2, v_3 and the graph, vertex 5 is more likely to be the center of this graph.

Example 6.4.2. This is a clique cycle graph with sixteen vertices in Figure 6.3. After formulating P on this graph, since the eigenvalues $\lambda_2 = 0.8965 > \alpha = 0.8, \lambda_3 = 0.8965, \lambda_4 = 0.7676, \lambda_5 = 0, \ldots,$ v_2 is important for clustering. The eigenvectors v_3 and v_4 are also important because $\lambda_3/\lambda_2 = 1 > \beta = 0.85$ and $\lambda_4/\lambda_3 = 0.8562 > \beta$. v_5 is of little importance obviously. Therefore we use v_2, v_3 and v_4 together to cluster vertices and the number of clusters $k = 4$. There is no such negative eigenvalue having the second largest modulus.

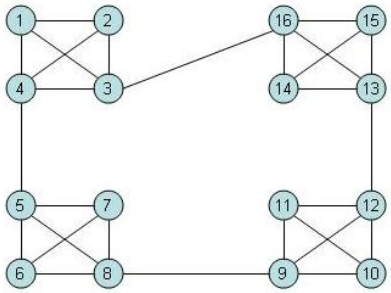

Figure 6.3: A clique cycle graph with sixteen vertices.

$$\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3} = \begin{pmatrix} 0.0807 & -0.1877 & -0.1631 \\ 0.0807 & -0.1877 & -0.1631 \\ 0.1028 & -0.1437 & -0.1063 \\ 0.0336 & -0.1734 & -0.1063 \\ -0.1437 & -0.1028 & 0.1063 \\ -0.1877 & -0.0807 & 0.1631 \\ -0.1877 & -0.0807 & 0.1631 \\ -0.1734 & -0.0336 & 0.1063 \\ -0.1028 & 0.1437 & -0.1063 \\ -0.0807 & 0.1877 & -0.1631 \\ -0.0807 & 0.1877 & -0.1631 \\ -0.0336 & 0.1734 & -0.1063 \\ 0.1437 & 0.1028 & 0.1063 \\ 0.1877 & 0.0807 & 0.1631 \\ 0.1877 & 0.0807 & 0.1631 \\ 0.1734 & 0.0336 & 0.1063 \end{pmatrix}.$$

The K-means method provides four cliques {1,2,3,4},{5,6,7,8},{9,10,11,12} and {13,14,15,16} as shown in Figure 6.3.

In the case of NCD Markov chains, there may be some real positive eigenvalues very close to 1. This fact can be associated with the first clustering technique using eigenvectors corresponding to these positive eigenvalues.

Example 6.4.3. Example 6.2.1 shows a 10×10 irreducible stochastic matrix P. Using our spectral clustering procedure, we calculate first several eigenvalues and their corresponding

eigenvectors of P. The eigenvalues in decreasing order are

$$\lambda = \begin{pmatrix} 1 & 0.999987 & 0.979026 & 0.793556 & 0.000154 & \cdots \end{pmatrix},$$

and their corresponding eigenvectors are the columns

$$\mathbf{v} = \begin{pmatrix} 0.3162 & -0.35187 & -0.000076 & 0.442391 & 0.000000 & \\ 0.3162 & -0.35187 & -0.000078 & 0.702501 & 0.006969 & \\ 0.3162 & -0.35187 & -0.000074 & 0.557477 & 0.000001 & \\ 0.3162 & -0.35186 & -0.000071 & -0.000376 & -0.006969 & \\ 0.3162 & -0.35186 & -0.000073 & -0.000419 & 0.708020 & \cdots \\ 0.3162 & -0.35186 & -0.000071 & -0.000157 & -0.706122 & \\ 0.3162 & 0.25354 & -0.265854 & -0.000007 & 0.000477 & \\ 0.3162 & 0.25353 & -0.258376 & -0.000005 & -0.000477 & \\ 0.3162 & 0.25354 & 0.659064 & -0.000007 & 0.000000 & \\ 0.3162 & 0.25353 & 0.654368 & -0.000006 & 0.000004 & \end{pmatrix}.$$

If we set two parameters $\alpha = 0.95$ and $\beta = 0.9$, then eigenvectors v_2 and v_3 are important for clustering because $\lambda_2 > \alpha = 0.95$ and $\lambda_3/\lambda_2 = 0.979 > \beta$, but v_4 is of little importance because $\lambda_4/\lambda_3 = 0.811 < \beta$. Hence, we apply the K-means method on v_2 and v_3 together to cluster states into three groups $\{1, \ldots, 6\}, \{7, 8\}$ and $\{9, 10\}$.

If β is decreased to 0.8 and α is unchanged, then v_4 now is important for clustering because $\lambda_4/\lambda_3 = 0.811 > \beta$, but v_5 is obvious of little importance since λ_5 almost equals to 0. The K-means method is applied on v_2, v_3 and v_4 to obtain four clusters $\{1, 2, 3\}, \{4, 5, 6\}, \{7, 8\}$ and $\{9, 10\}$.

We can see that if the eigenvector v_4 is used for clustering, then the group consisting of the first six states will be split into two clusters $\{1, 2, 3\}$ and $\{4, 5, 6\}$, however its corresponding eigenvalue λ_4 is not very close to 1. This is reasonable because there is a transition step from state 1 to state 4 with probability 0.5, which indicates this interaction is not weak.

Finally, we conclude that MARCA, TPABLO and the spectral clustering algorithm all work well on this 10×10 stochastic matrix to generate NCD blocks.

6.4.6 The Case of Large-Scale Data

In this section, we consider a spectral clustering algorithm suitable for the case of large-scale data. It is common that, in the case of NCD Markov chains, P is a large-scale,

sparse matrix, e.g., the number of states $n \approx 10^6$ or greater. It is also possible that the number of nodes in a graph in some applications is quite large. This implies that the number of clusters k can also be large. For spectral algorithms, it is not practical to calculate a large number of eigenvalues and their corresponding eigenvectors for a large-scale matrix P. However, it is easy to obtain the first several extreme eigenvalues of P. Therefore, we propose a new spectral hierarchical clustering algorithm for finding blocks (clusters) on a large-scale matrix (graph).

The idea is that we first partition a large-scale matrix P into several relative smaller submatrices, then for each submatrix, we apply the regular spectral clustering algorithm or other well known algorithms (MARCA and TPABLO) to obtain the final desired blocks. To implement this idea, we compute the first m ($m \approx 20$ to 40, $m \ll n$) real largest eigenvalues of P and their corresponding right eigenvectors. According to these m eigenvectors, the clustering method provides us with m relative smaller submatrices. After that, we continuous to partition each of these m submatrices by two approaches. One approach is to use TPABLO or MARCA for the application of finding NCD blocks. One other approach is to use the regular spectral clustering algorithm, which is suitable for more than the NCD case.

The procedure of the clustering algorithm for large-scale matrices consists of the following steps:

1. Given a graph G with n ($n \approx 10^6$) nodes, generate a probability matrix P by $P = D^{-1}A$; or given a large-scale NCD Markov chain with a probability matrix P.

2. Solve the eigenvalue problem $Pv = \lambda v$.

   Compute the first m ($m \approx 20$ to 40) real largest eigenvalues of P close to 1 and their corresponding right eigenvectors $v_1, \ldots, v_m$.

3. Form a matrix V whose column are $v_1, \ldots, v_m$.

4. Cluster the rows of V as points using the K-means algorithm with m clusters, then obtain m submatrices.

5. For each submatrix, there are two variants to do the postprocessing.

   Variant 1: Apply TPABLO or MARCA on each submatrix to find the NCD blocks;

Variant 2: For each submatrix S, rescale S to a new probability matrix by $P' = D^{-1}S$, where D is a diagonal matrix whose ith diagonal element is the ith row sum of S. Then, apply the regular spectral clustering algorithm on P' to obtain blocks.

We shall test this algorithm by using an example of NCD Markov chains generated from MARCA software in Section 7.2.

In this chapter, we presented three heuristic methods for decomposing real unsymmetric matrices. They are MARCA, TPABLO and spectral clustering method. our emphasis was on the spectral clustering algorithm procedure for obtaining clusters of graph nodes and blocks of NCD Markov chains. A modified procedure that is suitable for the case of large-scale data matrices was introduced.

Chapter 7

Applications and Experimental Results

Spectral clustering techniques are based on graph partitioning, which is of central importance in map coloring, scheduling and also appears in various forms in parallel computing and other important applications. In recent years, spectral clustering has been proved to be effective in applications of electronic circuit design, image segmentation and neural information processing. In this chapter, we introduce some new applications employing the spectral clustering techniques discussed in this dissertation.

Section 7.1 discusses the application of spectral clustering in software change impact analysis using the signless Laplacian to extract the clustering information of change records of a software system. In Section 7.2, we use the spectral clustering procedures discussed in Section 6.4 to find the nearly-completely-decomposable components of a large-scale NCD Markov chain. Incorporating the idea of modeling with Markov chains, spectral clustering techniques can also be applied to detect community structures in complex networks. This is described in Section 7.3.

7.1 Software Change Impact Analysis

In Sherriff et al. [33], the singular value decomposition is used to generate clusters in an analysis that is used to gauge the impact of software changes in a computer system, an analysis referred to as a *Software Change Impact Analysis*. The data is extracted from

a set of change records of the software system and used to construct an analysis matrix that describes the historical change between two files. The singular vectors of the analysis matrix, which constitute the clustering information of change records, are used to guide impact analysis. The analysis matrix used has exactly the same structure as the signless Laplacian matrix M and since the singular vectors of M are also the eigenvectors of M in spectral clustering, it follows that the singular vectors generate the required clusters. An example of such an analysis matrix M is

$$\mathbf{M} = \begin{pmatrix} 25 & 10 & 0 & 0 & 0 \\ 10 & 31 & 21 & 0 & 0 \\ 0 & 21 & 24 & 0 & 0 \\ 0 & 0 & 0 & 15 & 12 \\ 0 & 0 & 0 & 12 & 17 \end{pmatrix}.$$

This matrix depicts a small software system in which each row and column represents a separate file. There are five files in the system. The values in M represent the number of times that each file is changed in a track with another file. For example, File 2 has been changed 10 times with File 1; 21 times with File 3 and 0 times with itself (since $m_{22} = m_{21} + m_{23}$, where m_{ij} is the element of M in ith row and jth column). Similarly, File 5 has been changed 12 times with File 4 and 5 times with itself. We can create an undirected graph based on these change records using each separate file as a vertex. The weight of an edge represents the number of times that two files have been changed together. The self-loops in the graph mean that the file has been changed in isolation from other files. Thus, the adjacency matrix A and diagonal degree matrix D of the graph are

$$\mathbf{A} = \begin{pmatrix} 0 & 10 & 0 & 0 & 0 \\ 10 & 0 & 21 & 0 & 0 \\ 0 & 21 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 12 & 0 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 25 & 0 & 0 & 0 & 0 \\ 0 & 31 & 0 & 0 & 0 \\ 0 & 0 & 24 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 17 \end{pmatrix}.$$

It is easy to verify that the signless Laplacian is $M = D + A$. Hence clustering information concerning change records is obtained by computing the singular vectors (eigenvectors) of M. Since the graph is unconnected, the nonnegative matrix M is reducible: the first singular vector can be combined with the second to identify potential association clusters.

The singular vectors and singular values of M are, respectively,

$$\mathbf{U} = \mathbf{V} = \begin{pmatrix} -0.29 & 0 & 0.9 & 0.31 & 0 \\ -0.76 & 0 & -0.02 & -0.65 & 0 \\ -0.59 & 0 & -0.43 & 0.69 & 0 \\ 0 & -0.68 & 0 & 0 & -0.74 \\ 0 & -0.74 & 0 & 0 & 0.68 \end{pmatrix},$$

$$\mathbf{D} = \begin{pmatrix} 51.1 & 0 & 0 & 0 & 0 \\ 0 & 28.4 & 0 & 0 & 0 \\ 0 & 0 & 24.8 & 0 & 0 \\ 0 & 0 & 0 & 4.1 & 0 \\ 0 & 0 & 0 & 0 & 3.9 \end{pmatrix}.$$

These matrices show that Files 4 and 5 are strongly linked in isolation from the rest of the system. Also, Files 1, 2 and 3 are linked. In this particular example, a high singular value indicates that that association cluster is more prominent in matrix M, due to a greater number of changes that have occurred to that set of files.

Based on the graph and analysis matrix M, we can also obtain the unoriented incidence matrix I_G, where $M = I_G {I_G}^T$.

$$\mathbf{I_G} = \begin{pmatrix} \sqrt{15} & \sqrt{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{10} & \sqrt{21} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{21} & \sqrt{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{3} & \sqrt{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{12} & \sqrt{5} \end{pmatrix}.$$

Again the singular value decomposition is applied to I_G and we find

$$\mathbf{U} = \begin{pmatrix} 0.29 & 0 & -0.9 & 0.31 & 0 \\ 0.76 & 0 & 0.02 & -0.65 & 0 \\ 0.59 & 0 & 0.43 & 0.69 & 0 \\ 0 & 0.68 & 0 & 0 & -0.74 \\ 0 & 0.74 & 0 & 0 & 0.68 \end{pmatrix},$$

$$\mathbf{V} = \begin{pmatrix} 0.16 & 0 & -0.70 & 0.60 & -0.00 & 0.34 & 0.09 \\ 0.46 & 0 & -0.56 & -0.53 & 0.00 & -0.41 & -0.10 \\ 0.86 & 0 & 0.41 & 0.08 & -0.00 & 0.29 & 0.07 \\ 0.14 & 0 & 0.15 & 0.59 & 0.00 & -0.76 & -0.19 \\ 0 & 0.22 & 0 & 0 & -0.64 & -0.18 & 0.71 \\ 0 & 0.92 & 0 & 0 & -0.10 & 0.09 & -0.36 \\ 0 & 0.31 & 0 & 0 & 0.76 & -0.14 & 0.55 \end{pmatrix},$$

$$\mathbf{D} = \begin{pmatrix} 7.15 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4.98 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.02 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.99 & 0 & 0 \end{pmatrix}.$$

The matrix U is also the matrix of eigenvectors (singular vectors) of M, and provides clustering information for the five files (vertices) in the software system. The matrix V provides the information for the interaction (edges) of these five files. The singular values of I_G are the square root of the eigenvalues of M.

In this application, the signless Laplacian provides us a general way of modeling a set of change records of a software system for clustering files within the system. Our approach shows that clustering analysis in software change impact analysis is based on the graph partitioning objective of maximum association. The SVD on a node-edge incidence matrix can also generate the clusters of files in a set of change records.

7.2 NCD Markov Chains

The right-hand eigenvectors of the transition probability matrix P provide state clustering information for Markov chains based on two clustering techniques. The eigenvector corresponding to the second largest eigenvalue of P in the sense of graph balanced minimum cut is generally the most important in this regard. Let us apply our spectral clustering procedures on problems of NCD Markov chains to decompose transition probability matrices into blocks.

Example 7.2.1. Consider the 8×8 Courtois transition probability matrix P [35] with

NCD structure shown below,

$$\mathbf{P} = \left(\begin{array}{cccc|cc|cc} 0.85 & 0.0 & 0.149 & 0.0009 & 0.0 & 0.00005 & 0.0 & 0.00005 \\ 0.1 & 0.65 & 0.249 & 0.0 & 0.0009 & 0.00005 & 0.0 & 0.00005 \\ 0.1 & 0.8 & 0.0996 & 0.0003 & 0.0 & 0.0 & 0.0001 & 0.0 \\ \hline 0.0 & 0.0004 & 0.0 & 0.7 & 0.2995 & 0.0 & 0.0001 & 0.0 \\ 0.0005 & 0.0 & 0.0004 & 0.399 & 0.6 & 0.0001 & 0.0 & 0.0 \\ \hline 0.0 & 0.00005 & 0.0 & 0.0 & 0.00005 & 0.6 & 0.2499 & 0.15 \\ 0.00003 & 0.0 & 0.00003 & 0.00004 & 0.0 & 0.1 & 0.8 & 0.0999 \\ 0.0 & 0.00005 & 0.0 & 0.0 & 0.00005 & 0.1999 & 0.25 & 0.55 \end{array}\right).$$

Its eigenvalues are given by

$$\lambda = \left(\begin{array}{cccccccc} 1.0000 & 0.9998 & 0.9985 & 0.7500 & 0.5501 & 0.4000 & 0.3007 & -0.1495 \end{array} \right).$$

Observe that the eigenvalues $\lambda_2 = 0.9998$ and $\lambda_3 = 0.9985$ are very close to the unit eigenvalue 1, and also very close to each other. If we set two parameters of the clustering procedure $\alpha = \beta = 0.95$, then the second and third right-hand eigenvectors v_2 and v_3 are both important for clustering because $\lambda_2 > \alpha$ and $\lambda_3/\lambda_2 > \beta$, but v_4 is of little importance due to $\lambda_4/\lambda_3 < \beta$. The corresponding right-hand eigenvectors of λ_2 and λ_3 are

$$\mathbf{v_2}, \mathbf{v_3} = \left(\begin{array}{cc} -0.3536 & -0.4876 \\ -0.3536 & -0.4878 \\ -0.3536 & -0.4883 \\ -0.3536 & 0.3783 \\ -0.3536 & 0.3777 \\ 0.3536 & 0.0073 \\ 0.3536 & 0.0073 \\ 0.3536 & 0.0073 \end{array}\right),$$

These two eigenvectors can be used to obtain the NCD blocks on the Markov chain. Based on v_2, the first five states form one cluster, the remaining states form the other cluster. After checking v_3, it is apparent that although the states can still be partitioned into two clusters, the last two states of the first group can be separated out. In this way, using both v_2 *and* v_3, the three blocks of states in this NCD Markov chain can be found. They are {1,2,3}, {4,5} and {6,7,8}.

Now we present another example to test our spectral hierarchical clustering algorithm for decomposing a large-scale NCD matrix into blocks.

Example 7.2.2. One NCD matrix corresponding to an interactive computer system is generated from the package MARCA due to Stewart [36]. We compared the partition result from our algorithm to those obtained by MARCA and TPABLO.

In the algorithm of MARCA, a parameter γ, which can be varied from 10^{-10} to 10^{-1}, is introduced. All elements of a matrix whose magnitudes are less than or equal to γ are discarded. The remaining matrix is treated as a directed graph, and the strongly connected components in this graph are determined as the NCD blocks. The NCD example matrix we tested has $23,426$ states and $156,026$ nonzero elements. The results given by MARCA with $\gamma = 10^{-2}$ and TPABLO with $\alpha = \beta = 0.5, \gamma = 10^{-2}$ are the same, i.e., 51 blocks have been found with the smallest size 1 and the largest size $1,326$.

Using our hierarchical algorithm, we compute first 20 largest real eigenvalues and their corresponding right-hand eigenvectors of P associated with this NCD chain. Then K-means clustering algorithm on these 20 eigenvectors provides 20 relative smaller submatrices than the original P. After applying TPABLO or MARCA with the same parameters as before on each submatrix, we finally obtain the desired 51 blocks.

Our approach successfully provides NCD components of an NCD Markov chain and shows that finding this NCD structure has quite close relation with the minimum cut of graph partitioning in spectral clustering. The number of NCD components can be determined by the number of eigenvalues of P close to 1.

In next section, we shall show some examples on which our spectral clustering algorithm can produce desired clusters, but MARCA and TPABLO fail.

7.3 Community Networks

Spectral clustering can be applied to detect community structures in a large complex network, depicted as a Markov random walk on a graph. We use our clustering algorithm to decompose the transition probability matrix P of this Markov random walk into blocks where each block represents an actual community in the network. In this situation of decomposing a matrix, MARCA and TPABLO fail to provide the meaningful blocks.

We show three real-world networks, which have been commonly used in research papers that discuss the community problems. Here, since all these networks are represented by graphs, our new spectral techniques on the random walks of these graphs can be applied.

Example 7.3.1. Karate club network in an American university analyzed by Zachary [41].

This network consists of 34 members of a karate club. Zachary used a weighted graph to depict this network, and observed that this club had been divided into two groups in practice. After formulating a random walk on this network using $P = D^{-1}A$, where A is an adjacency matrix of the graph, we apply the MARCA and TPABLO algorithms on P to find a partition of the nodes. However MARCA and TPABLO fail to give the real partition of this network observed by Zachary based on the input parameters. If calculating the second largest eigenvalue of P, $\lambda_2 = 0.8899$ and its corresponding right-hand eigenvector v_2, because we already know that this club has been separated into two groups (the number of clusters is given), we only use v_2 to group nodes into two clusters $\{1-8, 11-14, 17, 18, 20, 22\}$ and $\{9, 10, 15, 16, 19, 21, 23-34\}$. This is the same to the actual structure of this Karate club observed by Zachary after the split.

If we remove the weights from the graph and only analyze the graph structure, the eigenvalues of P now are $\lambda_2 = 0.8677, \lambda_3 = 0.7130, \lambda_4 = 0.6127, \lambda_5 = 0.3878, \ldots$. Since this time the number of clusters is not known, we set $\alpha = \beta = 0.8$, then $\lambda_2 > \alpha$ and $\lambda_3/\lambda_2 = 0.8217 > \beta$, $\lambda_4/\lambda_3 = 0.8593 > \beta$, $\lambda_5/\lambda_4 = 0.6329 < \beta$. The important eigenvectors for clustering are v_2, v_3 and v_4. Therefore we use these three eigenvectors to group the nodes into four clusters. Observe that the eigenvalue with the second largest modulus is positive. The K-means method gives clusters $\{1-4, 8, 12-14, 18, 20, 22\}, \{5-7, 11, 17\}, \{9, 10, 15, 16, 19, 21, 23, 27, 30, 31, 33, 34\}$ and $\{24-26, 28, 29, 32\}$. This result coincides with the 'optimal' partition of Li, Zhang et al.[21] which is shown in Figure 7.1. In the work of Li, Zhang et al., they use a quantitative function for community partition, called modularity density or D-value. To detect communities in a network, they try to find a partition such that the modularity density D is maximized. However, the search for optimal D is a NP-hard problem [21].

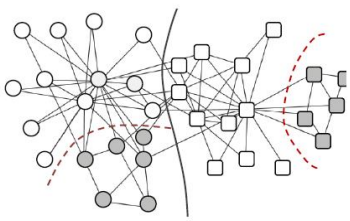

Figure 7.1: Karate club network

If we apply MARCA and TPABLO on P, they also fail to provide the actual two groups of the club after the split or the result of four groups in Figure 7.1.

Example 7.3.2. Journal citation network constructed by Rosvall and Bergstrom [32].

In this network, 40 journals from four different fields: physics, chemistry, biology and ecology, are considered as nodes. In each field, 10 journals with the highest impact factor were selected. There are 189 links connecting nodes if one or more articles from one journal cite articles in another journal during 2004. Using the method of analyzing the random walk on this network, we obtain a probability matrix as $P = D^{-1}A$. We first use MARCA and TPABLO to analyze P. However they fail to generate the actual structure of this journal citation network with four fields.

Using the spectral algorithm, we compute the eigenvalues of P, $\lambda_2 = 0.8813$, $\lambda_3 = 0.7301$, $\lambda_4 = 0.6209$ and $\lambda_5 = 0.2932$. If the two parameters of our clustering algorithm are $\alpha = \beta = 0.8$, then the eigenvectors v_2, v_3 and v_4 are marked as important for partitioning. Therefore we use these three eigenvectors to group nodes into four clusters. The K-means method finally gives clusters $\{1, \ldots, 10\}$ as physics, $\{11, \ldots, 20\}$ as chemistry, $\{21, \ldots, 30\}$ as biology and $\{31, \ldots, 40\}$ as ecology.

Example 7.3.3. The American college football network during the regular fall 2000 season [15].

There are 115 teams denoted by nodes on this network. The teams are divided into 12 conferences containing around 8-12 teams each. After formulating a Markov random walk on this network using $P = D^{-1}A$, we first run Marca and TPABLO to analyze P. The partitioning results of 115 teams from them are far from the actual conferences structure in the real world.

Using our clustering algorithm mentioned in Section 6.4, we then solve the eigenvalue problem of P of the associated Markov chain. The eigenvalues from λ_2 to λ_{13} are:

$$\begin{array}{cccccc} 0.8632 & 0.8171 & 0.7749 & 0.7604 & 0.7177 & 0.7001 \\ 0.6753 & 0.6227 & 0.5900 & 0.5419 & 0.4488 & 0.3740 \end{array}$$

If the two parameters of our clustering algorithm are $\alpha = 0.8$ and $\beta = 0.9$, then we shall obtain the important eigenvectors for clustering $v_2, \ldots, v_{11}$. This implies that our algorithm

produces 11 conferences in this American college football network. Actually it is reasonable. In the original conference organization, five members of the $12th$ conference have few edges between them, therefore in our result, these five nodes are distributed to other clusters. Our result coincides with the observation of Li, Zhang et al.[21]. Meanwhile, one new community, nodes $\{37, 59, 60, 64, 98\}$, is constructed in Li, Zhang et al., which is not recommended by the eigenvalues and eigenvectors of P. In our result, the five nodes in this small new community are assigned to the conferences with which they are most closely associated. For example, 59-"LouisianaTech" 60-"LouisianaMonroe" 64-"MiddleTennesseeState" are assigned to the Southeastern. This is why 11 clusters are recommended on this network. Using our approach we find

1. Cluster 1: $\{4, 6, 11, 41, 53, 73, 75, 82, 85, 99, 103, 108\}$,
2. Cluster 2: $\{12, 25, 29, 51, 70, 91\}$,
3. Cluster 3: $\{2, 26, 34, 38, 46, 90, 104, 106, 110\}$,
4. Cluster 4: $\{20, 30, 31, 36, 56, 80, \underline{81}, \underline{83}, 95, 102\}$,
5. Cluster 5: $\{47, 50, 54, 68, 74, 84, 89, 111, 115\}$,
6. Cluster 6: $\{3, 7, 14, 16, 33, 40, 48, 61, 65, 101, 107\}$,
7. Cluster 7: $\{8, 9, 22, 23, 52, 69, 78, 79, 109, 112\}$,
8. Cluster 8: $\{45, 49, 58, 67, 76, 87, 92, 93, \underline{98}, 113\}$,
9. Cluster 9: $\{13, 15, 19, 27, 32, 35, \underline{37}, 39, 43, 44, 55, 62, 72, 86, 100\}$,
10. Cluster 10: $\{1, 5, 10, 17, 24, 42, 94, 105\}$,
11. Cluster 11: $\{18, 21, 28, 57, \underline{59}, \underline{60}, 63, \underline{64}, 66, 71, 77, 88, 96, 97, 114\}$.

Since 12 conferences exist in the college football network, we can also apply the K-means method with $k = 12$ on the corresponding eigenvectors $v_2, \ldots, v_{12}$ to get 12 clusters. The nodes that are underlined in the 11 clusters above are collected to form a new cluster (community) {37,59,60,64,81,83,98}, and the other nodes are not changed. Therefore, the only difference now with the result of Li, Zhang et al., is the nodes 81, 83 in this new

community. In the real conference partition, nodes 81, 83 are in the Independents with node 37, which means nodes 81, 83 in this new cluster is reasonable. However, this new cluster has few edges between each node rather than other clusters, so the 11 conferences recommended by our clustering algorithm is an appropriate choice.

In this chapter, three applications of spectral clustering techniques are presented: (1) software change impact analysis, (2) finding NCD blocks of a matrix and (3) obtaining reasonable communities structure in complex real-world networks. As we have seen, spectral clustering algorithms are effective at clustering graph nodes and decomposing matrices into blocks. Although spectral based methods are expensive in computation and complexity compared with MARCA and TPABLO algorithms, it works well in the case of studying the community structure in practice where MARCA and TPABLO do not.

Chapter 8

Conclusions

In this dissertation, we introduced the concept of modeling the dual problem of graph partitioning, namely graph clustering, using the signless Laplacian matrix. We presented a new spectral method which shows that the eigenvector associated with the second largest eigenvalue of the signless Laplacian provides a heuristic solution for clustering nodes based on a maximum association objective. This clustering result coincides with the graph partitioning solution based on minimum cut. The spectral algorithm generates the cluster values (eigenvalues of the signless Laplacian) which can be compared with cut values (eigenvalues of the Laplacian). We also extended the spectral clustering technique to a graph with negative edge weights. One application of partitioning a graph having negative edge weights is that involving the clustering analysis of a multivariate data matrix. The result obtained from spectral graph partitioning is related to the result obtained from principal components analysis.

In addition, we reveal the connection between the SVD and eigendecomposition in applications of spectral graph partitioning and clustering. The clustering from an SVD is in the sense of graph partitioning. We showed that the SVD of the node-edge incidence matrix can generate edge clusters of a graph and not just node clusters. This idea is generalized to a methodology for clustering two classes of states, i.e., applying SVD on a rectangle incidence matrix to show the relationship between the two classes. We also presented an application to simultaneously obtain clusters of terms and documents from a term-document matrix.

Depicting spectral clustering in the framework of a Markov random walk on a graph, we were able to propose two different clustering measures on the states of Markov

chains. The first is a partition measure based on the normalized cut and the second, a measure of the distance of states from steady state. We paid particular attention to negative eigenvalues having large modulus, especially if one of them is the subdominant eigenvalue. Such eigenvalues are usually ignored in spectral clustering. It is shown that their corresponding eigenvectors provides important clustering information. We relate these two clustering techniques to obtain more comprehensive information concerning clusters on graph nodes.

A new regular spectral clustering algorithm incorporating two clustering techniques on Markov chains is concluded in this dissertation. According to the control of two parameters, this algorithm can determine the number of clusters on a network, and choose significant eigenvectors of the transition matrix of a Markov random walk to cluster states. We also propose a hierarchical clustering method which is suitable for large-scale matrices, especially in the application of NCD Markov chains. Experiment results show that our algorithms work well.

We applied our algorithm to real-world networks and obtained reasonable communities structure on these networks as well. If a Markov chain generated on a network is periodic, then we can not use the second measure on states of Markov chains to interpret the clustering information. However we can also get the communities of this network in the sense of normalized cut. For some networks with special structure, e.g. cycle graph and complete graph, all nodes have the same role so that it makes no sense to cluster. These kinds of special structures can be detected by their spectrums before the clustering algorithm is applied.

Bibliography

[1] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. 9th International Conference on Artificial Intelligence and Statistics, 2003.

[2] P. Chan, M. Schlag, and J. Zien, Spectral k-way ratio cut partitioning, IEEE Trans. CAD-Integrated Circuits and Systems, Vol. 13, pp. 1088-1096, 1994.

[3] H. Choi and D. B. Szyld, Application of threshold partitioning of sparse matrices to Markov chains.

[4] F. R. K. Chung, Spectral Graph Theory, Providence, RI: Amer. Math. Soc., 1997.

[5] D. Cvetkovic, M. Doob, and H. Sachs. Spectra of Graphs. Theory and application. J. A. Barth Verlag, Leipzig, 3rd edition, 1995.

[6] D. Cvetkovic, P. Rowlinson, and S. K. Simic. Signless Laplacians of finite graphs. Linear Algebra and its Applications, 423:155-171, 2007.

[7] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 26-29, 2001, San Francisco, California, USA.

[8] I. S. Dhillon, Y. Guan, and B. Kulis, A unified view of kernel k-means, spectral clustering and graph partitioning, UTCS Technical Report #TR-04-25, June 30, 2004.

[9] W. E, T. Li, E. Vanden-Eijnden, Optimal partition and effective dynamics of complex networks, *Proc. Natl. Acad. Sci. USA* 105, 7907-7912, 2008.

[10] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. Technical Report 82CRD130, GE Corporate Research, 1982.

[11] M. Fiedler. Algebraic connectivity of graphs. Czecheslovak Mathematical Journal, 23:298-305, 1973.

[12] M. Fiedler. A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. Czechoslovak Mathematical Journal, 25:619-632, 1975.

[13] M. Filippone, F. Camastra, F. Masulli and S. Rovetta. A survey of kernel and spectral methods for clustering. Pattern Recognition, Vol. 41, No. 1., pp. 176-190, 2008.

[14] M.R.Garey and D.S.Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H.Freeman & Company, 1979.

[15] M. Girvan, M. E. J. Newman, Community structure in social and biological networks, *Proc. Natl. Acad. Sci. USA* 99, 7821-7826, 2002.

[16] G.H. Golub and C.F.Van Loan. Matrix computations. Johns Hopkins University Press, 3rd edition, 1996.

[17] W. Hardle and L. Simar, Applied Multivariate Statistical Analysis, Springer, 2nd edition, 2007

[18] D. J. Higham and M. Kibble. A unified view of spectral clustering. University of Strathclyde Mathematics Research Report, 2004.

[19] P. V. Hilgers and A. N. Langville. The five greatest applications of Markov Chains. 2006.

[20] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 29(2): 291-307, 1970.

[21] Z. Li, S. Zhang, R. S. Wang, X. S. Zhang, L. Chen, A novel quantitative function for community detection, *Physical Review E.*, 77 (1), 036109, 2008.

[22] N. Liu and W. J. Stewart, Clustering procedures for Graphs and Markov chains.

[23] C. D. Manning, P. Raghavan and H. Schutze. Introduction to information retrieval, Cambridge University Press, 2008.

[24] M. Meila and J. Shi. Learning segmentation by random walks, in: NIPS, pp. 873-879, 2000.

[25] M. Meila and J. Shi. A random walks view of spectral segmentation, 2001.

[26] Carl D. Meyer. Matrix analysis and applied linear algebra. SIAM, 2000.

[27] A. Y. Ng, M. I. Jordan and Y. Weiss, On spectral clustering: Analysis and an algorithm, Advances in Neural Information Processing Systems 14, pages 849-856, Cambridge, MA, MIT Press, 2002.

[28] C. Nicholas and R. Dahlberg. Spotting topics with the singular value decomposition. PODDP'98, LNCS 1481, pp.82-91, 1998.

[29] J. O'Neil and D. B. Szyld, A block ordering method for sparse matrices.

[30] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. SIAM Journal on Matrix Analysis and Applications, 11(3):430-452, July 1990.

[31] W.H. Press, S.A.Teukolsky, W.T. Vetterling, B.P. Flannery. Numerical recipes in Fortran 77: the art of scientific computing, Cambridge University Press, 1994.

[32] M. Rosvall and C. T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, *Proc. Natl. Acad. Sci. USA* 104, 7327-7331, 2007.

[33] M. Sherriff, and L. Williams. Empirical software change impact analysis using singular value decomposition. International Conference on Software Testing, Verification, and Validation, Lillehammer, Norway, April 9-11, 2008.

[34] J. Shi and J. Malik, Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22:888-905, 2000.

[35] W. J. Stewart, Introduction to the Numerical Solution of Markov Chains, Princeton University Press, 1994.

[36] W. J. Stewart, MARCA: Markov chain analyzer, Numerical Solution of Markov Chains, pp. 37-61, Marcel Dekker, New York - Basel - Hong Kong, 1991.

[37] W. J. Stewart, MARCA: Markov chain analyzer a software Package for Markov modelling, Version 3.0, 1996.

[38] D. Verma and M. Meila, A comparison of spectral clustering algorithms, UW CSE Technical report 03-05-01, 2003.

[39] Douglas B. West. Introduction to graph theory. Prentice Hall, second edition.

[40] S. X. Yu and J. Shi, Multiclass spectral clustering, International Conference on Computer Vision, Nice, France, 11-17 Oct 2003.

[41] W. W. Zachary, An information flow model for conflict and fission in small groups, *J. Anthropol. Res.* 33, 452-473, 1977.

Appendix

The Academic Ranking of World Universities (ARWU) is published by the center for world-class universities and the institute of higher education of Shanghai Jiao Tong University. ARWU is updated on an annual basis. More than 1000 universities are ranked by ARWU each year and the top 100 universities in 2005 are listed in Table A.1. For each university observation, there are eight variables.

X_1: Rank number,
X_2: Name of the institution,
X_3: Score on the number of alumni,
X_4: Score on the number of staff winning Nobel Prizes and Fields Medals,
X_5: Score on the number of highly cited researchers selected by Thomson Scientific,
X_6: Score on the number of articles published in journals of *Nature* and *Science*,
X_7: Score on the number of articles indexed in Science Citation Index,
X_8: Score on per capita performance with respect to the size of an institution.

Table A.1: Top 100 universities of ARWU in 2005.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
1	Harvard Univ	100	100	100	100	100	72.4
2	Univ Cambridge	99.8	93.4	53.3	56.6	70.9	66.9
3	Stanford Univ	41.1	72.2	88.5	70.9	72.3	65
4	Univ California - Berkeley	71.8	76	69.4	73.9	72.2	52.7
5	Massachusetts Inst Tech (MIT)	74	80.6	66.7	65.8	64.3	53
6	California Inst Tech	59.2	68.6	59.8	65.8	52.5	100
7	Columbia Univ	79.4	60.6	56.1	54.2	69.5	45.4
8	Princeton Univ	63.4	76.8	60.9	48.7	48.5	59.1
9	Univ Chicago	75.6	81.9	50.3	44.7	56.4	42.2
10	Univ Oxford	64.3	59.1	48.4	55.6	68.4	53.2

Table A.1: Continued.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
11	Yale Univ	52.1	44.5	60.3	57.2	63.9	49.3
12	Cornell Univ	46.5	52.4	55	48.8	66.3	39.8
13	Univ California - San Diego	17.7	34.7	59.8	56.5	64.5	46.6
14	Univ California - Los Angeles	27.3	32.8	56.7	50.1	75.6	34.3
15	Univ Pennsylvania	35.5	35.1	56.7	42.9	71.8	39.1
16	Univ Wisconsin - Madison	43	36.3	52.1	46.3	68.7	29
17	Univ Washington - Seattle	28.8	32.4	53.9	47.1	73.8	27.2
18	Univ California - San Francisco	0	37.6	55.6	57.9	58.8	45.2
19	Johns Hopkins Univ	51.4	28.3	41.6	52.2	67.7	24.9
20	Tokyo Univ	36	14.4	38.5	52.1	86.5	34.7
21	Univ Michigan - Ann Arbor	43	0	61.9	43	76.5	30.9
22	Kyoto Univ	39.7	34.1	34.2	37	72.3	31.1
23	Imperial Coll London	20.8	38.1	40.8	38.2	64.6	40.3
24	Univ Toronto	28.1	19.7	39.3	38.9	76.7	41.9
25	Univ Illinois - Urbana Champaign	41.6	37.4	44.4	34.1	58	26
26	Univ Coll London	30.7	32.9	37.7	41.5	60.5	38.8
27	Swiss Fed Inst Tech - Zurich	40.2	37	35.1	41.1	43.4	52.4
28	Washington Univ - St. Louis	25.1	26.6	38.5	46.5	53.9	39.9
29	New York Univ	33.8	25	43	35.3	55.4	26.3
30	Rockefeller Univ	22.6	59.8	28.3	44.1	24	35.9
31	Northwestern Univ	21.7	19.3	44.4	33.8	57.6	36.2
32	Duke Univ	20.8	0	47.1	45.3	60.8	38.9
32	Univ Minnesota - Twin Cities	36	0	49.7	35.2	68.4	23.8
34	Univ California - Santa Barbara	0	36	42.3	39	44.1	35.8
35	Univ Colorado - Boulder	16.6	29.8	40.8	36.6	46.3	29.5
36	Univ Texas - Austin	21.7	17.1	49.1	30	54.8	21.7
37	Univ British Columbia	20.8	19.3	32.4	32.5	60.4	33.9
38	Univ Texas Southwestern Med Center	24.3	33.9	31.4	38.2	37.9	31
39	Pennsylvania State Univ - Univ Park	14	0	45.8	37.9	59.9	24
39	Vanderbilt Univ	12.5	30.2	34.2	24.5	49.2	35.6
41	Univ California - Davis	0	0	46.5	34.5	64	29.8
41	Univ Utrecht	30.7	21.4	27.2	27.3	55.7	25.9
43	Rutgers State Univ - New Brunswick	15.4	20.4	36.9	32.9	47.1	24.1
43	Univ Pittsburgh - Pittsburgh	25.1	0	40.1	25.9	64.3	28.2
45	Karolinska Inst Stockholm	30.7	27.8	33.3	19.7	47.3	25.1
46	Univ Paris 06	35.7	23.9	23.6	24.2	51.2	30
47	Univ California - Irvine	0	30	32.4	28.5	48.2	31.1
47	Univ Edinburgh	22.6	17.1	26.1	35.8	49.4	29.9
47	Univ Maryland - Coll Park	25.9	0	40.8	33.6	54.6	25.6
50	Univ Southern California	0	27.3	37.7	23.6	52.8	25.8

Table A.1: Continued.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
51	Univ Munich	37.1	21.1	15.7	30.4	51.9	30
52	Tech Univ Munich	43	24.1	24.8	20.7	46.5	29.2
53	Univ Manchester	27.3	19.3	23.6	22.6	57.3	30.4
54	Carnegie Mellon Univ	30.7	33.5	32.4	14.7	38.3	31.4
55	Univ North Carolina - Chapel Hill	12.5	0	35.1	32.8	59.5	27.3
56	Australian Natl Univ	17.7	12.9	36.9	29	45.1	27.8
57	Univ Copenhagen	30.7	24.7	23.6	22.8	45.7	27.7
57	Univ Florida	15.4	0	35.1	25	65.2	25.8
57	Univ Zurich	12.5	27.3	19.2	30.3	47.2	30.6
60	Uppsala Univ	25.9	32.9	11.1	28.7	49.1	21.6
61	Univ Paris 11	33.2	34.2	13.6	19.6	44.9	27.9
62	Osaka Univ	12.5	0	23.6	31.1	66.8	29.2
63	Ohio State Univ - Columbus	17.7	0	40.8	21.5	61.2	19.5
64	Univ Bristol	10.9	18.2	30.4	24.5	47.5	27.4
65	Univ Rochester	33.2	9.1	27.2	25.3	43	36.1
65	Univ Sheffield	23.5	14.4	23.6	29.2	46.6	27.1
67	McGill Univ	28.8	0	28.3	23.6	56.8	30
67	Moscow State Univ	51.4	34.9	0	7.5	54	31.6
69	Case Western Reserve Univ	40.7	11.8	20.8	23	44.8	33.7
69	Univ Oslo	25.9	34.1	17.6	18.2	41.5	26.4
71	Univ Heidelberg	10.9	27.7	20.8	20.9	48.1	26.9
72	Univ Leiden	25.1	15.8	27.2	19.3	46.7	28.1
73	Tohoku Univ	18.8	0	19.2	26.9	65.3	29
73	Univ Arizona	0	0	29.4	36.8	55.8	25.7
75	Purdue Univ - West Lafayette	18.8	17.1	27.2	21.4	49.8	19.4
76	Univ Helsinki	18.8	18.2	15.7	21.4	54.5	27.5
77	Michigan State Univ	12.5	0	37.7	26.6	51	18.7
78	Hebrew Univ Jerusalem	33.2	0	23.6	27.1	46.6	26.9
78	Rice Univ	21.7	22.3	23.6	24.4	30.8	31
80	Boston Univ	15.4	0	31.4	28.1	50.8	17.5
80	King's Coll London	16.6	23.5	20.8	17.4	44.6	24.8
82	Univ Melbourne	15.4	14.4	22.2	18.7	53.5	19.9
83	Univ Nottingham	15.4	20.4	20.8	19	45.6	24.8
84	Univ Goettingen	38.7	20.4	15.7	17.5	40.2	24.5
85	Univ Vienna	25.1	15.8	7.9	22.7	52.2	26.4
86	Brown Univ	0	13.9	29.4	25.5	40.7	27.9
87	Indiana Univ - Bloomington	14	18.2	24.8	21.2	42	18.2
87	Univ Basel	25.9	17.5	19.2	21.4	34.5	33.8
89	Texas A&M Univ - Coll Station	0	0	32.4	24.4	55	20.4
90	McMaster Univ	16.6	19.3	22.2	15.9	43.5	23.9

Table A.1: Continued.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
90	Univ Freiburg	25.1	21.4	17.6	18.2	39.5	23.2
92	Univ Strasbourg 1	29.4	22.9	19.2	19.5	32.7	22.4
93	Ecole Normale Super Paris	47.8	25	13.6	18.1	27.2	23.3
93	Stockholm Univ	29.4	30.2	15.7	14.3	35.3	18.9
93	Tokyo Inst Tech	16.6	0	22.2	22.5	50.6	31.5
93	Univ Utah	0	0	30.4	31.4	45.5	24.8
97	Univ Roma - La Sapienza	16.6	15.8	11.1	21.8	54.6	15.1
98	Univ Birmingham	25.1	11.2	22.2	13.4	47	24.5
99	Lund Univ	29.4	0	24.8	19.4	50.1	18.1
100	Tufts Univ	18.8	17.1	20.8	19.1	37.4	25.2