ABSTRACT

JIANG, HANSI. Modularity Component Analysis. (Under the direction of Carl Meyer.)

In data clustering when high dimensional data is involved it is often necessary to cluster the data while reducing the number of dimensions. The principal component analysis (PCA) is a very popular analysis tool to achieve the goals, but one drawback of PCA is that the sparsity of the data may be destroyed while centering. Another data clustering method, the modularity clustering algorithm, does not require data centering, but the theory requires a bipartition of the data and a hierarchy has to be built if the number of clusters is more than two.

In this paper the exact linear relation between the leading eigenvectors of the modularity matrix and the singular vectors of an uncentered data matrix is developed. Based on this analysis the concept of a modularity component is defined, and its properties are developed. It is shown that modularity component analysis can be used to cluster data similar to how traditional principal component analysis is used except that modularity component analysis does not require data centering. Some practical examples of using the modularity component analysis method developed in this paper is provided to corroborate the theoretical results.

Modularity Component Analysis

by
Hansi Jiang

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

Applied Mathematics

Raleigh, North Carolina

2016

APPROVED BY:

Zhilin Li	Shaina Race
Ernie Stitzinger	Carl Meyer Chair of Advisory Committee

DEDICATION

To my Mom and Zheng for their endless love, and to Prof. Meyer for his great help and encouragement.

BIOGRAPHY

The author was born in Beijing, China on February 10, 1987. He spent his first 20 years in Beijing before going to the University of Hong Kong to study Mathematics and Physics. He received his Bachelor's degree in Mathematics and Physics in 2010, then went to North Carolina State University to begin his graduate study in Applied Mathematics. He received his Master's degree in 2012 and decided to stay in NC State to continue his research in linear algebra and data clustering. He joined SAS Institute Inc. in 2011 as a graduate intern and will be a research developer in SAS after graduation.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Prof. Carl Meyer. He is a great mathematician and educator. I deeply appreciate his patience, encouragement, and the tremendous help in my research and my life. I also want to thank his wife, Mrs. Meyer, for caring about me and the delicious dinners.

Next, I want to thank the other three professors that in my committee. Prof. Li always cares about my life and my study. Prof. Stitzinger helped me a lot in many issues in the department. Prof. Race is always there to help.

I also appreciate Ralph Abbey and Phuong Hoang for reading this dissertation and the valuable feedbacks. Thank Le Li for the suggestions about preparing for graduation. Thank Maria Jahja for counting down the dates for me and giving another good name for MCA. Special thanks to Minghui Liu for all the help that I don't even know where to start.

There are a lot of colleagues in SAS, including Gul Ege, Arin Chaudhuri, Alex Chien, Ying Zhu that I want to express my appreciation for the support they gave in my work.

My biggest thanks to my mom, Lijuan Zhu, and my girlfriend, Zheng Wang. Your endless love and support is what makes me keep going.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The purpose of this dissertation is to give theoretical support to use multiple eigenvectors of the modularity matrices in clustering and dimension reduction. In data analysis, sometimes people want to pursue higher accuracy. A more accurate result can give better description of the data. However, sometimes the efficiency of the algorithms is more focused on, especially in the industries where datasets often have larger sizes than in academia. Faster algorithms are more preferred to save more time because the new data is coming in rapidly or the users may request the result to be updated frequently. More efficient algorithms are performed with or without sacrificing the accuracy in the data analysis results. It is almost certain that the best algorithms may have both accuracy and efficiency meeting the desired requirements, but more often people have to choose one between the two. For the data with high dimensions, the data analysis tools that can perform dimension reduction is more favorited since most of the information in the original data will be kept in the results, and the number of dimensions are greatly reduced so the data can be easily described and stored.

In this paper a novel data analysis method that can help to reveal the cluster structures in the data as well as to perform dimension reduction will be introduced. In Chapter 2 and Chapter 3 some important data analysis methods in the literature will be discussed. In Chapter 4 we will give the definition to the modularity components, and in Chapter 5 some important properties of the modularity components will be stated and proven. Chapter 6 contains some experimental results and Chapter 7 is the conclusion.

1.1 Cluster Analysis

Although there is no clear definition about cluster analysis, different people have similar descriptions for it. Webster [83] defines cluster analysis as "a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics." By [36], cluster analysis organizes of a group of patterns into clusters based on similarity. By [75], cluster analysis groups data based on their features and relationships. Although different in expressions, there is something in common in these descriptions. First, clustering analysis is based on the characteristics or features of the data. These characteristics may be already given by the data or to be extracted. Sometimes there are redundant features so some feature selection methods should be applied. Second, clustering analysis picks some measurement to build the similarity among the data points. Different measurements would probably lead to different clustering results. Third, cluster analysis aims to group data points that are similar to each other, and separate the data points that are dissimilar to each other.

In the next section we will talk more about data.

1.2 About Data

1.2.1 Attributes

A data set usually contains some attributes and data points. Attributes are used to describe some basic characteristics of the data points. In [75], attributes are classified into four types:

1. **Nominal**: Variables that are used to distinguish different objects. The information given by nominal variables cannot be used to order objects. Examples: gender, zip codes, travel destinations.

2. **Ordinal**: Variables that provide information to order objects, but the differences between values have no meanings. Examples: Level of education, street numbers.

3. **Interval**: Variables that provide information to order objects, and the differences between values have some meaning, but the ratios between values have no meanings. For example, calendar dates is a interval attribute. There are three days strictly between July 1st, 2015 and July 5th, 2015, but the ratio between the two end dates has no meaning.

4. **Ratio**: Variables that both differences and ratios between different values have meanings. For example, age is a ratio attribute. A 40-year-old man is twenty years older than a 20-year-old woman, and is twice as old as the woman.

Nominal and ordinal attributes are also called categorical attributes, and interval and ratio attributes are also called numeric attributes. In this paper we will assume that the attributes are numeric. It is also noticeable that since the attributes are used to describe the characteristics of the data, it is possible that different attributes have different units and scales. In these cases sometimes it is necessary to normalize the attributes to make them having the same scale to extract useful information. More discussions about normalizing attributes will be given in Chapter 2.

1.2.2 Data Quality

Before applying clustering analysis, it is more desirable to get familiar with the data and perform some preprocessing if necessary. It is quite often that the data contains some noise, outliers or missing values, and the data analysis tools may be sensitive to them. For instance, principle component analysis (PCA) is a very popular data analysis method. The method can give cluster structure of the data while reducing the number of dimensions, and is widely used in image and motion processing [58, 61, 78]. However, it is mentioned in [14] and [33] that it is very sensitive to the outliers. In order to get more reliable clustering results, outlier detection algorithms may be applied before cluster analysis. So before using PCA some outlier detection methods may be applied. Here we list some data quality issues that data analysts may pay attention to. Hodge [31] gives a nice survey about outlier detection methods.

1. **Noise**: Noise is irrelevant or meaningless information in the data. It is often very difficult to eliminate the noise in the data completely, but there are several ways to reduce the effect of noise, such as aggregation and dimension reduction [75]. In the retail industry it is common to aggregate the data up to a high level where the data mining techniques are performed. This is to reduce the effect of noise in both estimation and prediction. In statistics, sometimes the noise is assumed to be

independent and identically distributed (also known as i.i.d.). In this paper, it will be assumed that the noise is distributed independently and identically.

2. **Outliers**: Outliers are the data points that are different from most of other data. Like PCA, some data analysis algorithms are sensitive to the outliers, so it may be necessary to detect the outliers in the preprocessing step. Such techniques are also called anomaly detection. Support vector data description (SVDD) [76] is one of the anomaly detection methods. The basic idea of SVDD is to draw a "circle" based on the known information in the training step, and use the circle to identify the outliers in the testing step. More methods about anomaly detection can be found in [75].

3. **Missing values**: Missing value occurs when there is no information provided for a data point in some feature. For example, the rating of a movie given by a user may be missing just because the user has not watched the movie. The missing values can be eliminated, estimated or ignored with some advantages or disadvantages. In this paper, it will be assumed that the missing values, if they exist, are properly handled so they will not get in our way.

1.3 Practical Questions about Cluster Analysis

1.3.1 How to Get Better Clustering Results?

Although there are countless clustering algorithms, there is no one that is better than the others. This is mentioned in [44] and [48] and referred to as "The Fundamental Theorem of Cluster Analysis":

Theorem 1.1 (The Fundamental Theorem of Cluster Analysis)**.** *There does not exist a best method, that is, one which is superior to all other methods, for solving all problems in a given class of problems.*

A realistic problem we have to face is that different clustering algorithms performed on the same data may give different clustering results because they use the same information differently. Even the same clustering algorithm applied on the same data may give different results due to different initial points given to the algorithm. This disagreement from the algorithms may bring confusion and difficulty in selecting proper algorithms to analyze the data. A promising solution to this problem is the idea of consensus clustering [59, 57, 67]. The basic idea of consensus clustering is that, if two objects are supposed to be in the same cluster, then the majority of the algorithms should agree to group them together. On the other hand, if two objects are supposed to be separated to different clusters, then the majority of the algorithms should agree to separate them. One example of consensus clustering is introduced in [84]. In the paper different clustering algorithms are run with different initial settings and number of clusters. An adjacency matrix is built for each run to represent if two data points are put in the same cluster or not, and then the sum of all adjacency matrices are calculated for further analysis.

1.3.2 How Many Clusters?

Another practical problem for cluster analysis is to determine the number of clusters. In fact, the answer to the problem is it depends on how to "look at" the data. For example, some people may think the Ruspini dataset [71] in Figure 1.1 has four clusters while some others may think there is only one cluster. Although there is no right or wrong answers to the question, some methods can be applied to help us to get a more desirable number of clusters. The method of using plots of sum of squared error (SSE) or silhouette coefficient versus the number of clusters is introduced in [75]. Meyer and Wessell use the number of eigenvalues in the Perron Cluster discussed in [15] and [16]. Race [66] uses the consensus clustering technique to determine the number of clusters. In this paper we will focus on building the modularity components and discussing their properties, so we will just assume in this paper that the number of clusters is already given to us.

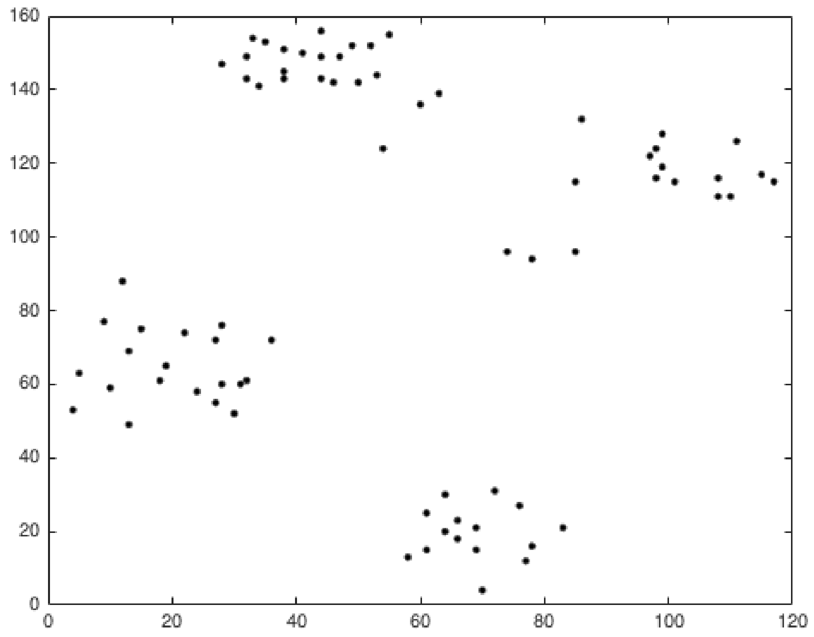

Figure 1.1: Ruspini Dataset.

1.4 Notations

In this paper a lot of mathematical notations will be used. To avoid confusions, here the notations that will be used in this paper will be listed.

1. Capital bold-faced letters ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{X}$, etc.) denote matrices.
2. Lower cased bold-faced letters ($\mathbf{u}$, $\mathbf{v}$, $\mathbf{b}$, etc.) denote vectors.
3. Single subscripts ($\mathbf{v}_i$) of a vector denote the index of a column in a matrix or an eigenvector of a matrix.
4. Greek letters (α, β, λ, etc.) denote the eigenvalues of matrices. Single subscripts of Greek letters (λ_i) denote the indices of eigenvalues of a matrix.
5. Double subscripts ($\mathbf{A}_{ij}$) denote the row and column indices of an entry in a matrix unless with some other explanations.
6. $\mathbf{e}$ denotes a vector with all 1's with proper size.
7. $\| * \|_p$ denotes the p-norm of a vector.

CHAPTER 2

Background and Literature Review

In this chapter we will look at some data analysis methods that can help to reveal the cluster structures in the literature.

2.1 K-means

K-means is a clustering method aims to partition the data points $\mathbf{x}_1$, $\mathbf{x}_2$, $\cdots$, $\mathbf{x}_n$ into k clusters such that each data point belongs to the cluster that has the nearest mean. The means are also called the centroids, and act as the prototypes of the clusters [75]. The term "k-means" was first used in MacQueen's paper [52], although the idea was already used in Steinhaus's paper [73]. The basic algorithm of k-means was proposed by Lloyd [50] and Forgy [25].

When the distance measure is Euclidian distance, the objective function is then the sum of squared error:

$$\text{SSE} = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mathbf{c}_i\|_2^2, \tag{2.1}$$

where the C_i's are the clusters and the $\mathbf{c}_i$'s are the centroids of the clusters. It is proved in [75] that the centroids minimizes Eq. 2.1 are the means:

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j. \tag{2.2}$$

The algorithm of k-means is presented in Algorithm 1.

Algorithm 1 Euclidian k-means

```
Require: Data points $\mathbf{x}_1$, $\mathbf{x}_2$, $\cdots$, $\mathbf{x}_n$, initial centroids $\mathbf{c}_1$, $\mathbf{c}_2$, $\cdots$, $\mathbf{c}_k$.
  • repeat
      Assign each data point to the cluster with nearest centroid with Euclidean distance.
      Calculate the new centroids with Eq. 2.2.
  • until The centroids do not change.
  return Clusters $C_1$, $C_2$, $\cdots$, $C_k$.
```

It is discussed in [75] that k-means may output undesired clusters with poor initial centroids, and the result may be just a local minimum for Eq. 2.1 rather than a global minimum. There are several ways to overcome this disadvantage. One may run several times of the algorithm and pick the result with lowest SSE, or use some other algorithms to help to determine the initial centroids [66], or combine/split the clusters as a postprocessing step [75].

2.2 Principal Component Analysis

It is not rare for real world data to have variables correlated with each other. When the number of variables is more than needed, then feature selection or extraction methods may be necessary. In this case a good clustering algorithm should be able to achieve the following goals:

1. To reveal the cluster structures in the data;
2. To reduce the dimension in the data;
3. To keep as much as possible the useful information in the original data.

The modularity component analysis method we are going to introduce in this paper and the principal component analysis method can both achieve these goals. PCA is first introduced by Pearson [65] and Hotelling [32], while the singular vector decomposition, which underlies PCA, is already discussed in [3] and [42]. Jolliffe's book [40] has more discussions and applications about PCA.

2.2.1 Definition of Principal Components

The main idea of principal component analysis is to reduce the number of dimensions of the original data and to keep as much as possible the variance in the data. The algorithm provides a set of orthonormal vectors called the principal components. The first principal component has the maximal variance in the data, and each successive component has the maximal variance with the constraint that it is orthogonal to the preceding components.

Each data point has a score on each of the components based on how the data is "related with" the component. The principal component analysis can be understood as changing a basis from the elementary basis $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_p\}$ to a new basis $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_p\}$, and the scores are the new coordinates of the data points in the new coordinate system.

To give a mathematical definition of the principal components, suppose we have a $p \times n$ data matrix $\mathbf{X}_0$ where p is the number of variables and n is the number of data points. We want to find a vector $\mathbf{u}_1$ such that $\|\mathbf{u}_1\|_2 = 1$ and the sample variance of the vector

$$\mathbf{u}_1^T \mathbf{X}_0 = \begin{pmatrix} \mathbf{u}_1^T \mathbf{x}_1 & \mathbf{u}_1^T \mathbf{x}_2 & \cdots & \mathbf{u}_1^T \mathbf{x}_n \end{pmatrix}$$

is maximized. The word "sample" means we are dealing with data rather than random variables, and will be omitted in the later discussion for the sake of simplicity. If the mean vector of the $\mathbf{x}_i$'s is $\overline{\mathbf{x}}$, then the problem becomes of solving

$$\max_{\|\mathbf{u}_1\|_2=1} \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{u}_1^T (\mathbf{x}_i - \overline{\mathbf{x}}))^2. \tag{2.3}$$

Subtracting the mean vector from each data point vector is also called *centering.* Eq. 2.3 can be formulated as

$$\max_{\|\mathbf{u}_1\|_2=1} \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1, \tag{2.4}$$

where

$$\mathbf{S}_{p\times p} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T \tag{2.5}$$

is the covariance matrix of the data. It can be also written in the matrix form:

$$\mathbf{S} = \frac{1}{n-1} \mathbf{X}\mathbf{X}^T, \tag{2.6}$$

where

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 - \overline{\mathbf{x}} & \mathbf{x}_2 - \overline{\mathbf{x}} & \cdots & \mathbf{x}_n - \overline{\mathbf{x}} \end{pmatrix} \tag{2.7}$$

represents the centered data. The Rayleigh-Ritz Theorem [51] says the solution to Eq. 2.4 is given by the eigenvector corresponding to the largest eigenvalue λ_1 of $\mathbf{S}$, and the maximum value of Eq. 2.5 is given by λ_1. That means λ_1 is the largest variance in the centered data. The i-th principal components where $i \geq 2$ is the solution to the problem

$$\max_{\substack{\|\mathbf{u}_i\|_2=1 \\ \mathbf{u}_i \perp \mathbf{u}_j, j=1,\cdots,i-1}} \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i, \tag{2.8}$$

and by the Rayleigh-Ritz Theorem $\mathbf{u}_i$ is the eigenvector corresponding to the i-th largest eigenvalue λ_i of $\mathbf{S}$, and the maximum value of Eq. 2.8 is given by λ_i. That means λ_i is the largest variance in the centered data that is uncorrelated with the preceding principal components. Since the eigenvectors of $\mathbf{S}$ and $\mathbf{X}\mathbf{X}^T$ are the same, and the eigenvalues of $\mathbf{S}$ are $1/(n-1)$ proportional to the eigenvalues of $\mathbf{X}\mathbf{X}^T$, the principal components can be also computed from $\mathbf{X}\mathbf{X}^T$, or singular vectors of $\mathbf{X}$.

One point to note is that although in our analysis it is not required that each variable is normalized (i.e. to divide each variable by its 2-norm), sometimes it is still important to do it. Jolliffe [40] discussed the importance of normalizing the variables. When the PCA algorithm is applied on real data, it is highly probable that the variables have different units. Applying PCA without normalizing the variables may cause the first principal component dominated by the variables with larger entries. For example, suppose we have a data with four variables, and the fourth variable is in centimeters while the other three are in meters, but they describe similar length. To illustrate the point more clearly, the variance of the first three variables are set to one, and the variance of the fourth variable is set to 10000. With the variables normalized, the covariance matrix is

$$\mathbf{S}_1 = \begin{pmatrix} 1.0000 & -0.1176 & 0.8718 & 0.8179 \\ -0.1176 & 1.0000 & -0.4284 & -0.3661 \\ 0.8718 & -0.4284 & 1.0000 & 0.9629 \\ 0.8179 & -0.3661 & 0.9629 & 1.0000 \end{pmatrix},$$

and the first principal component is

$$\mathbf{u}_1 = \begin{pmatrix} 0.5211 \\ -0.2693 \\ 0.5804 \\ 0.5649 \end{pmatrix}.$$

The covariance matrix of the unnormalized data is

$$\mathbf{S}_1' = \begin{pmatrix} 1.0000 & -0.1176 & 0.8718 & 81.7941 \\ -0.1176 & 1.0000 & -0.4284 & -36.6126 \\ 0.8718 & -0.4284 & 1.0000 & 96.2865 \\ 81.7941 & -36.6126 & 96.2865 & 10,000 \end{pmatrix},$$

and the first principal component is

$$\mathbf{u}_1' = \begin{pmatrix} 0.0082 \\ -0.0037 \\ 0.0096 \\ 0.9999 \end{pmatrix}.$$

It can be seen that in the unnormalized case, the direction of the first principal component is almost identical to $\mathbf{e}_4$, and that means the data is described mostly by the fourth variable, which is misleading. Therefore, when the data has different scales it is often necessary to normalize the variables after centering. In the following discussion in this section, it will be assumed that the data has been centered and normalized.

2.2.2 Singular Value Decomposition and Scoring the Data

The singular value decomposition (SVD) technique is a matrix factorization method and plays a very important role in PCA. Here we write down the definition of SVD in [56]:

Definition 2.1. *For each $\mathbf{A} \in \mathbb{R}^{m\times n}$ of rank r, there are orthogonal matrices $\mathbf{U}_{m\times m}$, $\mathbf{V}_{n\times n}$ and a diagonal matrix $\mathbf{D}_{r\times r} = diag(\sigma_1, \sigma_2, \cdots, \sigma_r)$ such that*

$$\mathbf{A} = \mathbf{U}\begin{pmatrix}\mathbf{D} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}_{m\times n}\mathbf{V} \text{ with } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0. \tag{2.9}$$

The σ_i's are called the nonzero ***singular values*** *of $\mathbf{A}$. When $r < p = \min\{m, n\}$, $\mathbf{A}$ is said to have $p - r$ additional zero singular values. The factorization in Eq. 2.9 is called a* ***singular value decomposition*** *of $\mathbf{A}$, and the columns in $\mathbf{U}$ and $\mathbf{V}$ are called the left-hand and right-hand* ***singular vectors*** *for $\mathbf{A}$, respectively.*

The SVD is related with PCA in two ways. First, the principal components we defined in Section 2.2.1 can be derived from SVD. To see this, suppose $\mathbf{X}_{p\times n}$ is the data matrix after centering and normalization. Let the SVD of $\mathbf{X}$ to be

$$\mathbf{X} = \mathbf{U\Sigma V}^T = \mathbf{U}\begin{pmatrix}\mathbf{D} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\mathbf{V}^T. \tag{2.10}$$

Then the matrix $\mathbf{X}\mathbf{X}^T$ can be written as

$$\mathbf{X}\mathbf{X}^T = \mathbf{U} \begin{pmatrix} \mathbf{D}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}_{p \times p} \mathbf{U}^T. \tag{2.11}$$

Therefore, the eigenvalues of $\mathbf{X}\mathbf{X}^T$ are the squares of the singular values of $\mathbf{X}$, and the eigenvectors of $\mathbf{X}\mathbf{X}^T$ are the left-hand singular values of $\mathbf{X}$. To calculate the eigenvalues and eigenvectors of $\mathbf{X}\mathbf{X}^T$, it is sufficient to apply SVD on $\mathbf{X}$.

Second, SVD is related to the scores of the data on the principal components. The inner product $\mathbf{u}_j^T\mathbf{x}_i$ is the projection of $\mathbf{x}_i$ onto the span of $\mathbf{u}_j$ and is also called the *score* of the i-th data point on the j-th principal component [40]. From $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ we have

$$\mathbf{U}^T\mathbf{X} = \mathbf{\Sigma}\mathbf{V}^T = \begin{pmatrix} \mathbf{u}_1^T\mathbf{x}_1 & \mathbf{u}_1^T\mathbf{x}_2 & \cdots & \mathbf{u}_1^T\mathbf{x}_n \\ \mathbf{u}_2^T\mathbf{x}_1 & \mathbf{u}_2^T\mathbf{x}_2 & \cdots & \mathbf{u}_2^T\mathbf{x}_n \\ \vdots & & \ddots & \vdots \\ \mathbf{u}_p^T\mathbf{x}_1 & \mathbf{u}_p^T\mathbf{x}_2 & \cdots & \mathbf{u}_p^T\mathbf{x}_n \end{pmatrix}, \tag{2.12}$$

which contains the scores of each data point on each principal component. Therefore, the *score matrix* $\mathbf{U}^T\mathbf{X}$ can be computed with $\mathbf{\Sigma}\mathbf{V}^T$. An advantage of using $\mathbf{\Sigma}\mathbf{V}^T$ to compute the score matrix is the $\mathbf{\Sigma}$ is very sparse and the computations can be very efficient.

2.2.3 Dimension Reduction, Low Rank Representation of Data and Clustering

A very important feature of PCA is it can help to reduce the number of dimensions in the data if there are correlated variables, while keep as much as possible the variance in the original data. The total variance in the data is defined as the sum of the variances in each component, and is just the trace of the covariance matrix $\mathbf{S}$. Since the trace of a matrix is equal to the sum of its eigenvalues, it can be computed that how much variance is kept by the first k principal components:

$$\text{Variance retained}, V_k = \frac{\sum_{i=1}^{k} \lambda_i}{\text{trace}(\mathbf{S})}. \tag{2.13}$$

Therefore, if the sum of the first k eigenvalues of the covariance matrix can take a large portion of the total sum of all eigenvalues, then the first k principal components retain the majority of the variance in the data. The score matrix

$$\mathbf{T}_k = \mathbf{U}_k^T \mathbf{X} = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_k^T \end{pmatrix} \mathbf{X} \tag{2.14}$$

is then a k-dimensional representation of the original data and may keep the cluster structure of the data [40, 68]. Therefore, after dimension reduction the clustering algorithms such as k-means can be applied on the low-rank representation of the data to reveal the clusters. For example, Figure 2.1 is a 2-dimensional representation of the iris data [24]. Although it is argued that sometimes the first principal components may not give clear structure [88], it is pointed out in [40] that this kind of cases is vary rare, since the principal components are calculated with all information of the data. Thus in our experiments, we will combine the PCA with k-means to get the clusters.

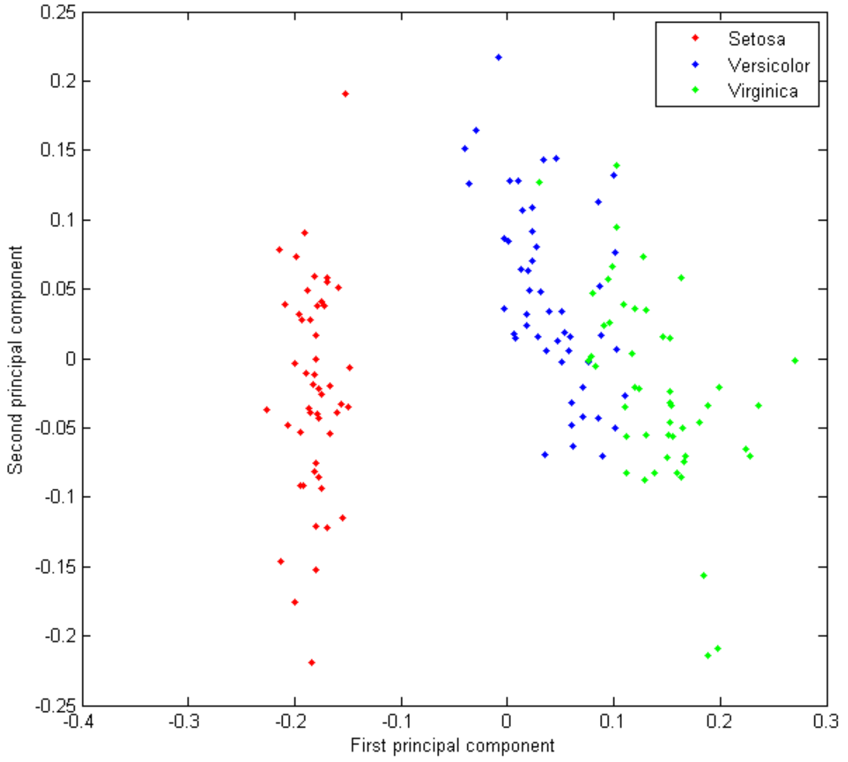

Figure 2.1: A 2-dimensional representation of the iris data given by PCA.

2.2.4 Summary of Properties of Principal Components

Now we summarize the important properties of the principal components. We will first describe the properties, then state the properties again by writing them as theorems. To make the points more clear, it will be assumed that the data matrix $\mathbf{X}$ has full row rank.

- The principal components are orthogonal to each other.
- If we project the data onto the span of a principal component, we get a scalar multiple of the score vector that can reveal the cluster structure in the data based on the signs of the entries in the score vector.
- The first principal component has the largest variance of the data. Each succeeding principal component has the largest variance with the constraint that it is orthogonal to all previous principal components.

Next we state the properties as theorems. Let the $\mathbf{u}_i$'s be the principal components.

Theorem 2.2. *For $1 \leq i, j \leq p$, we have*

$$\mathbf{u}_i^T \mathbf{u}_j = \begin{cases} 1 & i = j, \\ 0 & i \neq j. \end{cases} \tag{2.15}$$

Proof. By definition. □

Theorem 2.3. *Let $\mathbf{P}_{\mathbf{u}_i}$ be the projector onto the span of $\mathbf{u}_i$. Then we have*

$$\mathbf{P}_{\mathbf{u}_i}\mathbf{X} = \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{2.16}$$

where σ_i is the i-th singular value of $\mathbf{X}$ and $\mathbf{v}_i$ is the i-th right-hand singular vector of $\mathbf{X}$.

Proof. Let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the SVD of $\mathbf{X}$.

$$\mathbf{P}_{\mathbf{u}_i}\mathbf{X} = \mathbf{u}_i\mathbf{u}_i^T\mathbf{X} = \mathbf{u}_i\begin{pmatrix}\mathbf{u}_i^T\mathbf{x}_1 & \mathbf{u}_i^T\mathbf{x}_2 & \cdots & \mathbf{u}_i^T\mathbf{x}_n\end{pmatrix} = \mathbf{u}_i\left(\mathbf{\Sigma}\mathbf{V}^T\right)_{i*} = \sigma_i\mathbf{u}_i\mathbf{v}_i^T.$$

□

Theorem 2.4. *Let σ_i be the i-th largest singular value of $\mathbf{X}$. Moreover, for $2 \leq i \leq p$, let*

$$\mathbf{X}_i = \mathbf{X} - \sum_{j=1}^{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{X}, \tag{2.17}$$

then σ_i^2 is the largest eigenvalue of $\mathbf{X}_i\mathbf{X}_i^T$, and $\mathbf{u}_i$ is the corresponding eigenvector of σ_i^2.

Proof. By the discussion in Section 2.2.1. □

After defining the modularity components and stating their properties, we will come back to compare the properties of modularity components with the ones of principal components. Here we write down the algorithm of clustering with PCA as following.

Algorithm 2 Clustering with Principal Component Analysis

```
Require: A $p \times n$ data matrix $\mathbf{X}_0$, number of clusters $k$, number of principal components $t$.
    • Center and normalize $\mathbf{X}_0$ to matrix $\mathbf{X}$.
    • Compute the SVD of $\mathbf{X}$ with Eq. 2.10.
    • Let the score matrix $\mathbf{T}_t$ be the first $t$ rows of the matrix $\mathbf{\Sigma V}^T$.
    • Let $\mathbf{p}_i$ be the $i$-th colomn of $\mathbf{T}_t$.
    • Cluster the $\mathbf{p}_i$'s with k-means into clusters $C_1, C_2, \cdots, C_k$.
return Clusters $G_1, G_2, \cdots, G_k$ such that $G_i = \{j|\mathbf{p}_j \in C_i\}$.
```

2.2.5 How Many Principal Components?

From the discussion above it can be seen that PCA is hoped to perform dimension reduction while retain as much as possible the variance in the original data and keep the cluster structure. Then there is a natural but essential question: How many principal components should be used to achieve these goals? A true but diplomatic answer is “It depends on the data”. Accepting this answer but being a little bit arbitrary, we seek for some rules and research papers that may help to get a proper number of principal components, rather than trying numbers from one to p.

The first method is to set a threshold for the variance to be retained, and use Eq. 2.13 to determine k. The threshold is usually set to be 80% or 90% [40]. Some researchers [53, 74] studied the possible distributions of V_k. Jackson's book [35] contains some other criterion based on V_k.

The second method is just set a threshold on the λ_k in Eq. 2.13. *Kaiser's rule* says that only the λ_k's that exceed 1 and their corresponding principal components should be retained. Jolliffe [41] argues that setting $\lambda_k = 1$ to be the threshold may discard too much useful information, and 0.7 may be a better threshold.

The third method is to plot the λ_k's against the k to get a *scree graph* [11]. Figure 2.2 is an example of a scree graph. People may calculate

$$b_k = \frac{\lambda_{k-1} - \lambda_k}{\lambda_k - \lambda_{k+1}},\ 1 < k < p - 1 \tag{2.18}$$

to get the k^* that has the maximal b_k (also called the "elbow"), and drop the λ_k's that are smaller than k^*. For the wine data [49] that has the scree plot in Figure 2.2, the elbow rule suggests to keep the first four principal components.

There are other techniques that can be applied to determine how many principal components to keep. Statistical methods are studied in [2, 35, 77, 5, 4]. These research focused on using hypothesis tests to determine how many principal component are statistical significant. Wold [86], Eastment and Krzanowski [19], and Krzanowski and Kline [45] studied the residual between the original data and the low-rank estimation of the data given by SVD. The principal components would be dropped if adding it causes no significant decrease to the residual. There are also some papers comparing these methods, such as [69, 27, 20].

2.2.6 Other Research about Principal Components

There are numerous research papers about principal components. Moore et al. [60] proposed a partitioning algorithm named the principal component partitioning algorithm.

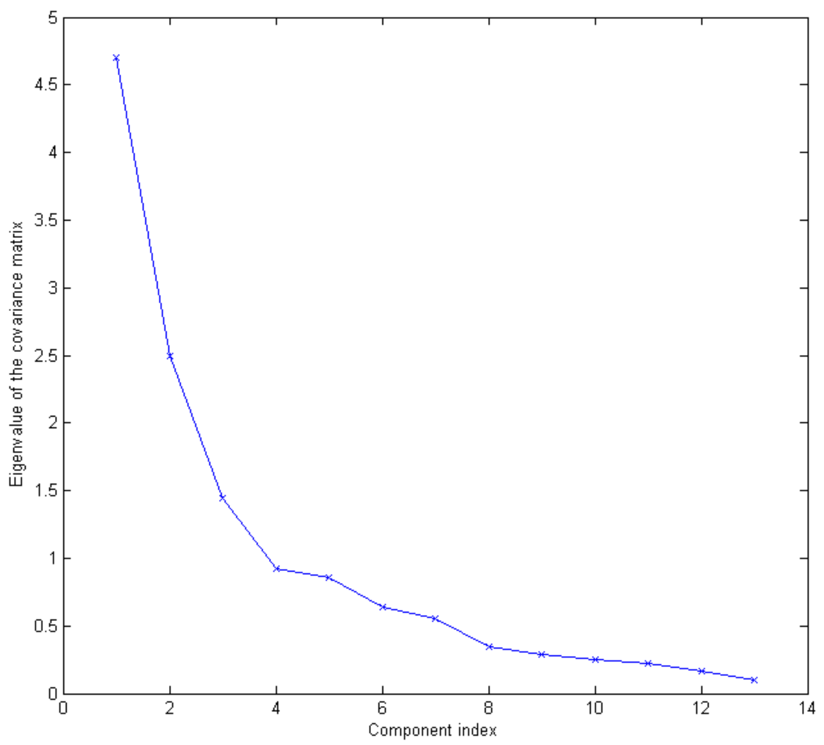

Figure 2.2: Scree graph for the covariance matrix: wine data.

It is mainly used to cluster web pages, and iteratively uses the first principal component to bipartition the data. Boley [7] continued this research and proposed the principal direction divisive partitioning algorithm (PDDP). Candes et al. [10] and Wright et al. [87] considered a way to recover a low-rank representation of the data when the data is corrupted. Wang et al. [82] and Jeng [37] discussed a moving window PCA method that can be used on time series.

2.3 Spectral Clustering

The spectral clustering method is one of the most widely used techniques for graph partitioning. The theory is based on Miroslav Fiedler's research [21, 22, 23]. In this section we will first introduce the classical spectral clustering algorithm, and then some extended versions of spectral clustering will be discussed.

2.3.1 Classical Spectral Clustering

We start with a graph $G(V, E)$ with V the set of vertices and E the set of edges. We assume that the graph G is weighted and undirected, so its adjacency matrix $\mathbf{A}$ is symmetric. The (i, j)-th entry of $\mathbf{A}$ is the weight on the edge between vertices i and j, or zero if there is no edge between them. The graph is *connected* if there is a path between every pair of vertices in the graph. A *connected component* is a maximal connected subgraph of G. Suppose $|V| = n$ and $|E| = m$, then the degree of vertex i is defined by the sum of weights on the edges connected with i, i.e.

$$d_i = \sum_{i=1}^{n} a_{ij}. \tag{2.19}$$

It can be seen that d_i is also the i-th row sum of $\mathbf{A}$. The degree matrix $\mathbf{D}$ is a diagonal matrix with d_i on its i-th diagonal entry. Now we give the definition of the Laplacian matrix.

Definition 2.5. *The Laplacian matrix* $\mathbf{L}$ *of a weighted and undirected graph is defined as*

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2.20}$$

The Laplacian matrix is symmetric and positive semi-definite. The smallest eigenvalue of $\mathbf{L}$ is zero, and its corresponding eigenvector is the vector $\mathbf{e}$. For any vector $\mathbf{v} \in \mathbb{R}^n$, we have [79]

$$\mathbf{v}^T \mathbf{L} \mathbf{v} = \frac{1}{2} \sum_{i,j=1}^{n} a_{ij} (\mathbf{v}_i - \mathbf{v}_j)^2. \tag{2.21}$$

If the graph is not connected, then the number of connected components in the graph is equal to the algebraic multiplicity of the zero eigenvalue of $\mathbf{L}$. To see this, note that the Laplacian matrix of a graph that contains k connected components has the form

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_k \end{pmatrix},$$

where $\mathbf{L}_1$ is the Laplacian matrix of the first connected component, $\mathbf{L}_2$ is the Laplacian matrix of the second connected component, etc. Each of the Laplacian matrices $\mathbf{L}_i$ have one and only one zero eigenvalue, therefore there are k zero eigenvalues of the matrix $\mathbf{L}$

in total. If the graph is connected, then consider the matrix

$$\mathbf{M} = \mathbf{A} - \mathbf{D} + d_{\max}\mathbf{I} = d_{\max}\mathbf{I} - \mathbf{L}, \tag{2.22}$$

where $d_{\max} = \max\{d_1, d_2, \cdots, d_n\}$. Then $\mathbf{M}$ is nonnegative and irreducible. From the Perron-Frobenius theorem [56], the largest eigenvalue of $\mathbf{M}$ is simple. Also notice that there is a relation between the eigenvalues of $\mathbf{M}$ and the eigenvalues of $\mathbf{L}$:

$$\lambda_i(\mathbf{L}) = d_{\max} - \lambda_i(\mathbf{M}), \tag{2.23}$$

therefore the smallest eigenvalue of $\mathbf{L}$ is simple.

In [21], Fiedler defined the second smallest eigenvalue of the Laplacian matrix, λ_2, as the *algebraic connectivity* of the graph. The eigenvector of $\mathbf{L}$ corresponding to λ_2 is called the Fiedler vector. In [22], Fiedler proved that if a connected graph is partitioned by the signs of the Fiedler vector (an arbitrary decision has to be make if some entries are zero), then the two subgraphs are also connected. If the desired number of clusters k is given, then by recursively bipartition the graph with Fiedler vectors, k connected subgraphs of G can be achieved.

2.3.2 Graph Cuts and Spectral Clustering

In this section we will look at some graph cutting techniques and their relation with spectral clustering. Suppose we want to cut a weighted, undirected graph G into k parts. The simplest way to do this is to solve the mincut problem, i.e. to find a partition G_1,

$G_2, \cdots, G_k$, such that

$$\mathrm{cut}(G_1, G_2, \cdots, G_k) = \frac{1}{2}\sum_{i=1}^{k} W(G_i, \overline{G_i}), \tag{2.24}$$

is minimized, where

$$W(A, B) = \sum_{i\in A, j\in B} a_{ij} \tag{2.25}$$

and $\overline{G_i}$ is the complement of G_i. The meaning of Eq. 2.24 is to cut the graph such that minimized number of edges are cut in the process. Although intuitive, in practice the solution to this problem is often helpless, because in many cases the solution put a single vertex into one group and all other vertices in another group. One way to avoid getting this kind of unsatisfactory results is to try to cut the graph while keeping the sizes of the clusters more balanced. Two examples of such cutting techniques are solving the *RatioCut* and *Normalized Cut* problems.

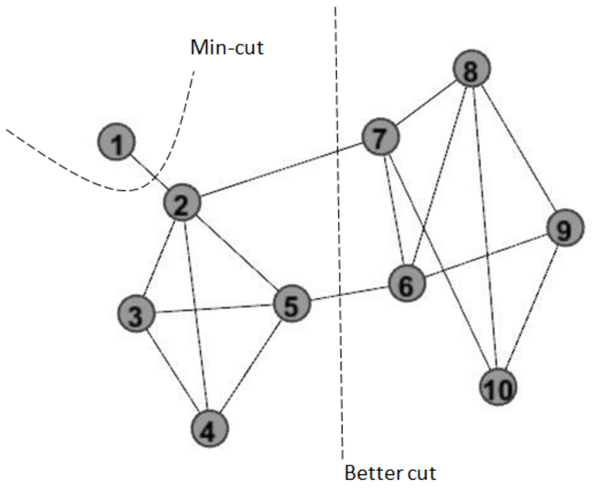

Figure 2.3: A case where minimum cut gives an unsatisfactory partition.

RatioCut

Now we give another way to cut the graph called the RatioCut [30]:

Definition 2.6. *Suppose G_1, G_2, $\cdots$, G_k is a partition of G. The RatioCut is defined by*

$$RatioCut(G_1, G_2, \cdots, G_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(G_i, \overline{G_i})}{|G_i|} = \sum_{i=1}^{k} \frac{cut(G_i, \overline{G_i})}{|G_i|}, \tag{2.26}$$

where $|G|$ is the number of vertices in G.

Suppose a set of k indicator vectors $\mathbf{v}_i$ is defined by

$$\mathbf{v}_{ij} = \begin{cases} \frac{1}{\sqrt{|G_j|}} & \text{if node } i \text{ in } G_j \\ 0 & \text{otherwise.} \end{cases} \tag{2.27}$$

Then it can be seen that the $\mathbf{v}_i$'s are orthonormal to each other. If the columns of the matrix $\mathbf{V} \in \mathbb{R}^{n\times k}$ are the $\mathbf{v}_i$'s, then $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. In [79], it is proven that minimizing the RatioCut is equivalent to minimizing the trace of the matrix $\mathbf{V}^T\mathbf{L}\mathbf{V}$. However, solving the equation with the constraint to the form of $\mathbf{V}$ is NP-hard. Then the constraint to $\mathbf{V}$ is relaxed by allowing the entries in $\mathbf{V}$ to take any real values. The relaxed problem becomes solving

$$\min_{\mathbf{V}\in\mathbb{R}^{n\times k}} \mathrm{Tr}(\mathbf{V}^T\mathbf{L}\mathbf{V}) \text{ subject to } \mathbf{V}^T\mathbf{V} = \mathbf{I}. \tag{2.28}$$

By the Rayleigh-Ritz theorem [51], the columns of $\mathbf{V}$ are formed by the eigenvectors corresponding to the first k smallest eigenvalues of $\mathbf{L}$. The rows in $\mathbf{V}$ can be treated as representatives of the nodes, and other clustering algorithms such as k-means can be used on the rows. This method is also referred as the unnormalized spectral clustering [79] and the algorithm is given as following.

Algorithm 3 Unnormalized Spectral Clustering [79]

```
Require: An $n \times n$ adjacency or similarity matrix $\mathbf{A}$, number of clusters $k$.
    • Compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$.
    • Compute the smallest $k$ eigenvalues of $\mathbf{L}$ and their corresponding eigenvectors.
    • Form the matrix $\mathbf{V}$ with the $k$ eigenvectors of $\mathbf{L}$ as columns.
    • Let $\mathbf{p}_i$ be the $i$-th row of $\mathbf{V}$.
    • Cluster the $\mathbf{p}_i$'s with $k$-means into clusters $C_1, C_2, \cdots, C_k$.
  return Clusters $G_1, G_2, \cdots, G_k$ such that $G_i = \{j|\mathbf{p}_j \in C_i\}$.
```

As an example to illustrate how to use Fiedler vector to partition a graph, we use Algorithm 3 to partition the graph in Figure 2.3. The adjacency matrix of the graph is

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

and the Laplacian matrix is

$$\mathbf{L} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 5 & -1 & -1 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 3 \end{pmatrix}.$$

Then the Fiedler vector, i.e. the eigenvector corresponding to the second smallest eigenvalue, is

$$\mathbf{v}_2 = \begin{pmatrix} -0.5275 \\ -0.2383 \\ -0.2810 \\ -0.2810 \\ -0.1696 \\ 0.2148 \\ 0.1983 \\ 0.3364 \\ 0.3764 \\ 0.3716 \end{pmatrix}$$

By looking at the signs of the entries in $\mathbf{v}_2$, we put the first five vertices into one group, and the last five vertices into the other group.

Normalized Cut

There is another kind of spectral clustering technique that aims to minimize the objective function called *Normalized Cut*, or *NCut* introduced by Shi and Malik [72]:

Definition 2.7. *Suppose G_1, G_2, $\cdots$, G_k is a partition of G. The NCut is defined by*

$$\mathit{NCut}(G_1, G_2, \cdots, G_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(G_i, \overline{G_i})}{\mathit{vol}(G_i)} = \sum_{i=1}^{k} \frac{\mathit{cut}(G_i, \overline{G_i})}{\mathit{vol}(G_i)}, \tag{2.29}$$

where

$$\mathit{vol}(G) = \sum_{i \in G} d_i. \tag{2.30}$$

It can be seen that the technique used in solving the NCut problem is quite similar to the case of solving the RatioCut problem. Suppose a set of k indicator vectors $\mathbf{v}_i$ is defined by

$$\mathbf{v}_{ij} = \begin{cases} \frac{1}{\sqrt{\mathrm{vol}(G_i)}} & \text{if node } i \text{ in } G_j \\ 0 & \text{otherwise.} \end{cases} \tag{2.31}$$

Then the $\mathbf{v}_i$'s are orthonormal to each other. If the columns of the matrix $\mathbf{V} \in \mathbb{R}^{n \times k}$ are the $\mathbf{v}_i$'s, then $\mathbf{V}^T\mathbf{D}\mathbf{V} = \mathbf{I}$. In [79], it is proven that minimizing the NCut is equivalent to minimizing

$$\min_{\mathbf{V}} \mathrm{Tr}(\mathbf{V}^T\mathbf{L}\mathbf{V}) \text{ subject to } \mathbf{V}^T\mathbf{D}\mathbf{V} = \mathbf{I}. \tag{2.32}$$

However, solving this problem with the constraint to the form of $\mathbf{V}$ is NP-hard. Then the constraint to $\mathbf{V}$ is relaxed by allowing the entries in $\mathbf{V}$ to take any real values, and substitute $\mathbf{V}$ by $\mathbf{D}^{-1/2}\mathbf{T}$, where $\mathbf{T} \in \mathbb{R}^{n\times k}$. The relaxed problem becomes solving

$$\min_{\mathbf{T}\in\mathbb{R}^{n\times k}} \mathrm{Tr}(\mathbf{T}^T\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{T}) \text{ subject to } \mathbf{T}^T\mathbf{T} = \mathbf{I}. \tag{2.33}$$

By the Rayleigh-Ritz theorem again, the columns of $\mathbf{T}$ are formed by the eigenvectors corresponding to the first k smallest eigenvalues of

$$\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}. \tag{2.34}$$

Solving back for $\mathbf{V}$ with $\mathbf{V} = \mathbf{D}^{-1/2}\mathbf{T}$, it can be seen that the columns of $\mathbf{V}$ are the the eigenvectors corresponding to the first k smallest eigenvalues of

$$\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L}. \tag{2.35}$$

Then other clustering algorithms such as k-means can be used on the rows of $\mathbf{V}$. This method is also referred as the normalized spectral clustering [79]. Next we give the algorithm of normalized spectral clustering.

Algorithm 4 Normalized Spectral Clustering by Shi and Malik [72]

```
Require: An $n \times n$ adjacency or similarity matrix $\mathbf{A}$, number of clusters $k$.
    • Compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$.
    • Compute the Normalized Laplacian matrix $\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L}$.
    • Compute the smallest $k$ eigenvalues of $\mathbf{L}_{rw}$ and their corresponding eigenvectors.
    • Form the matrix $\mathbf{V}$ with the $k$ eigenvectors of $\mathbf{L}_{rw}$ as columns.
    • Let $\mathbf{p}_i$ be the $i$-th row of $\mathbf{V}$.
    • Cluster the $\mathbf{p}_i$'s with $k$-means into clusters $C_1, C_2, \cdots, C_k$.
  return Clusters $G_1, G_2, \cdots, G_k$ such that $G_i = \{j | \mathbf{p}_j \in C_i\}$.
```

Other Spectral Clustering Algorithms and Discussions

In 2002 Ng, Jordan and Weiss [64] introduced another normalized spectral clustering algorithm. Instead of using the eigenvectors of $\mathbf{L}_{rw}$, they used the eigenvectors of $\mathbf{L}_{sym}$.

```
Algorithm 5 Normalized Spectral Clustering by Ng, Jordan and Weiss [64]
Require: An $n \times n$ adjacency or similarity matrix $\mathbf{A}$, number of clusters $k$.
    • Compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$.
    • Compute the Normalized Laplacian matrix $\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$.
    • Compute the smallest $k$ eigenvalues of $\mathbf{L}_{sym}$ and their corresponding eigenvectors.
    • Form the matrix $\mathbf{V}$ with the $k$ eigenvectors of $\mathbf{L}_{sym}$ as columns.
    • Normalize each row of $\mathbf{V}$ to form a matrix $\mathbf{U}$.
    • Let $\mathbf{p}_i$ be the $i$-th row of $\mathbf{U}$.
    • Cluster the $\mathbf{p}_i$'s with $k$-means into clusters $C_1, C_2, \cdots, C_k$.
  return Clusters $G_1, G_2, \cdots, G_k$ such that $G_i = \{j | \mathbf{p}_j \in C_i\}$.
```

Other than using the eigenvectors of a different Laplacian matrix, Algorithm 8 normalizes each row of the $\mathbf{V}$ matrix. One reason of doing this, as pointed in [79], is because the eigenvector corresponding to the smallest eigenvalue of $\mathbf{L}_{sym}$ is $\mathbf{D}^{1/2}\mathbf{e}$ instead of $\mathbf{e}$. In the indicator matrices in Eq. 2.27 and Eq. 2.31 for the ideal cases, there is one and only one nonzero entry in each row. In the solutions to the relaxed problems (Eq. 2.28 and Eq. 2.33), it can be guaranteed that for each row of the matrix $\mathbf{V}$ there is at least one entry that has value $(1/\sqrt{n})\mathbf{e}$, since $\mathbf{e}$ is the eigenvector corresponding to the zero eigenvalue of $\mathbf{L}$ and $\mathbf{L}_{rw}$. So each row in the $\mathbf{V}$ matrix is "bounded away" from zero. However, if a vertex in the graph has very small degree, then it is possible that in the $\mathbf{V}$ matrix its corresponding row are very close to zero and difficult to cluster. By normalizing each row in $\mathbf{V}$, it can be guaranteed that each row is not close to the origin.

There are many other discussions about spectral clustering. Chung [12] discussed many properties of unnormalized and normalized Laplacian matrices. Von Luxburg and others [81, 80] discussed the consistency of spectral clustering algorithms and compared the unnormalized spectral clustering with normalized spectral clustering. Kannan and others [43] examined the quality of spectral clustering while the algorithm are in polynomial time.

CHAPTER 3

Modularity Graph Partitioning

The data analysis method introduced in this paper is based on the modularity graph partitioning method introduced by Norman and Girvan in [63], and further explained by Newman in [62]. In this chapter we will define the modularity matrix and discuss how to use the modularity matrix to partition a graph. We will also discuss some other research about modularity partitioning.

3.1 The Modularity Matrix

We start with the definition of the modularity matrices.

Definition 3.1. *For a weighted, undirected graph, suppose its adjacency matrix* $\mathbf{A}$ *is* *given. Then the modularity matrix corresponding to the graph is defined by*

$$\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^T}{2m}, \tag{3.1}$$

where m is the number of edges in the graph and

$$\mathbf{d} = \begin{pmatrix} d_1 & d_2 & \cdots & d_n \end{pmatrix}^T \tag{3.2}$$

is the degree vector.

In Definition 3.1, the (i, j)-th entry in the rank-one matrix $(\mathbf{d}\mathbf{d}^T)/(2m)$ represents the expected number of edges between nodes i and j in a graph having random edges where the degrees of the nodes i and j are d_i and d_j, respectively. So the meaning of the matrix $\mathbf{B}$ is the comparison between the given graph and the graph that contains the expected number of edges. If two nodes should be put in the same cluster, then the number of edges between these nodes in the actual graph should be more than the expected number of edges, and the corresponding entry in the modularity matrix should be positive. On the other hand, if two nodes should be put in different clusters, then the number of edges between these nodes in the actual graph should be less than the expected number of edges, and the corresponding entry in the modularity matrix should be negative.
Here are some basic properties of the modularity matrix:

1. A modularity matrix is symmetric and all of its eigenvalues are real;

2. A modularity matrix has an eigenpair $(0, \mathbf{e})$;

3. A modularity matrix may have both positive and negative eigenvalues.

In the following sections we will discuss how to use the modularity matrix to partition a graph.

3.2 Bipartitioning a Graph

Suppose we are given a weighted, undirected graph, so its adjacency matrix **A** is known. Then for a particular bipartition (G_1, G_2) of the graph, we can express the partition with a vector **s** by letting

$$\mathbf{s}_i = \begin{cases} 1 & \text{if node } i \text{ in } G_1 \\ -1 & \text{otherwise.} \end{cases} \tag{3.3}$$

Then we can define the modularity corresponding the graph with the given bipartition.

Definition 3.2. *Given a graph with a bipartition* **s***, the modularity of the graph corresponding to the bipartition is defined by*

$$Q(\mathbf{s}) = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s}. \tag{3.4}$$

For a given graph, since its number of edges is constant, the definition of modularity can be also written as

$$Q(\mathbf{s}) = \mathbf{s}^T\mathbf{B}\mathbf{s}. \tag{3.5}$$

It can also be written in the summation form

$$Q(\mathbf{s}) = \sum_{1\le i,j\le n} s_i s_j \mathbf{B}_{ij}. \tag{3.6}$$

As mentioned above, the (i, j)-th entry in the matrix **B** represents the difference of the numbers of edges between nodes i and j in the actual graph and the graph that contains the expected number of edges. Therefore we may have four cases for the term $s_i s_j \mathbf{B}_{ij}$:

1. If s_i and s_j have the same sign, which means in the given partition the two nodes are grouped together, and at the same time $\mathbf{B}_{ij}$ is positive, which means the number of edges in the actual graph is more than expected, then the term $s_i s_j \mathbf{B}_{ij}$ will be positive.

2. If s_i and s_j have different signs, which means in the given partition the two nodes are grouped in different clusters, and at the same time $\mathbf{B}_{ij}$ is negative, which means the number of edges in the actual graph is less than expected, then the term $s_i s_j \mathbf{B}_{ij}$ will also be positive.

3. If s_i and s_j have the same sign, and $\mathbf{B}_{ij}$ is negative, then $s_i s_j \mathbf{B}_{ij}$ will be negative.

4. If s_i and s_j have different signs, and $\mathbf{B}_{ij}$ is positive, then $s_i s_j \mathbf{B}_{ij}$ will be negative.

From these four cases it can be seen that $Q(\mathbf{s})$ is positive when the signs of $\mathbf{B}_{ij}$ and $s_i s_j$ are the same, and negative when the signs are different. This property can be understood as the "correctness" of grouping nodes i and j together. To be more clear, $Q(\mathbf{s})$ increases when the partition make the "correct" decisions, and decreases when the "wrong" decisions are made. Therefore, the goal of modularity graph partitioning is to find such a vector $\mathbf{s}$ that subject to Eq. 3.3 to maximize $Q(\mathbf{s})$. However, the problem is NP-hard due to the special restriction on the vector $\mathbf{s}$. Newman [62] relaxed the problem by allowing the entries in $\mathbf{s}$ to be any arbitrary real values, and the graph will be partitioned based on the signs of the entries in the vector $\mathbf{s}$. It is easy to see that the dominant eigenvector of $\mathbf{B}$ can solve the relaxed problem.

Note that since zero is always an eigenvalue of $\mathbf{B}$, the largest eigenvalue cannot be negative. However, it is possible that the largest eigenvalue of the modularity matrix is zero. Newman [62] discussed this case. Since the eigenvector corresponding to the zero eigenvalue is the **e** vector, the largest eigenvalue is zero indicates that *all* vertices in the graph should be put into the same cluster. That means there is only one cluster in the graph. No partitioning should be done in this case.

To generate more than two subgraphs, the partitioning method discussed above wil be applied several times to build a hierarchy until the desired number of clusters k is reached. Each time the subgraph that can contribute more modularity while being partitioned. Next we give the algorithm for modularity clustering.

Algorithm 6 Modularity Clustering by Newman [62]

```
Require: An $n \times n$ adjacency or similarity matrix $\mathbf{A}$, number of clusters $k$.
  • repeat
        Compute the modularity matrix $\mathbf{B}$ with Eq. 3.1.
        Compute the largest eigenvalue $\lambda_1$ of $\mathbf{B}$.
        If $\lambda_1 = 0$ then return the whole data as one cluster.
        If $\lambda_1 > 0$ then compute its corresponding eigenvector $\mathbf{v}_1$.
        Cluster the entries in $\mathbf{v}_1$ by their signs to get clusters $C_1$ and $C_2$.
        Bipartition the graph to get subgraphs $G_1$ and $G_2$ such that $G_i = \{j | \mathbf{v}_{1j} \in C_i\}$.
        Pick the subgraph with larger $Q(\mathbf{s})$ in Eq. 3.4
  • until The number of desired number of clusters is reached or no positive eigen-
    values can be found for the modularity matrices of the subgraphs. Suppose in
    the latter case there are $k'$ subgraphs.
  return Clusters $G_1$, $G_2$, $\cdots$, $G_{k^*}$ where $k^* = \min\{k, k'\}$.
```

When the modularity clustering algorithm is applied on the toy example in Figure 2.3, the dominant eigenvector of the modularity matrix is

$$\mathbf{v}_1 = \begin{pmatrix} -0.1252 \\ -0.3479 \\ -0.3708 \\ -0.3708 \\ -0.3116 \\ 0.2276 \\ 0.2173 \\ 0.3991 \\ 0.3425 \\ 0.3398 \end{pmatrix},$$

which indicates that the first five vertices should be put into the one group and the last five vertices into the the other group.

Although the modularity clustering algorithm can do a good job for our toy example, there is a performance issue for it. If the desired number of clusters is three, then we need to apply the algorithm triple times because after the first bipartition, we need to calculate the $Q(\mathbf{s})$ value for each subgraph, and calculating $Q(\mathbf{s})$ is essentially equivalent to partitioning the subgraph because the dominant eigenvector has to be solved. If for each subgraph we have to compute the modularity matrix and compute its dominant eigenvector, the total work could be huge, and many calculations are wasted. A natural way to avoid this huge work is to modify the algorithm to be like Algorithm 3, that several eigenvectors of $\mathbf{B}$ together give all the clusters so we only have to form the modularity matrix once and compute the eigenvectors once. However, the question is why the eigenvectors other than the dominant eigenvector of the **B** matrix can give us the information of the clusters. Those eigenvectors may only contain irrelevant information or just noise in the data. If the eigenvectors contain useful information, then should we keep all of them? If there is an order of "importance" for the eigenvectors, then how the order is defined and how to order the eigenvectors? These questions need to be answered before just taking several eigenvectors and partition the graph.

3.3 Other Discussions about Modularity Clustering

The questions raised in the last section will be answered in the following chapters. Before diving into the discussion we first look at some other discussions about modularity clustering in the literature. The modularity partitioning algorithm has been widely applied and discussed since proposed by Newman and Girvan [63]. For instance, it has been applied to reveal human brain functional networks [55] and ecological networks [26], and used in image processing [54]. Blondel et al. [6] proposed a heuristic that can reveal the community structure for large networks. Rotta and Noack [70] compared several heuristics in maximizing modularity. DasGupta and Desai [13] studied the complexity of modularity clustering. The limitations of the modularity maximization technique are discussed in [29] and [46]. Bolla [8] normalized the modularity matrix to form

$$\mathbf{B}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{B}\mathbf{D}^{-\frac{1}{2}}, \tag{3.7}$$

just like the way $\mathbf{L}_{sym}$ is formed in Eq. 2.34. Zhang and Zhao [89] normalized the modularity matrix in another way to form

$$\mathbf{B}_{rw} = \mathbf{D}^{-1}\mathbf{B}, \tag{3.8}$$

as the way $\mathbf{L}_{rw}$ is formed in Eq. 2.35. To make the discussion more complete, here we write down the normalized modularity clustering algorithms:

```
Algorithm 7 Normalized Modularity Clustering by Bolla [8]
Require: An $n \times n$ adjacency or similarity matrix $\mathbf{A}$, number of clusters $k$.
    • Compute the modularity matrix $\mathbf{B}$ with Eq. 3.1.
    • Compute the normalized modularity matrix $\mathbf{B}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{B}\mathbf{D}^{-\frac{1}{2}}$.
    • Compute the largest $k$ eigenvalues of $\mathbf{B}_{sym}$ and their corresponding eigenvectors.
    • Form the matrix $\mathbf{V}$ with the $k$ eigenvectors of $\mathbf{B}_{sym}$ as columns.
    • Let $\mathbf{p}_i$ be the $i$-th row of $\mathbf{V}$.
    • Cluster the $\mathbf{p}_i$'s with $k$-means into clusters $C_1, C_2, \cdots, C_k$.
  return Clusters $G_1, G_2, \cdots, G_k$ such that $G_i = \{j|\mathbf{p}_j \in C_i\}$.
```

```
Algorithm 8 Normalized Modularity Clustering by Zhang and Zhao [89]
Require: An $n \times n$ adjacency or similarity matrix $\mathbf{A}$, number of clusters $k$.
    • Compute the modularity matrix $\mathbf{B}$ with Eq. 3.1.
    • Compute the Normalized Laplacian matrix $\mathbf{B}_{rw} = \mathbf{D}^{-1}\mathbf{B}$.
    • Compute the smallest $k$ eigenvalues of $\mathbf{B}_{rw}$ and their corresponding eigenvectors.
    • Form the matrix $\mathbf{V}$ with the $k$ eigenvectors of $\mathbf{B}_{rw}$ as columns.
    • Let $\mathbf{p}_i$ be the $i$-th row of $\mathbf{V}$.
    • Cluster the $\mathbf{p}_i$'s with $k$-means into clusters $C_1, C_2, \cdots, C_k$.
  return Clusters $G_1, G_2, \cdots, G_k$ such that $G_i = \{j|\mathbf{p}_j \in C_i\}$.
```

Jiang and Meyer [39] proved that although the unnormalized spectral clustering and modularity clustering may give different results, the Fiedler vector of $\mathbf{L}_{sym}$ and the dominant eigenvector of $\mathbf{B}_{sym}$ are exactly the same, and consequently they will give same bipartition results.

CHAPTER 4

Definition of Modularity Components

In this section we will define the modularity components and will discuss their properties in the next section. The research in these chapters is mainly based on [38]. In this chapter we will first introduce some properties of the diagonal plus rank one (DPR1) matrices and use the properties to prove some lemmas about the relation between the eigenvectors of a particular kind of similarity matrices that can be fed in the modularity algorithm and the singular vectors of the *uncentered* data matrix. The lemmas will help us to define the modularity components. In this and the next chapters, we will assume that $\mathbf{X}_{p\times n}$ is the uncentered and normalized data matrix.

4.1 Eigenvalues and Eigenvectors of DPR1 Matrices

Suppose the SVD of the uncentered data matrix $\mathbf{X}$ is $\mathbf{X} = \mathbf{U\Sigma V}^T$ and there are k nonzero singular values. Then the similarity matrix

$$\mathbf{A} = \mathbf{X}^T\mathbf{X} = \mathbf{V\Sigma}^T\mathbf{\Sigma V}^T \tag{4.1}$$

has k positive eigenvalues. From the interlacing theorem mentioned in [9] and [85], it is guaranteed that the largest $k-1$ eigenvalues of the modularity matrix

$$\mathbf{B} = \mathbf{A} - \frac{\mathbf{dd}^T}{2m} = \mathbf{X}^T\mathbf{X} - \frac{\mathbf{dd}^T}{2m} \tag{4.2}$$

are positive. If the k dominant eigenvalues of $\mathbf{A}$ are simple, then the eigenvectors of $\mathbf{B}$ corresponding to the largest $k-1$ eigenvalues can be written as linear combinations of the eigenvectors of $\mathbf{A}$. The proof of the lemma is based on a theorem from [9] about the interlacing property of a diagonal matrix and its rank-one modification and how to calculate the eigenvectors of a DPR1 matrix [56]. The theorem can also be found in [85].

Theorem 4.1. *Let $\mathbf{C} = \mathbf{D} + \rho\mathbf{vv}^T$, where $\mathbf{D}$ is diagonal, $\|\mathbf{v}\|_2 = 1$. Let $d_1 \le d_2 \le \cdots \le d_n$ be the eigenvalues of $\mathbf{D}$, and let $\tilde{d}_1 \le \tilde{d}_2 \le \cdots \le \tilde{d}_n$ be the eigenvalues of $\mathbf{C}$. Then $\tilde{d}_1 \le d_1 \le \tilde{d}_2 \le d_2 \le \cdots \le \tilde{d}_n \le d_n$ if $\rho < 0$. If the d_i are distinct and all the elements of $\mathbf{v}$ are nonzero, then the eigenvalues of $\mathbf{C}$ strictly separate those of $\mathbf{D}$.*

Corollary 4.2. *With the notations in Theorem 4.1, the eigenvector of $\mathbf{C}$ corresponding to the eigenvalue $\tilde{d}_i$ is given by*

$$(\mathbf{D} - \tilde{d}_i\mathbf{I})^{-1}\mathbf{v}. \tag{4.3}$$

Theorem 4.1 tells us that the eigenvalues of a DPR1 matrix are interlaced with the eigenvalues of the original diagonal matrix. In the next section we will use the theorems about the DPR1 matrices to state and prove the lemma that the $k-1$ dominant eigenvectors of $\mathbf{B}$ can be written as a linear combination of the eigenvectors of $\mathbf{A}$.

4.2 Dominant Eigenvectors of Modularity Matrices

In this section we will develop the relation between the dominant eigenvectors of the modularity matrices and the eigenvectors of $\mathbf{X}^T\mathbf{X}$, or the singular vectors of $\mathbf{X}$. This relation will help us to define the modularity components.

Lemma 4.3. *Suppose the largest $k-1$ eigenvalues of $\mathbf{B}$ are $\beta_1 > \beta_2 > \cdots > \beta_{k-1}$ and the nonzero eigenvalues of $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ are $\alpha_1 > \alpha_2 > \cdots > \alpha_k$. Further suppose that for $1 \le i \le k-1$ we have $\beta_i \ne \alpha_i$ and $\beta_i \ne \alpha_{i+1}$. Then the eigenvector $\mathbf{b}_i$ of $\mathbf{B}$ can be written by*

$$\mathbf{b}_i = \sum_{j=1}^{k} \gamma_{ij}\mathbf{v}_j, \tag{4.4}$$

where

$$\gamma_{ij} = \frac{\mathbf{v}_j^T\mathbf{d}}{(\alpha_j - \beta_i)\|\mathbf{d}\|_2}. \tag{4.5}$$

Proof. If $\mathbf{A} = \mathbf{X}^T\mathbf{X}$, and if the SVD of $\mathbf{X}$ is $\mathbf{X} = \mathbf{U\Sigma V}^T$, then

$$\mathbf{A} = \mathbf{V\Sigma^T\Sigma V}^T = \mathbf{V\Sigma_A V}^T, \tag{4.6}$$

where $\mathbf{\Sigma_A}$ is an $n \times n$ diagonal matrix. Suppose the rows and columns of $\mathbf{A}$ are ordered such that $\mathbf{\Sigma_A} = \mathrm{diag}(\alpha_1, \alpha_2, \cdots, \alpha_n)$, where $\alpha_1 > \alpha_2 > \cdots > \alpha_k > \alpha_{k+1} = \cdots = \alpha_n = 0$. Let $\mathbf{V} = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{pmatrix}$. Similarly, since $\mathbf{B}$ is symmetric, it is orthogonally similar to a diagonal matrix. Suppose the eigenvalues of $\mathbf{B}$ are $\beta_1, \beta_2, \cdots, \beta_n$ with largest $k-1$

eigenvalues $\beta_1 > \beta_2 > \cdots > \beta_{k-1}$.

Since $\mathbf{B} = \mathbf{A} - \mathbf{d}\mathbf{d}^T/(2m)$, we have

$$\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^T}{2m} = \mathbf{V}\boldsymbol{\Sigma}_{\mathbf{A}}\mathbf{V}^T - \frac{\mathbf{d}\mathbf{d}^T}{2m} = \mathbf{V}(\boldsymbol{\Sigma}_{\mathbf{A}} + \rho\mathbf{y}\mathbf{y}^T)\mathbf{V}^T, \tag{4.7}$$

where $\mathbf{y} = \mathbf{V}^T\mathbf{d}/\|\mathbf{V}^T\mathbf{d}\|_2$ and $\rho = -\|\mathbf{V}^T\mathbf{d}\|_2^2/(2m)$. Since $\boldsymbol{\Sigma}_{\mathbf{A}} + \rho\mathbf{y}\mathbf{y}^T$ is also symmetric, it is orthogonally similar to a diagonal matrix. So we have

$$\mathbf{B} = \mathbf{V}\mathbf{U}'\boldsymbol{\Sigma}_{\mathbf{B}}\mathbf{U}'^T\mathbf{V}^T, \tag{4.8}$$

where $\mathbf{U}'$ is orthogonal and $\boldsymbol{\Sigma}_{\mathbf{B}}$ is diagonal. Since $\boldsymbol{\Sigma}_{\mathbf{A}} + \rho\mathbf{y}\mathbf{y}^T$ is a DPR1 matrix, $\rho < 0$ and $\|\mathbf{y}\|_2 = 1$, the interlacing theorem applies to the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$. More specifically, we have

$$\alpha_k < \beta_{k-1} < \alpha_{k-1} < \beta_{k-2} < \cdots < \beta_2 < \alpha_2 < \beta_1 < \alpha_1.$$

The strict inequalities hold because of our assumptions. Let $\mathbf{B}_1 = \boldsymbol{\Sigma}_{\mathbf{A}} + \rho\mathbf{y}\mathbf{y}^T$. Since $\mathbf{B} = \mathbf{V}\mathbf{B}_1\mathbf{V}^T$, we have $\mathbf{B}\mathbf{V} = \mathbf{V}\mathbf{B}_1$. Suppose $(\lambda, \mathbf{u})$ is an eigenpair of $\mathbf{B}_1$, then

$$\mathbf{B}\mathbf{V}\mathbf{u} = \mathbf{V}\mathbf{B}_1\mathbf{u} = \lambda\mathbf{V}\mathbf{u} \tag{4.9}$$

implies that $(\lambda, \mathbf{u})$ is an eigenpair of $\mathbf{B}_1$ if and only if $(\lambda, \mathbf{V}\mathbf{u})$ is an eigenpair of $\mathbf{B}$. By Corollary 4.2, the eigenvector of $\mathbf{B}_1$ corresponding to β_i, $1 \le i \le k-1$ is given by

$$\mathbf{p}_i = (\boldsymbol{\Sigma}_{\mathbf{A}} - \beta_i\mathbf{I})^{-1}\mathbf{y} = (\boldsymbol{\Sigma}_{\mathbf{A}} - \beta_i\mathbf{I})^{-1}\frac{\mathbf{V}^T\mathbf{d}}{\|\mathbf{V}^T\mathbf{d}\|_2}, \tag{4.10}$$

and hence the eigenvector of $\mathbf{B}$ corresponding to β_i, $1 \le i \le k-1$ is given by

$$\mathbf{b}_i = \mathbf{V}\mathbf{p}_i = \mathbf{V}(\mathbf{\Sigma_A} - \beta_i \mathbf{I})^{-1} \frac{\mathbf{V}^T\mathbf{d}}{\|\mathbf{V}^T\mathbf{d}\|_2} = \frac{1}{\|\mathbf{d}\|_2} \sum_{j=1}^{n} \frac{\mathbf{v}_j^T\mathbf{d}}{\alpha_j - \beta_i}\mathbf{v}_j. \tag{4.11}$$

Since $\mathbf{d} = \mathbf{A}\mathbf{e} = \mathbf{V}\mathbf{\Sigma_A}\mathbf{V}^T\mathbf{e}$ where $\mathbf{e}$ is a column vector with all ones, we have

$$\mathbf{v}_j^T\mathbf{d} = \mathbf{v}_j^T\mathbf{V}\mathbf{\Sigma_A}\mathbf{V}^T\mathbf{e} = \mathbf{e}_j^T\mathbf{\Sigma_A}\mathbf{V}^T\mathbf{e}. \tag{4.12}$$

Since $\text{rank}(\mathbf{A}) = k$, we have $\mathbf{v}_j^T\mathbf{d} = 0$ for $j > k$. Therefore, the eigenvector of $\mathbf{B}$ corresponding to β_i, $1 \le i \le k-1$ is given by

$$\mathbf{b}_i = \sum_{j=1}^{k} \gamma_{ij}\mathbf{v}_j, \tag{4.13}$$

where

$$\gamma_{ij} = \frac{\mathbf{v}_j^T\mathbf{d}}{(\alpha_j - \beta_i)\|\mathbf{d}\|_2}. \tag{4.14}$$

□

The point of Lemma 4.3 is to realize that the vector $\mathbf{b}_i$ is a linear combination of the $\mathbf{v}_i$. The next lemma gives the linear expression of the vectors $\mathbf{b}_i^T\mathbf{X}^\dagger$ in terms of the $\mathbf{u}_i$, where $\mathbf{X}^\dagger$ is the Moore-Penrose inverse of $\mathbf{X}$.

Lemma 4.4. *With the assumptions in Lemma 4.3, we have*

$$\mathbf{b}_i^T\mathbf{X}^\dagger = \sum_{j=1}^{k} \frac{\gamma_{ij}}{\sigma_j}\mathbf{u}_j^T, \tag{4.15}$$

where σ_j is the j-th the nonzero singular value of $\mathbf{X}$.

Proof.

$$\mathbf{b}_i^T \mathbf{X}^\dagger = \left(\sum_{j=1}^{k} \gamma_{ij} \mathbf{v}_j^T \right) \mathbf{V} \mathbf{\Sigma}^\dagger \mathbf{U}^T$$

$$= \begin{pmatrix} \gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ik} & 0 & \cdots & 0 \end{pmatrix}_{1\times n} \mathbf{\Sigma}^\dagger \mathbf{U}^T$$

$$= \begin{pmatrix} \frac{\gamma_{i1}}{\sigma_1} & \frac{\gamma_{i2}}{\sigma_2} & \cdots & \frac{\gamma_{ik}}{\sigma_k} & 0 & \cdots & 0 \end{pmatrix}_{1\times p} \mathbf{U}^T$$

$$= \sum_{j=1}^{k} \frac{\gamma_{ij}}{\sigma_j} \mathbf{u}_j^T.$$

□

Lemma 4.4 shows that if $\mathbf{b}_i$ can be written as a linear combination of the $\mathbf{v}_j$, then the vectors $\mathbf{b}_i^T \mathbf{X}^\dagger$ can be written as a linear combination of the $\mathbf{u}_i$. In the next section we give the formal definition of the modularity components.

4.3 Definition of Modularity Components

Based on Lemma 4.3 and Lemma 4.4, we may define a set of vectors that we will call modularity components. We will prove in the next chapter that the modularity components have some properties that are analogous to the ones of principal components.

Definition 4.5. *Suppose $\mathbf{X}_{p\times n}$ is the data matrix, $\mathbf{b}_i$ is the eigenvector corresponding to the i-th largest eigenvalue of $\mathbf{B}$, where*

$$\mathbf{B} = \mathbf{X}^T \mathbf{X} - \frac{\mathbf{d}\mathbf{d}^T}{2m}. \tag{4.16}$$

Under the assumptions in Lemma 4.3, let

$$\mathbf{m}_i^T = \mathbf{b}_i^T \mathbf{X}^\dagger = \sum_{j=1}^{k} \frac{\gamma_{ij}}{\sigma_j} \mathbf{u}_j^T. \tag{4.17}$$

The i-th modularity component is defined to be

$$\mathbf{c}_i = \frac{\mathbf{m}_i}{\|\mathbf{m}_i\|_2}. \tag{4.18}$$

By Lemma 4.3 and Lemma 4.4 it can be seen that as long as the assumptions in Lemma 4.3 are met, the modularity components are well-defined, and the definition of $\mathbf{c}_i$ is based on the linear combination of $\mathbf{b}_i^T \mathbf{X}^\dagger$ in terms of the $\mathbf{u}_i$. In the next chapter some important properties of the modularity components are established.

CHAPTER 5

Properties of Modularity Components

In this chapter we will use the definition of modularity components to prove some important properties of modularity components. The properties of the modularity components will be given, and it will be explained why these properties can help in data clustering and dimension reduction. We will also compare the properties of modularity components with the ones of principal components discussed in Chapter 2. It can be seen that the properties of modularity components are quite similar to some of the properties of principal components.

5.1 Orthogonality of Modularity Components

Now we state and prove the first important property of modularity components. We will prove that if the assumptions in Lemma 4.3 are met, then the modularity components are orthogonal to each other.

Theorem 5.1. *With the assumptions in Lemma 4.3, suppose* $\mathbf{X}_{p\times n}$ *is the unnormalized data matrix,* $\mathbf{A} = \mathbf{X}^T\mathbf{X}$*, and* $\mathbf{B} = \mathbf{A} - \mathbf{d}\mathbf{d}^T/(2m)$*. Suppose* $\mathbf{b}_i$*,* $\mathbf{b}_j$ *are the eigenvectors of* $\mathbf{B}$ *corresponding to eigenvalues* λ_i *and* λ_j*,* $1 \leq i, j \leq k-1$*, respectively. Then we have*

$$\mathbf{B} = (\mathbf{B}\mathbf{X}^\dagger)(\mathbf{B}\mathbf{X}^\dagger)^T \tag{5.1}$$

and

$$\mathbf{c}_i \perp \mathbf{c}_j \tag{5.2}$$

for $i \neq j$*.*

Proof. It is sufficient to prove that $\mathbf{m}_i \perp \mathbf{m}_j$ for $i \neq j$. From $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ we have

$$\mathbf{d} = \mathbf{A}\mathbf{e} = \mathbf{X}^T\mathbf{X}\mathbf{e}, \tag{5.3}$$

and

$$2m = \mathbf{d}^T\mathbf{e} = \mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}. \tag{5.4}$$

Therefore,

$$\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^T}{2m} = \mathbf{X}^T\mathbf{X} - \frac{(\mathbf{X}^T\mathbf{X}\mathbf{e})(\mathbf{X}^T\mathbf{X}\mathbf{e})^T}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}} = \mathbf{X}^T\mathbf{X} - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T\mathbf{X}}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}}. \tag{5.5}$$

Since $\mathbf{X}^T\mathbf{X}\mathbf{X}^\dagger = \mathbf{X}^T$ is always true, we have

$$\mathbf{B}\mathbf{X}^\dagger = \left(\mathbf{X}^T\mathbf{X} - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T\mathbf{X}}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}}\right)\mathbf{X}^\dagger$$

$$= \mathbf{X}^T\mathbf{X}\mathbf{X}^\dagger - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{X}^\dagger}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}} = \mathbf{X}^T - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}}. \tag{5.6}$$

Consequently,

$$(\mathbf{B}\mathbf{X}^\dagger)(\mathbf{B}\mathbf{X}^\dagger)^T = \left(\mathbf{X}^T - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}}\right)\left(\mathbf{X} - \frac{\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T\mathbf{X}}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}}\right)$$

$$= \mathbf{X}^T\mathbf{X} - \frac{2\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T\mathbf{X}}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}} + \frac{(\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e})\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T\mathbf{X}}{(\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e})^2}$$

$$= \mathbf{X}^T\mathbf{X} - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T\mathbf{X}^T\mathbf{X}}{\mathbf{e}^T\mathbf{X}^T\mathbf{X}\mathbf{e}}. \tag{5.7}$$

Therefore $\mathbf{B} = (\mathbf{B}\mathbf{X}^\dagger)(\mathbf{B}\mathbf{X}^\dagger)^T$. Since $\mathbf{B}\mathbf{b}_i = \lambda_i\mathbf{b}_i$, $\mathbf{B}\mathbf{b}_j = \lambda_j\mathbf{b}_j$, $\lambda_i \neq 0$, $\lambda_j \neq 0$, we have

$$\mathbf{m}_i^T\mathbf{m}_j = (\mathbf{b}_i^T\mathbf{X}^\dagger)(\mathbf{b}_j^T\mathbf{X}^\dagger)^T = \left(\frac{1}{\lambda_i}\mathbf{b}_i^T\mathbf{B}\mathbf{X}^\dagger\right)\left(\frac{1}{\lambda_j}\mathbf{b}_j^T\mathbf{B}\mathbf{X}^\dagger\right)^T$$

$$= \frac{1}{\lambda_i\lambda_j}\mathbf{b}_i^T(\mathbf{B}\mathbf{X}^\dagger)(\mathbf{B}\mathbf{X}^\dagger)^T\mathbf{b}_j = \frac{1}{\lambda_i\lambda_j}\mathbf{b}_i^T\mathbf{B}\mathbf{b}_j = \frac{1}{\lambda_i}\mathbf{b}_i^T\mathbf{b}_j = 0, \tag{5.8}$$

so

$$\mathbf{c}_i^T\mathbf{c}_j = \frac{\mathbf{m}_i^T\mathbf{m}_j}{\|\mathbf{m}_i\|_2\|\mathbf{m}_j\|_2} \tag{5.9}$$

implies $\mathbf{c}_i \perp \mathbf{c}_j$ for $i \neq j$. $\square$

From Theorem 5.1, it can be seen that the modularity components are orthogonal to each other. Therefore, like the principal components, the modularity components are also a set to orthonormal vectors.

5.2 Projection of Data onto the Span of Modularity Components

In this section we will prove that the projection of the uncentered data onto the span of $\mathbf{c}_i$ is a scalar multiple of $\mathbf{b}_i$.

Theorem 5.2. *With the assumptions in Lemma 4.3, let $\mathbf{P}_{\mathbf{c}_i}$ be the projector onto the span of $\mathbf{c}_i$. Then*

$$\mathbf{P}_{\mathbf{c}_i}\mathbf{X} = \frac{1}{\|\mathbf{m}_i\|_2}\mathbf{c}_i\mathbf{b}_i^T. \tag{5.10}$$

Proof.

$$\mathbf{P}_{\mathbf{c}_i}\mathbf{X} = \mathbf{c}_i\mathbf{c}_i^T\mathbf{X} = \frac{1}{\|\mathbf{m}_i\|_2}\mathbf{c}_i\mathbf{m}_i^T\mathbf{U\Sigma V}^T = \frac{1}{\|\mathbf{m}_i\|_2}\mathbf{c}_i\left(\sum_{j=1}^{k}\frac{\gamma_{ij}}{\sigma_j}\mathbf{u}_j^T\right)\mathbf{U\Sigma V}^T$$

$$= \frac{1}{\|\mathbf{m}_i\|_2}\mathbf{c}_i\begin{pmatrix}\frac{\gamma_{i1}}{\sigma_1} & \frac{\gamma_{i2}}{\sigma_2} & \cdots & \frac{\gamma_{ik}}{\sigma_k} & 0 & \cdots & 0\end{pmatrix}_{1\times p}\mathbf{\Sigma V}^T$$

$$= \frac{1}{\|\mathbf{m}_i\|_2}\mathbf{c}_i\begin{pmatrix}\gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ik} & 0 & \cdots & 0\end{pmatrix}_{1\times n}\mathbf{V}^T$$

$$= \frac{1}{\|\mathbf{m}_i\|_2}\mathbf{c}_i\sum_{j=1}^{k}\gamma_{ij}\mathbf{v}_i^T = \frac{1}{\|\mathbf{m}_i\|_2}\mathbf{c}_i\mathbf{b}_i^T.$$

□

The property given by Theorem 5.2 is very similar to the principal component analysis in the sense that if we project the data onto the span of the components, we get a scalar multiple of a vector, and the vector can give the cluster structure of the data based on the signs of the entries in the eigenvectors.

5.3 Meaning of Eigenvalues of B and the Corresponding Modularity Components

In this section we will prove that the first modularity component has the largest modularity in the data, and each succeeding modularity component has the largest modularity with the constraint that it is orthogonal to all previous modularity components. We will also how the eigenvalues of the modularity matrix defines the “importance” of each modularity component.

Theorem 5.3. *With the assumptions in Lemma 4.3, we have*

$$\beta_i = \frac{1}{\|\mathbf{m}_i\|_2^2}, \tag{5.11}$$

for $1 \leq i \leq k-1$*. Moreover, let* $\mathbf{X}_1 = \mathbf{X}$ *and for* $1 < i \leq k-1$*,*

$$\mathbf{X}_i = \mathbf{X} - \sum_{j=1}^{i-1} \mathbf{c}_j \mathbf{c}_j^T \mathbf{X} \tag{5.12}$$

and $\mathbf{d}_i$*,* m_i *defined correspondingly, then* β_i *is the largest eigenvalue of*

$$\mathbf{B}_i = \mathbf{X}_i^T \mathbf{X}_i - \frac{\mathbf{d}_i \mathbf{d}_i^T}{2m_i}, \tag{5.13}$$

and $(\beta_i, \mathbf{b}_i)$ *is an eigenpair for both* $\mathbf{B}$ *and* $\mathbf{B}_i$*. Moreover, we have*

$$\mathbf{b}_i^T \mathbf{X}_i^\dagger = \mathbf{b}_i^T \mathbf{X}^\dagger. \tag{5.14}$$

Proof. For $i = 2$, since it is proved in [62] that $\mathbf{b}_1$ is the vector $\bar{\mathbf{s}}$ that maximizes Q in

Eq. 3.5, we have

$$Q_{max1} = \mathbf{b}_1^T \mathbf{B} \mathbf{b}_1 = \beta_1 \mathbf{b}_1^T \mathbf{b}_1 = \beta_1. \tag{5.15}$$

By Theorem 5.1,

$$\max_{\mathbf{s}} \mathbf{s}^T \mathbf{B} \mathbf{s} = \max_{\mathbf{s}} \mathbf{s}^T (\mathbf{B}\mathbf{X}^\dagger)(\mathbf{B}\mathbf{X}^\dagger)^T \mathbf{s} = \max_{\mathbf{s}} \|(\mathbf{B}\mathbf{X}^\dagger)^T \mathbf{s}\|_2^2 = \max_{\mathbf{s}} \|(\mathbf{X}^\dagger)^T \mathbf{B}\mathbf{s}\|_2^2$$

$$= \|(\mathbf{X}^\dagger)^T \mathbf{B}\mathbf{b}_1\|_2^2 = \|(\mathbf{X}^\dagger)^T \beta_1 \mathbf{b}_1\|_2^2 = \|\beta_1 \mathbf{m}_1\|_2^2 = \beta_1. \tag{5.16}$$

Therefore

$$\beta_1 = \frac{1}{\|\mathbf{m}_1\|_2^2}. \tag{5.17}$$

Then $\mathbf{X}_2$ is defined by

$$\mathbf{X}_2 = \mathbf{X} - \mathbf{c}_1 \mathbf{c}_1^T \mathbf{X} = (\mathbf{I} - \mathbf{c}_1 \mathbf{c}_1^T)\mathbf{X}. \tag{5.18}$$

Since $\mathbf{I} - \mathbf{c}_1 \mathbf{c}_1^T$ is idempotent, we have

$$\mathbf{X}_2^T \mathbf{X}_2 = \mathbf{X}^T (\mathbf{I} - \mathbf{c}_1 \mathbf{c}_1^T)\mathbf{X} = \mathbf{X}^T \mathbf{X} - \mathbf{X}^T \mathbf{c}_1 \mathbf{c}_1^T \mathbf{X}. \tag{5.19}$$

By Theorem 5.2, we know that $\mathbf{c}_1 \mathbf{c}_1^T \mathbf{X} = \mathbf{c}_1 \mathbf{b}_1^T / \|\mathbf{m}_1\|_2$, so $\mathbf{c}_1^T \mathbf{X} = \sqrt{\beta_1} \mathbf{b}_1^T$ and then

$$\mathbf{X}_2^T \mathbf{X}_2 = \mathbf{X}^T \mathbf{X} - \beta_1 \mathbf{b}_1 \mathbf{b}_1^T. \tag{5.20}$$

Recall that

$$\mathbf{B}_2 = \mathbf{X}_2^T \mathbf{X}_2 - \frac{\mathbf{d}_2 \mathbf{d}_2^T}{2m_2} = \mathbf{X}_2^T \mathbf{X}_2 - \frac{\mathbf{X}_2^T \mathbf{X}_2 \mathbf{e}\mathbf{e}^T \mathbf{X}_2^T \mathbf{X}_2}{\mathbf{e}^T \mathbf{X}_2^T \mathbf{X}_2 \mathbf{e}}, \tag{5.21}$$

and in this use Eq. 5.20 together with $\mathbf{b}_1^T \mathbf{e} = 0$ (because $\mathbf{b}_1$ and $\mathbf{e}$ are eigenvectors corresponding to different eigenvalues of $\mathbf{B}$) to obtain

$$\mathbf{B}_2 = \mathbf{B} - \beta_1 \mathbf{b}_1 \mathbf{b}_1^T. \tag{5.22}$$

So by Brauer's theorem [56](Exercise 7.1.17), the eigenpairs of $\mathbf{B}_2$ are the same as those of $\mathbf{B}_1$ with β_1 replaced by zero. So β_2 is the largest eigenvalue of $\mathbf{B}_2$ and $\mathbf{b}_2$ is the eigenvector of $\mathbf{B}_2$ corresponding to β_2. Therefore $(\beta_2, \mathbf{b}_2)$ is an eigenpair for both $\mathbf{B}$ and $\mathbf{B}_2$.
To prove

$$\mathbf{b}_2^T \mathbf{X}_2^\dagger = \mathbf{b}_2^T \mathbf{X}^\dagger, \tag{5.23}$$

notice that

$$\begin{aligned}
\mathbf{b}_2^T \mathbf{X}_2^\dagger = \frac{1}{\lambda_2}\mathbf{b}_2^T \mathbf{B}_2 \mathbf{X}_2^\dagger &= \frac{1}{\lambda_2}\mathbf{b}_2^T \left(\mathbf{X}_2^T - \frac{\mathbf{X}_2^T \mathbf{X}_2 \mathbf{e}\mathbf{e}^T \mathbf{X}_2^T}{\mathbf{e}^T \mathbf{X}_2^T \mathbf{X}_2 \mathbf{e}} \right) \text{ by Eq. 5.6} \\
&= \frac{1}{\lambda_2}\mathbf{b}_2^T \left(\mathbf{X}_2^T - \frac{(\mathbf{X}^T\mathbf{X} - \beta_1 \mathbf{b}_1 \mathbf{b}_1^T)\mathbf{e}\mathbf{e}^T \mathbf{X}_2^T}{\mathbf{e}^T (\mathbf{X}^T\mathbf{X} - \beta_1 \mathbf{b}_1 \mathbf{b}_1^T)\mathbf{e}} \right) \text{ by Eq. 5.20} \\
&= \frac{1}{\lambda_2}\mathbf{b}_2^T \left(\mathbf{X}_2^T - \frac{\mathbf{X}^T \mathbf{X} \mathbf{e}\mathbf{e}^T \mathbf{X}_2^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X} \mathbf{e}} \right) \\
&= \frac{1}{\lambda_2}\mathbf{b}_2^T \left(\mathbf{I} - \frac{\mathbf{X}^T \mathbf{X} \mathbf{e}\mathbf{e}^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X} \mathbf{e}} \right) \left(\mathbf{X} - \mathbf{c}_1 \mathbf{c}_1^T \mathbf{X} \right)^T \text{ by Eq. 5.18} \\
&= \frac{1}{\lambda_2}\mathbf{b}_2^T \left(\mathbf{I} - \frac{\mathbf{X}^T \mathbf{X} \mathbf{e}\mathbf{e}^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X} \mathbf{e}} \right) \mathbf{X}^T - \frac{1}{\lambda_2}\mathbf{b}_2^T \left(\mathbf{I} - \frac{\mathbf{X}^T \mathbf{X} \mathbf{e}\mathbf{e}^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X} \mathbf{e}} \right) \mathbf{X}^T \mathbf{c}_1 \mathbf{c}_1^T \\
&= \mathbf{m}_2^T - \mathbf{m}_2^T \mathbf{c}_1 \mathbf{c}_1^T \text{ by Eq. 4.17 .}
\end{aligned} \tag{5.24}$$

Since $\mathbf{m}_2$ is on the span of $\mathbf{c}_2$, we have $\mathbf{m}_2^T \mathbf{c}_1 \mathbf{c}_1^T = 0$. Therefore

$$\mathbf{b}_2^T \mathbf{X}_2^\dagger = \mathbf{m}_2^T = \mathbf{b}_2^T \mathbf{X}^\dagger. \tag{5.25}$$

For the case when $2 < i \leq k-1$, let

$$Q_{i-1} = \mathbf{s}^T \mathbf{B}_{i-1} \mathbf{s}. \tag{5.26}$$

Notice that $\mathbf{b}_{i-1}$ is the vector $\mathbf{s}$ that maximizes Q_{i-1}. Then by similar steps we can prove that

$$\beta_{i-1} = \frac{1}{\|\mathbf{m}_{i-1}\|_2^2}. \tag{5.27}$$

Then $\mathbf{X}_i$ can be defined by

$$\mathbf{X}_i = \mathbf{X} - \sum_{j=1}^{i-1} \mathbf{c}_j \mathbf{c}_j^T \mathbf{X} = (\mathbf{I} - \sum_{j=1}^{i-1} \mathbf{c}_j \mathbf{c}_j^T)\mathbf{X}. \tag{5.28}$$

It is easy to prove that $\sum_{j=1}^{i-1} \mathbf{c}_j \mathbf{c}_j^T$ is idempotent. Then we have

$$\mathbf{X}_i^T \mathbf{X}_i = \mathbf{X}^T (\mathbf{I} - \sum_{j=1}^{i-1} \mathbf{c}_j \mathbf{c}_j^T)\mathbf{X}$$

$$= \mathbf{X}^T \mathbf{X} - \mathbf{X}^T (\sum_{j=1}^{i-1} \mathbf{c}_j \mathbf{c}_j^T)\mathbf{X} = \mathbf{X}^T \mathbf{X} - \sum_{j=1}^{i-1} \beta_j \mathbf{b}_j \mathbf{b}_j^T. \tag{5.29}$$

Recall that

$$\mathbf{B}_i = \mathbf{X}_i^T \mathbf{X}_i - \frac{\mathbf{d}_i \mathbf{d}_i^T}{2m_i} = \mathbf{X}_i^T \mathbf{X}_i - \frac{\mathbf{X}_i^T \mathbf{X}_i \mathbf{e} \mathbf{e}^T \mathbf{X}_i^T \mathbf{X}_i}{\mathbf{e}^T \mathbf{X}_i^T \mathbf{X}_i \mathbf{e}}, \tag{5.30}$$

and in this use Eq. 5.29 together with $\mathbf{b}_j^T \mathbf{e} = 0$ (because $\mathbf{b}_j$ and $\mathbf{e}$ are eigenvectors corresponding to different eigenvalues of $\mathbf{B}$) to obtain

$$\mathbf{B}_i = \mathbf{B} - \sum_{j=1}^{i-1} \beta_j \mathbf{b}_j \mathbf{b}_j^T = \mathbf{B}_{i-1} - \beta_{i-1} \mathbf{b}_{i-1} \mathbf{b}_{i-1}^T. \tag{5.31}$$

So by Brauer's theorem again, the eigenpairs of $\mathbf{B}_i$ are the same as those of $\mathbf{B}_{i-1}$ with β_{i-1} replaced by zero. So β_i is the largest eigenvalue of $\mathbf{B}_i$ and $\mathbf{b}_i$ is the eigenvector of $\mathbf{B}_i$ corresponding to β_i. Therefore $(\beta_i, \mathbf{b}_i)$ is an eigenpair for both $\mathbf{B}$ and $\mathbf{B}_i$.

To prove

$$\mathbf{b}_i^T \mathbf{X}_i^\dagger = \mathbf{b}_i^T \mathbf{X}^\dagger \tag{5.32}$$

for $2 < i \leq k-1$, notice that

$$\mathbf{b}_i^T \mathbf{X}_i^\dagger = \frac{1}{\lambda_i}\mathbf{b}_i^T \mathbf{B}_i \mathbf{X}_i^\dagger = \frac{1}{\lambda_i}\mathbf{b}_i^T \left(\mathbf{X}_i^T - \frac{\mathbf{X}_i^T \mathbf{X}_i \mathbf{e}\mathbf{e}^T \mathbf{X}_i^T}{\mathbf{e}^T \mathbf{X}_i^T \mathbf{X}_i \mathbf{e}}\right) \text{ by Eq. 5.6}$$

$$= \frac{1}{\lambda_i}\mathbf{b}_i^T \left(\mathbf{X}_i^T - \frac{(\mathbf{X}^T\mathbf{X} - \sum_{j=1}^{i-1}\beta_j \mathbf{b}_j \mathbf{b}_j^T)\mathbf{e}\mathbf{e}^T \mathbf{X}_j^T}{\mathbf{e}^T(\mathbf{X}^T\mathbf{X} - \sum_{j=1}^{i-1}\beta_j \mathbf{b}_j \mathbf{b}_j^T))\mathbf{e}}\right) \text{ by Eq. 5.29}$$

$$= \frac{1}{\lambda_i}\mathbf{b}_i^T \left(\mathbf{X}_i^T - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T \mathbf{X}_i^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X}\mathbf{e}}\right)$$

$$= \frac{1}{\lambda_i}\mathbf{b}_i^T \left(\mathbf{I} - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X}\mathbf{e}}\right)\left(\mathbf{X} - \sum_{j=1}^{i-1}\mathbf{c}_j\mathbf{c}_j^T\mathbf{X}\right)^T \text{ by Eq. 5.12}$$

$$= \frac{1}{\lambda_i}\mathbf{b}_i^T \left(\mathbf{I} - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X}\mathbf{e}}\right)\mathbf{X}^T - \frac{1}{\lambda_i}\mathbf{b}_i^T \left(\mathbf{I} - \frac{\mathbf{X}^T\mathbf{X}\mathbf{e}\mathbf{e}^T}{\mathbf{e}^T \mathbf{X}^T \mathbf{X}\mathbf{e}}\right)\left(\sum_{j=1}^{i-1}\mathbf{X}^T\mathbf{c}_j\mathbf{c}_j^T\right)$$

$$= \mathbf{m}_i^T - \sum_{j=1}^{i-1}\mathbf{m}_i^T\mathbf{c}_j\mathbf{c}_j^T \text{ by Eq. 4.17 .} \tag{5.33}$$

Since $\mathbf{m}_i$ is on the span of $\mathbf{c}_i$, we have $\mathbf{m}_i^T\mathbf{c}_j\mathbf{c}_j^T = 0$. Therefore

$$\mathbf{b}_i^T \mathbf{X}_i^\dagger = \mathbf{m}_i^T = \mathbf{b}_i^T \mathbf{X}^\dagger. \tag{5.34}$$

□

The meaning of Theorem 5.3 is, if we take out the part of data in $\mathbf{X}$ that lies along the span of $\mathbf{c}_1$ with Eq. 5.12 and use the data left $\mathbf{X}_2$ to build the new modularity matrix $\mathbf{B}_2$ with Eq. 5.13, all the eigenpairs of $\mathbf{B}$ are kept in $\mathbf{B}_2$ except for the first pair. Moreover, by Eq. 5.14 the modularity component corresponding to the largest eigenvalue of $\mathbf{B}_2$ is $\mathbf{b}_2$, which is also the modularity component corresponding to the second largest eigenvalue of $\mathbf{B}$. Similarly, if we take out the part of data in $\mathbf{X}$ that lies along the span of $\mathbf{c}_j$, $1 \leq j \leq i-1$, with Eq. 5.12 and use the data left $\mathbf{X}_i$ to build the new modularity matrix $\mathbf{B}_i$ with Eq. 5.13, although $\mathbf{d}_i$ and m_i are different from the original data so $\mathbf{B}_i$ is also different from $\mathbf{B}$, all the eigenpairs of $\mathbf{B}$ are kept in $\mathbf{B}_i$ except for the first $i-1$ pairs. Moreover, by Eq. 5.14 the modularity component corresponding to the largest eigenvalue of $\mathbf{B}_i$ is $\mathbf{b}_i$, which is also the modularity component corresponding to the i-th largest eigenvalue of $\mathbf{B}$. The conclusion is the first modularity component has the largest modularity of the data $\mathbf{X}$ and each succeeding modularity component has the largest modularity with the constraint that it is orthogonal to all previous modularity components.

Combining Theorem 5.2 and Theorem 5.3 we get the following corollary.

Corollary 5.4. *With the assumptions in Lemma 4.3, let $\mathbf{P}_{\mathbf{c}_i}$ be the projector onto the direction of $\mathbf{c}_i$. Then*

$$\mathbf{P}_{\mathbf{c}_i}\mathbf{X} = \sqrt{\beta_i}\mathbf{c}_i\mathbf{b}_i^T. \tag{5.35}$$

By Corollary 5.4, the "level of importance" of each modularity component is ordered by their corresponding eigenvalues of the modularity matrix $\mathbf{B}$. This corollary also tell us that if the data is projected onto the span of the modularity components, we will get a low-rank representation of the data. More specifically, if the first t modularity components are used, and let

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_t \end{pmatrix}, \tag{5.36}$$

then the t-dimensional representation of the uncentered data, or the score matrix, is

$$\mathbf{T} = \mathbf{C}^T\mathbf{X} = \begin{pmatrix} \sqrt{\beta_1} & 0 & \cdots & 0 \\ 0 & \sqrt{\beta_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\beta_t} \end{pmatrix} \begin{pmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_t^T \end{pmatrix}. \tag{5.37}$$

Then, as analogous to clustering with PCA, other clustering algorithms such as k-means can be used on the rows of $\mathbf{T}$ to get the clusters.

5.4 Some Discussions

5.4.1 Modularity Components versus Principal Components

In Section 2.2.4, we listed some important properties of the principal components. Here we list them again, then we summarize the properties of the modularity components:

- The principal components are orthogonal to each other.
- If we project the data onto the span of a principal component, we get a scalar multiple of the score vector that can reveal the cluster structure in the data based on the signs of the entries in the score vector.
- The first principal component has the largest variance of the data. Each succeeding principal component has the largest variance with the constraint that it is orthogonal to all previous principal components.

From Theorem 5.1 to Theorem 5.3, we can summarize the properties of the modularity components:

- Theorem 5.1 says that the modularity components are orthogonal to each other.
- Theorem 5.2 says that if we project the data onto the span of a modularity component, we get a scalar multiple of the score vector that can reveal the cluster structure in the data based on the signs of the entries in the score vector.
- Theorem 5.3 says that the first modularity component has the largest modularity of the data. Each succeeding modularity component has the largest modularity with the constraint that it is orthogonal to all previous modularity components.

If we compare the properties of modularity components with the ones of principal components, it can be seen that they are quite similar. Both principal components and modularity components are sets of orthonormal vectors. The projections of data onto the span of both kinds of components can reveal cluster structure. Both kinds of components have the property that a component has the largest "information" in the data with the constraint that it is orthogonal to all previous components. Both principal components and modularity components can give a low-rank representation of the data so they can both be used to perform dimension reduction. We will call the data analysis method that uses modularity components to reveal the cluster structure and perform dimension reduction the **modularity component analysis**, or **MCA**. Figure 5.1 is a 2-dimensional representation of the iris data given by MCA. It can be compared with the representation given by PCA in Figure 2.1. Since the modularity components and the principal components are so similar, it can be expected that the cluster structure revealed by the two kinds of components are comparable. However, there is a point that is not shared by principal components and modularity components. The principal component analysis requires data centering before extracting the components and cluster structure, while it can be avoided with modularity components. Since both PCA and MCA requires using SVD, if a SVD solver can take the advantage of the sparsity of the input data, PCA may not have this efficiency. On the other hand, the MCA does not require data centering so it can benefit from the sparsity of the data.

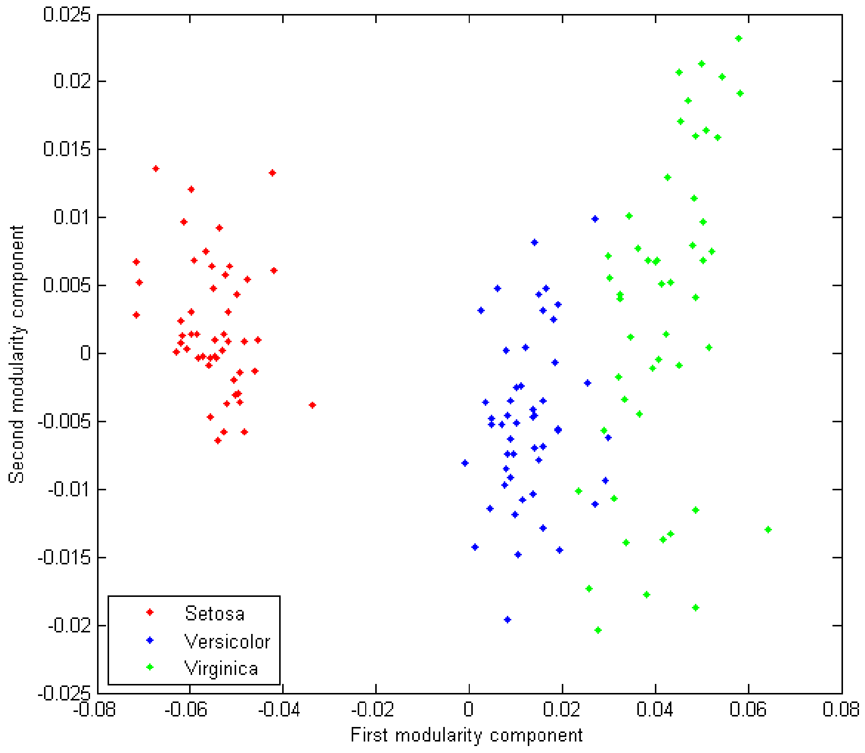

Figure 5.1: A 2-dimensional representation of the iris data given by MCA.

5.4.2 Dimension Reduction and Number of Modularity Components

Since the modularity components have properties that are analogous to the properties of the principal components, the modularity component analysis can also be used as to reduce the number of dimensions. The first dimension lies along the span of the first modularity component, and has the largest modularity of the normalized data. The p-th dimension lies along the span of the p-th modularity components, and has the largest modularity of the normalized data with the constraint that it is orthogonal to all preceding modularity components. The first a few components are expected to have the majority of the modularity of the whole data.

While the modularity component analysis provide a possible way of performing dimension reduction, to determine the number of modularity components is still an open question. It may be helpful to borrow some of the ideas for determining the number of principal components discussed in Section 2.2.5. In Chapter 6 a scree plot for modularity component analysis is used to determine the number of modularity components.

5.4.3 Is Lemma 4.3 Legitimate?

It can be seen that the theorems proven in this chapter are based on the condition that the Lemma 4.3 holds true. Lemma 4.3 assumes the largest $k-1$ eigenvalues of the modularity matrix are not equal to any of the largest eigenvalues of the adjacency matrix. The assumption is necessary to make the denominator of Eq. 4.5 nonzero. The assumption holds true for all the practical cases discussed in Chapter 6.

5.4.4 Algorithm of Modularity Component Analysis

As a conclusion of the discussion in this chapter, here we write down the algorithm to perform modularity component analysis. Note that although the components are not explicitly computed, they give the theoretical support to using multiple eigenvectors of the modularity matrix to reveal the cluster structures in data.

Algorithm 9 Clustering with Modularity Component Analysis

```
Require: A $p \times n$ normalized data matrix $\mathbf{X}$, number of clusters $k$, number of modularity components $t$.
    • Compute the modularity matrix $\mathbf{B}$ with eq. 3.1.
    • Compute the largest $t$ eigenvalues of $\mathbf{B}$, $\beta_1 > \beta_2 > \cdots > \beta_t > 0$, and their corresponding eigenvectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_t$.
    • Compute the score matrix $\mathbf{S} = \mathbf{\Sigma}\mathbf{T}^T$, where $\mathbf{\Sigma} = \mathrm{diag}(\sqrt{\beta_1}, \sqrt{\beta_2}, \cdots, \sqrt{\beta_t})$ and $\mathbf{T} = (\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_t)$.
    • Let $\mathbf{p}_i$ be the $i$-th colomn of $\mathbf{S}$.
    • Cluster the $\mathbf{p}_i$'s with k-means into clusters $C_1$, $C_2$, $\cdots$, $C_k$.
  return Clusters $G_1$, $G_2$, $\cdots$, $G_k$ such that $G_i = \{j|\mathbf{p}_j \in C_i\}$.
```

CHAPTER 6

Numerical Experiments

In this chapter we will examine the performance of MCA combined with k-means. Three traditional clustering methods, k-means, modularity partitioning method by Newman and Girvan [63] and PCA combined with k-means are used as baseline methods. In Section 6.1 the datasets used in the experiments will be introduced. Then in Section 6.2 the time consumed by each method and their accuracy will be listed and discussed.

6.1 Datasets

6.1.1 Wine Dataset

The wine recognition data from the UCI data repository [49] is one of the most famous data sets used in data mining [28, 34, 66]. The data set is a result of chemical analysis of wines growing in the same region. The difference between the wines is that they are derived from three different cultivars. The data contains 178 wine samples, the labels of the samples that tell which kind of wine each sample is and 13 variables from chemical analysis. The variables are listed as follows:

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

The goal is to group the 130 samples into three clusters corresponding to the three cultivars.

6.1.2 Medlars, Cranfield, CISI Dataset

The Medlars, Cranfield and CISI dataset, also known as the MCC dataset or Classic3 dataset, is a widely used dataset for clustering [18, 17, 1]. The data contains 3891 documents from three smaller datasets in different fields:

1. Medlars: 1033 medical documents
2. Cranfield: 1398 aerodynamics documents
3. CISI: 1460 information science documents

Each data point is a row vector representing a document, with the entries the frequency of the words in the document. After preprocessing there are 11,001 terms appeared in at least one document, so each data is a 11,000 by 1 vector. The main goal is to separate the documents from the three smaller collections.

6.1.3 PenDigit Dataset

The PenDigit data set is a subset of the Mixed National Institute of Standards and Technology (MNIST) database [47]. The original data contains a training set of 60,000 handwritten digits from 44 writers. Each piece of data is a row vector converted from a greyscale image. Each image is 28 pixels in height and 28 pixels in width, so there are 784 pixels in total. Each row vector contains the label of the digit and the lightness of each pixel. Lightness of a pixel is represented by a number from 0 to 255 inclusively, and smaller numbers represent lighter pixels. The data used in the experiments contains digits 1, 2, 6, and 7. For each digit 200 samples are randomly selected from the original data. The number of principal components and modularity components are picked based on the eigenvalues plot of the modularity matrix. The scree plot of the modularity matrix shows how much modularity is reflected by each of the modularity components.

6.2 Experimental Setup and Results

6.2.1 Experimental Setup

For each of the datasets the k-means algorithm in Algorithm 1, Newman and Girvan's modularity partitioning algorithm in Algorithm 6, PCA with k-means in Algorithm 2 and MCA with k-means in Algorithm 9 are applied. When the k-means algorithm was involved the algorithm was run several times and the result with lowest SSE was recorded. For the wine dataset and MMC dataset the number of principal components were determined by the scree plots of PCA, which will be given in the next section. For these datasets the numbers of modularity components were set to be the same as the number of principal components. For the PenDigit dataset the number of modularity component was determined by the scree plot of MCA, and the number of principal components was set to be the same as the number of modularity components. The wine and MCC datasets were centered and normalized, while the PenDigit dataset was only centered when PCA was applied.

6.2.2 Results

The experimental results for the time consumed by each algorithm and the accuracy are listed in Table 6.1 and Table 6.2. The number of principal/modularity components used for each dataset is listed in Table 6.3. It can be seen that for all datasets MCA consumes less time than PCA. This is expected as MCA does not required the centering step, which is necessary for PCA. The baseline algorithms, k-means and Newman and Girvan's modularity partitioning algorithms, use more time than PCA and MCA for most of the time. In the perspective of accuracy, MCA has a relatively stable accuracy, and the accuracy is comparable with the best results for all datasets used. Next we post the scree

Table 6.1: Time Consumed by Each Algorithm. (Unit: Second)

	k-means	Modularity	PCA	MCA
Wine	0.0323	0.0651	0.0456	0.0426
MCC	361	17.0	3.50	2.65
PenDigit	0.187	0.210	0.0844	0.0695

Table 6.2: Accuracy by Each Algorithm.

	k-means	Modularity	PCA	MCA
Wine	0.702	0.781	0.949	0.921
MCC	0.405	0.984	0.932	0.962
PenDigit	0.656	0.418	0.861	0.848

plots for PCA and MCA, and the 2-dimensional representation plots of the data given by PCA and MCA for each datasets. For the PCA scree plots, the x-axes represent the indices of the eigenvalues, and the y-axes represent the values of the eigenvalues of $\mathbf{X}^T\mathbf{X}$ where $\mathbf{X}$ is the centered data matrix sometimes normalized. For the MCA scree plots, the x-axes represent the indices of the eigenvalues, and the y-axes represent the values of the

Table 6.3: Number of Principal/Modularity Components Used for Each Dataset.

	Wine	MCC	PenDigit
Number of components	4	3	4

eigenvalues of the modularity matrix. For the MCC and PenDigit datasets only the first 20 components are included in the scree plots. The 2-dimensional representation plots shows the projections of the data onto the span of the first two principal or modularity components. The x-axes represent the span of the first components, and the y-axes represent the span of the second components. A good low-rank representation should be able to show the difference between the data from different classes so high accuracy can be expected when the k-means algorithm is applied. For example, in Figure 6.7 and Figure 6.8 it can be seen that in the 2-dimensional representation of the MCC data, the documents from different fields lie along three different directions. Then it is relatively easy for k-means to separate the documents into three clusters. From the plots it can be seen that for all the datasets used, MCA can give similar low-rank representation of data compared with PCA, even MCA does not require data centering.

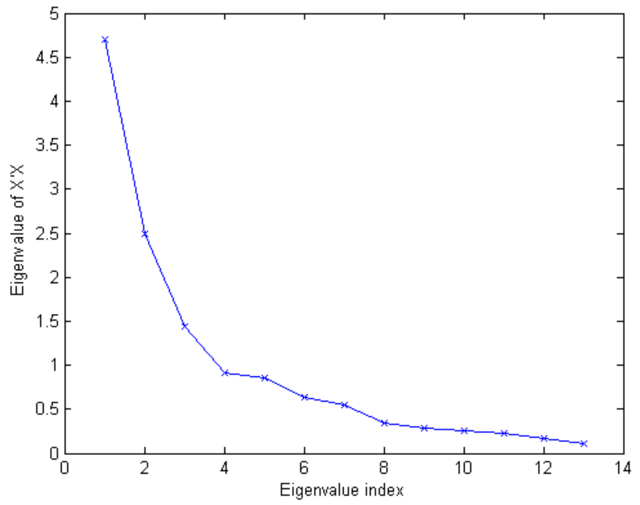

Figure 6.1: Scree plot of the wine data (centered and normalized) given by PCA.

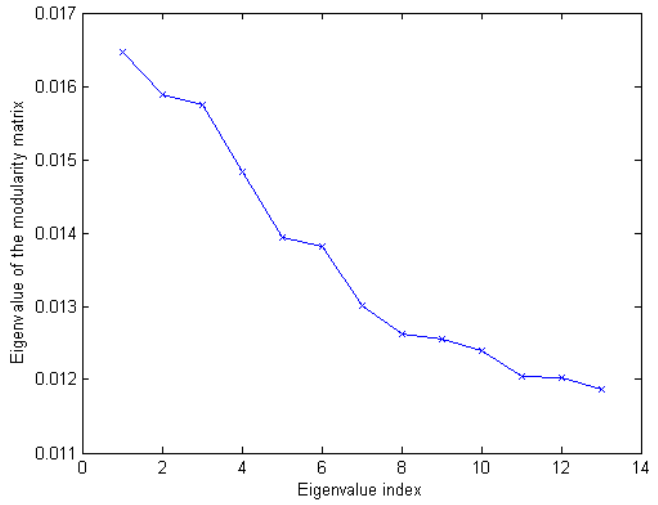

Figure 6.2: Scree plot of the wine data (normalized) given by MCA.

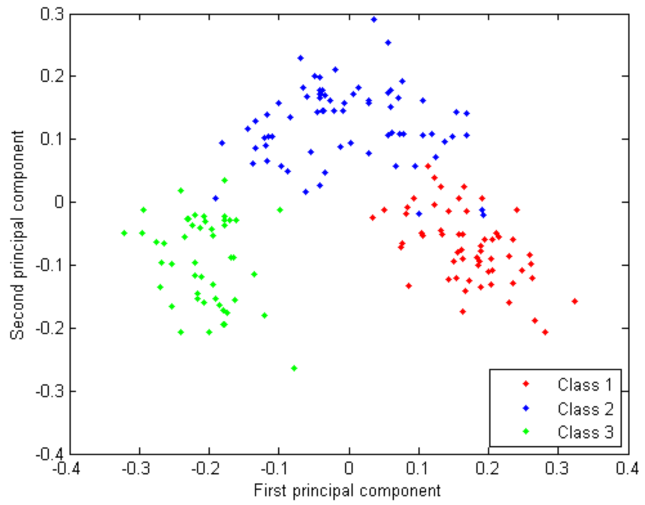

Figure 6.3: A 2-dimensional representation of the projection of the wine data onto the span of the first two principal components.

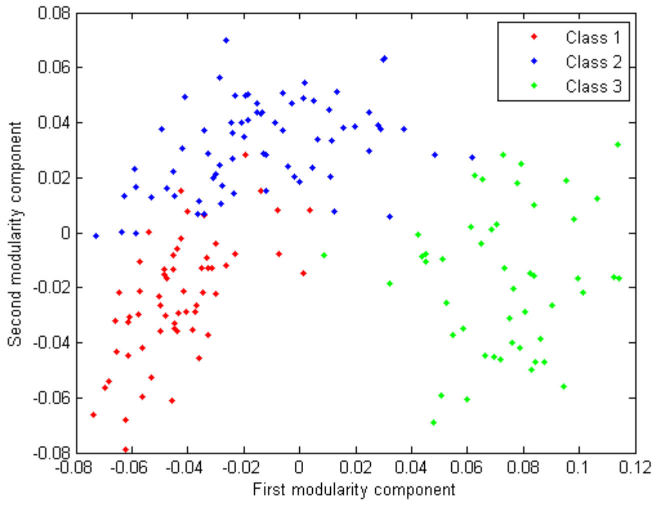

Figure 6.4: A 2-dimensional representation of the projection of the wine data onto the span of the first two modularity components.

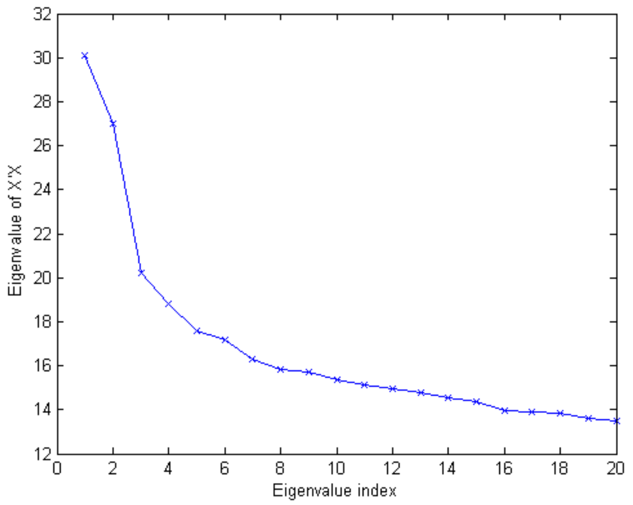

Figure 6.5: Scree plot of the MCC data (centered and normalized) given by PCA. First 20 components included.

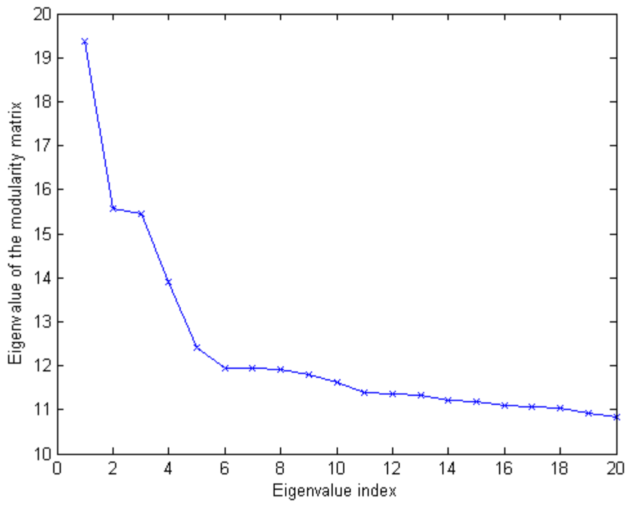

Figure 6.6: Scree plot of the MCC data (normalized) given by MCA. First 20 components included.

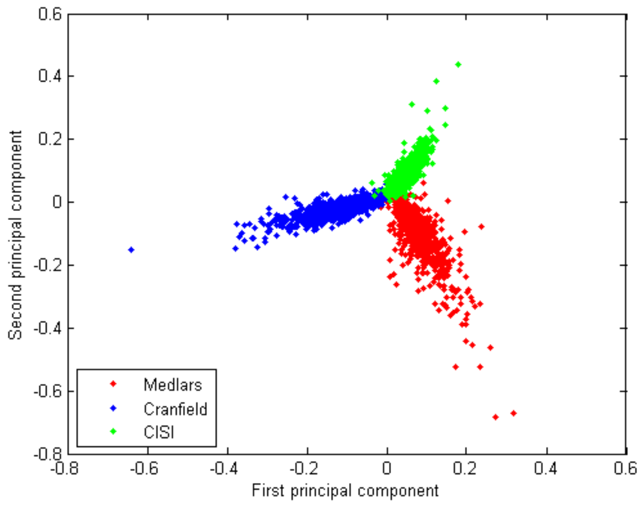

Figure 6.7: A 2-dimensional representation of the projection of the MCC data onto the span of the first two principal components.

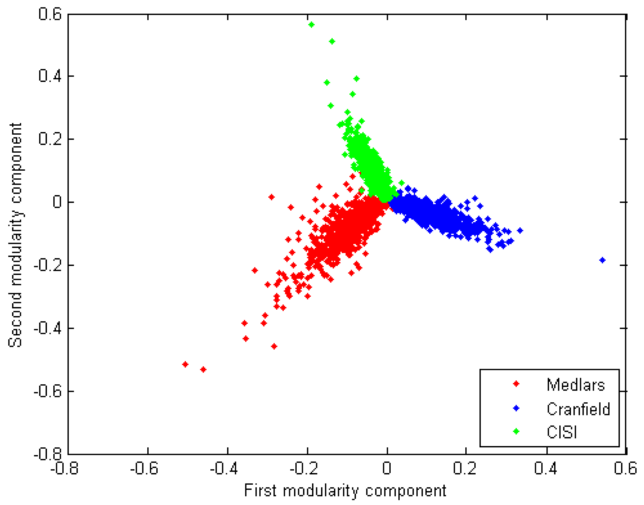

Figure 6.8: A 2-dimensional representation of the projection of the MCC data onto the span of the first two modularity components.

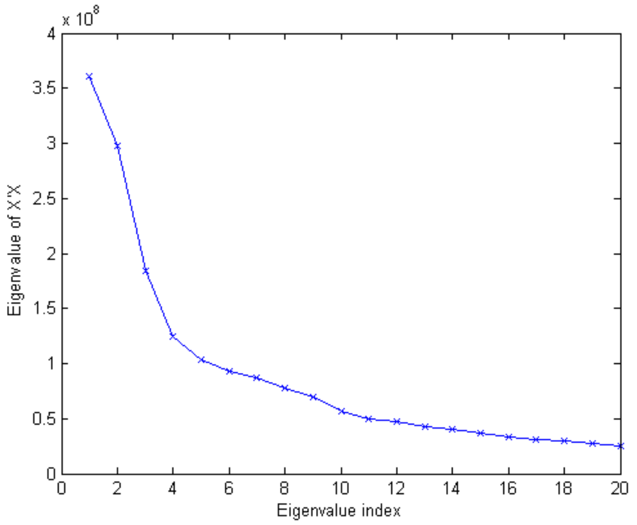

Figure 6.9: Scree plot of the PenDigit data (centered and unnormalized) given by PCA. First 20 components included.

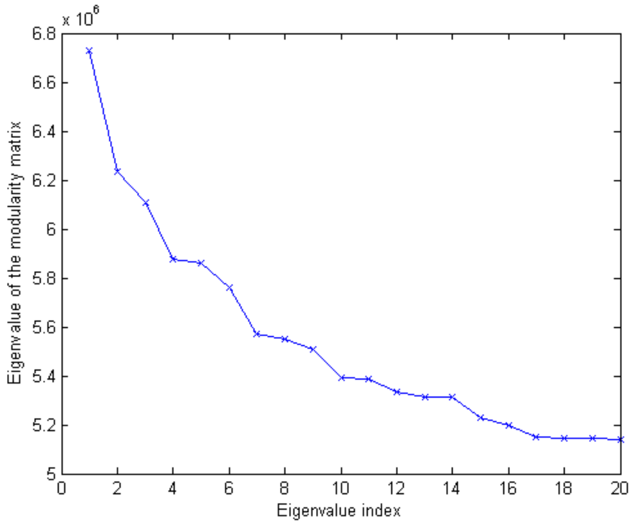

Figure 6.10: Scree plot of the PenDigit data (unnormalized) given by MCA. First 20 components included.

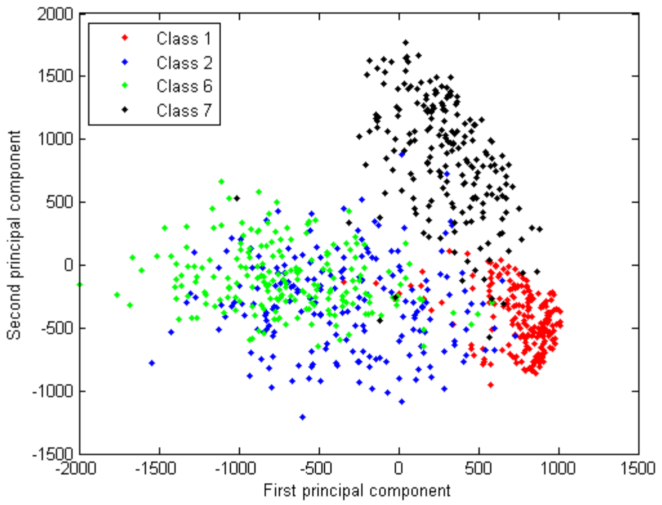

Figure 6.11: A 2-dimensional representation of the projection of the PenDigit data onto the span of the first two principal components.

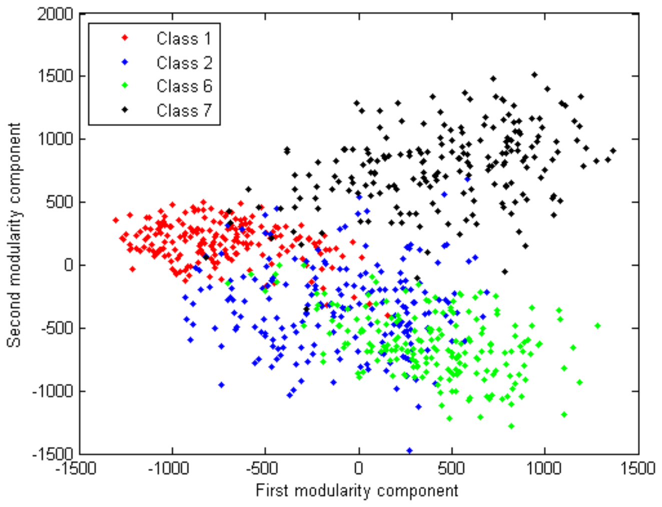

Figure 6.12: A 2-dimensional representation of the projection of the PenDigit data onto the span of the first two modularity components.

CHAPTER 7

Conclusion

The main purpose of this dissertation is to build the modularity components and discuss their properties. The discussion in this dissertation helps to address several problems in data analysis. First, when the traditional modularity partitioning algorithm is applied, the data or graph can only be partitioned into two groups in each iteration. When the desired number of clusters is more than two, a hierarchy has to be built and the same algorithm has to be applied for several times, and the total process would cost a lot of time and computations. While it is quite natural to use several eigenvectors of the modularity matrix to cluster data, the reason for doing so is still vague.

Second, while PCA is widely used in data analysis, one drawback is that the sparsity of the data may be destroyed while centering. Since the sparsity of the data can be utilized in performing SVD, the centering step in PCA makes the process less efficient. It is better if a data analysis algorithm can perform dimension reduction while keeping the sparsity of the data.

7.1 Contributions

In this dissertation the concept of modularity components is defined, and the properties of the components are developed. It is shown that the modularity components are orthonormal to each other, and the projections of the uncentered data onto the span of the components give scalar multiples of the eigenvectors of the modularity matrix. It is also proven that the first modularity component has the largest modularity in the data, and each succeeding modularity component has the largest modularity of the data with the restrict that it is orthogonal to all preceding components. The development of the modularity components in this dissertation gives theoretical justification for using multiple eigenvectors of the modularity matrix to cluster data.

It is also shown in this dissertation that the modularity component analysis provides a possible way of performing dimension reduction while keeping the sparsity of the original data. Just like PCA, the new dimensions given by MCA lies along the span of the modularity components. The sparsity of the original data is not destroyed since centering is not required for modularity clustering.

7.2 Future Research

1. Look for proper methods to determine the number of modularity components;
2. Look for numerical methods to efficiently compute the eigenvalues of modularity matrices;

3. Investigate more properties of the modularity components;
4. Use other clustering methods on the low-rank representation and compare the results with MCA combined with k-means;
5. Apply MCA on datasets with more variables or more data points to examine its efficiency and accuracy;
6. Apply MCA in facial recognition to see what facial information each modularity component can give.

BIBLIOGRAPHY

[1] Ralph Walter Abbey. *Stochastic clustering: Visualization and application.* North Carolina State University, 2014.

[2] Maurice S Bartlett. Tests of significance in factor analysis. *British Journal of statistical psychology*, 3(2):77–85, 1950.

[3] Eugenio Beltrami. Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Universita*, 11:98–106, 1873.

[4] Peter M Bentler and Ke-Hai Yuan. Tests for linear trend in the smallest eigenvalues of the correlation matrix. *Psychometrika*, 63(2):131–144, 1998.

[5] PM Bentler and Ke-Hai Yuan. Test of linear trend in eigenvalues of a covariance matrix with application to data analysis. *British Journal of Mathematical and Statistical Psychology*, 49(2):299–312, 1996.

[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[7] Daniel Boley. Principal direction divisive partitioning. *Data mining and knowledge discovery*, 2(4):325–344, 1998.

[8] Marianna Bolla. Penalized versions of the newman-girvan modularity and their relation to normalized cuts and k-means clustering. *Physical Review E*, 84(1):016108, 2011.

[9] James R Bunch, Christopher P Nielsen, and Danny C Sorensen. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978.

[10] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[11] Raymond B Cattell. The scree test for the number of factors. *Multivariate behavioral research*, 1(2):245–276, 1966.

[12] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[13] Bhaskar DasGupta and Devendra Desai. On the complexity of newman's community finding approach for biological and social networks. *Journal of Computer and System Sciences*, 79(1):50–67, 2013.

[14] Fernando De la Torre and Michael J Black. Robust principal component analysis for computer vision. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 362–369. IEEE, 2001.

[15] Peter Deuflhard, Wilhelm Huisinga, Alexander Fischer, and Ch Schütte. Identification of almost invariant aggregates in reversible nearly uncoupled markov chains. *Linear Algebra and its Applications*, 315(1):39–59, 2000.

[16] Peter Deuflhard and Marcus Weber. Robust perron cluster analysis in conformation dynamics. *Linear algebra and its applications*, 398:161–184, 2005.

[17] Inderjit Dhillon, Jacob Kogan, and Charles Nicholas. Feature selection and document clustering. In *Survey of Text Mining*, pages 73–100. Springer, 2004.

[18] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1-2):143–175, 2001.

[19] HT Eastment and WJ Krzanowski. Cross-validatory choice of the number of components from a principal component analysis. *Technometrics*, 24(1):73–77, 1982.

[20] Louis Ferré. Selection of components in principal component analysis: a comparison of methods. *Computational Statistics & Data Analysis*, 19(6):669–682, 1995.

[21] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[22] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

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

[24] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[25] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[26] Miguel A Fortuna, Daniel B Stouffer, Jens M Olesen, Pedro Jordano, David Mouillot, Boris R Krasnov, Robert Poulin, and Jordi Bascompte. Nestedness versus modularity in ecological networks: two sides of the same coin? *Journal of Animal Ecology*, 79(4):811–817, 2010.

[27] Scott B Franklin, David J Gibson, Philip A Robertson, John T Pohlmann, and James S Fralish. Parallel analysis: a method for determining significant principal components. *Journal of Vegetation Science*, 6(1):99–106, 1995.

[28] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2004.

[29] Benjamin H Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[30] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-aided design of integrated circuits and systems, ieee transactions on*, 11(9):1074–1085, 1992.

[31] Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[32] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[33] Mia Hubert and Sanne Engelen. Robust pca and classification in biosciences. *Bioinformatics*, 20(11):1728–1736, 2004.

[34] Hisao Ishibuchi and Takashi Yamamoto. Rule weight specification in fuzzy rule-based classification systems. *Fuzzy Systems, IEEE Transactions on*, 13(4):428–435, 2005.

[35] J Edward Jackson. *A user's guide to principal components*, volume 587. John Wiley & Sons, 2005.

[36] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[37] Jyh-Cheng Jeng. Adaptive process monitoring using efficient recursive pca and moving window pca algorithms. *Journal of the Taiwan Institute of Chemical Engineers*, 41(4):475–481, 2010.

[38] Hansi Jiang and Carl Meyer. Modularity component analysis versus principal component analysis. *arXiv preprint arXiv:1510.05492*, 2015.

[39] Hansi Jiang and Carl Meyer. Relations between adjacency and modularity graph partitioning. *arXiv preprint arXiv:1505.03481*, 2015.

[40] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[41] Ian T Jolliffe. Discarding variables in a principal component analysis. i: Artificial data. *Applied statistics*, pages 160–173, 1972.

[42] Camille Jordan. Mémoire sur les formes bilinéaires. *Journal de mathématiques pures et appliquées*, pages 35–54, 1874.

[43] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004.

[44] Jacob Kogan. *Introduction to clustering large and high-dimensional data*. Cambridge University Press, 2007.

[45] Wojtek J Krzanowski and Paul Kline. Cross-validation for choosing the number of important components in principal component analysis. *Multivariate Behavioral Research*, 30(2):149–165, 1995.

[46] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical review E*, 84(6):066122, 2011.

[47] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[48] George Leitmann. On one approach to the control of uncertain systems. *Journal of Dynamic Systems, Measurement, and Control*, 115(2B):373–380, 1993.

[49] M. Lichman. UCI machine learning repository, 2013.

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

[51] Helmut Lutkepohl. Handbook of matrices. *Computational Statistics and Data Analysis*, 2(25):243, 1997.

[52] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[53] John Mandel. Principal components, analysis of variance and data structure. *Statistica Neerlandica*, 26(3):119–129, 1972.

[54] Ryan A Mercovich, Anthony Harkin, and David Messinger. Automatic clustering of multispectral imagery by maximization of the graph modularity. In *SPIE Defense, Security, and Sensing*, pages 80480Z–80480Z. International Society for Optics and Photonics, 2011.

[55] David Meunier, Renaud Lambiotte, Alex Fornito, Karen D Ersche, and Edward T Bullmore. Hierarchical modularity in human brain functional networks. *Hierarchy and dynamics in neural networks*, 1:2, 2010.

[56] Carl D Meyer. *Matrix analysis and applied linear algebra*. Siam, 2000.

[57] Carl D Meyer and Charles D Wessell. Stochastic data clustering. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1214–1236, 2012.

[58] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 786–793. IEEE, 1995.

[59] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1-2):91–118, 2003.

[60] Jerome Moore, Eui-Hong Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, and Bamshad Mobasher. Web page categorization and feature selection using association rule and principal component clustering. *IBM shared research report/University of Minnesota (Minneapolis, Mn.)*, 98:3, 1997.

[61] Hiroshi Murase and Shree K Nayar. Visual learning and recognition of 3-d objects from appearance. *International journal of computer vision*, 14(1):5–24, 1995.

[62] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[63] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[64] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[65] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[66] Shaina Race. *Iterated Consensus Clustering*. North Carolina State University, 2014.

[67] Shaina Race, Carl Meyer, and Kevin Valakuzhy. Determining the number of clusters via iterative consensus clustering. *arXiv preprint arXiv:1408.0967*, 2014.

[68] C Radhakrishna Rao. The use and interpretation of principal component analysis in applied research. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 329–358, 1964.

[69] John Robert Reddon. The number of principal components problem: a monte carlo study. Unpublished, 1984.

[70] Randolf Rotta and Andreas Noack. Multilevel local search algorithms for modularity clustering. *Journal of Experimental Algorithmics (JEA)*, 16:2–3, 2011.

[71] Enrique H Ruspini. A new approach to clustering. *Information and control*, 15(1):22–32, 1969.

[72] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[73] Hugo Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804, 1956.

[74] T Sugiyama and Howell Tong. On a statistic useful in dimensionality reduction in multivariable linear stochastic system. *Communications in statistics-theory and methods*, 5(8):711–721, 1976.

[75] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. *Introduction to data mining*, volume 1. Pearson Addison Wesley Boston, 2006.

[76] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[77] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[78] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

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

[80] Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, pages 555–586, 2008.

[81] Ulrike Von Luxburg, Olivier Bousquet, and Mikhail Belkin. Limits of spectral clustering. In *Advances in neural information processing systems*, pages 857–864, 2004.

[82] Xun Wang, Uwe Kruger, and George W Irwin. Process monitoring approach using fast moving window pca. *Industrial & Engineering Chemistry Research*, 44(15):5691–5702, 2005.

[83] Merriam Webster. Merriam-webster online dictionary. http://www.merriam-webster.com/dictionary/cluster

[84] Charles Wessell. *Stochastic Data Clustering.* North Carolina State University, 2011.

[85] James Hardy Wilkinson. *The algebraic eigenvalue problem*, volume 87. Clarendon Press Oxford, 1965.

[86] Svante Wold. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405, 1978.

[87] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088, 2009.

[88] Ka Yee Yeung and Walter L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[89] Shuqin Zhang and Hongyu Zhao. Normalized modularity optimization method for community identification with degree adjustment. *Physical Review E*, 88(5):052802, 2013.