Title
Spectral methods for data clustering

Author
He, Xiaofeng

Publication Date
2002-05-31

LBNL-50205

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

Spectral Methods for Data Clustering

Xiaofeng He

Computing Sciences Directorate

May 2002

Ph.D. Thesis

DISCLAIMER

LBNL-50205

Spectral Methods for Data Clustering

Xiaofeng He
Ph.D. Thesis

Computing Sciences Directorate
Ernest Orlando Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720

May 2002

This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Spectral Methods for Data Clustering

by

Xiaofeng He

The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

SPECTRAL METHODS FOR DATA CLUSTERING

A Thesis in

Computer Science and Engineering

by

Xiaofeng He

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2002

Abstract

With the exponential growth of digital information in general, and on the Internet in particular, there is great demand for developing efficient and effective methods for organizing and retrieving available information. Examples include image segmentation and scientific data sets such as genomics and climate data sets. With the rapid expansion of the Internet, web document clustering plays an increasingly important role in information retrieval and taxonomy management for the World Wide Web and remains an interesting and challenging problem in the field of web computing.

In the past decade, one of the most active research areas of data clustering methods has been spectral graph partition. My thesis focuses on developing the new spectral graph partitioning methods to discover new patterns and knowledge of large data sets, especially the data set of web documents and pure text data. The *Min-Max* (Mcut) criterion is proposed following the experiments and theoretical analysis of its better performance over other popular methods such as the *normalized cut* (Ncut) and K-means methods. The application of the normalized cut and Min-max cut on the bipartite graph is developed theoretically. The close relations between the spectral method and other information retrieval methods such as Principle Direction Divisive Partitioning (PDDP) and the K-means method are discussed. Correspondence analysis is investigated in great detail from the point of view of the spectral bi-clustering method.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am most grateful and indebted to my thesis advisor, Hongyuan Zha, for the large doses of guidance, patience, and encouragement. I am especially indebted for the financial support that he has provided to me over the years. I am also grateful to Jesse Barlow for inspiration, enlightening discussions, and valuable suggestions in this research. I wish to thank Lee Giles for many interesting and inspiring discussions during my research. I thank my committee member, Bing Li, for his insightful commentary on my work.

I wish to thank following people for their help with my research. Horst D. Simon (Lawrence Berkeley National Laboratory) provided me with the opportunity as a student assistant to work with the Lawrence Berkeley National Lab. Chris Ding (Lawrence Berkeley National Laboratory) suggested many research topics. His intuition is very beneficial to me. Ming Gu (Math Department, U.C. Berkeley) had very important and interesting discussions with me. Hao Sun (Oracle) provided data on the Web link graph.

For their unconditional love, understanding, and moral support along the way, I am forever grateful to my mother and father.

This work was supported by NSF grant No. CCR-9901986 and partially by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098

Chapter 1

Introduction

1.1 Research motivation

The development of modern digital technology makes the availability of the information much more accessible to the general public. For example, users of the World Wide Web can readily access vast amount of information with a simple click of the mouse button. Currently, one dilemma facing people is the ability to distinguish relevant information from the great amount of information that exists, much of which is not organized. Similar questions arise in scientific and commercial areas. In these areas, large quantities of data need to be processed in order to reveal the underlying relation among seemingly irrelevant data objects. One research field, data analysis, is aimed at accomplishing this task.

Data analysis seeks to organize the data objects into groups based on either the hypothetical model or natural clustering of the data objects through analysis. Each data object assigned to a group is given a corresponding label. The method of grouping the unlabeled data objects based on some *labeled* data objects is called data classification, which is different from data clustering, which groups the data objects into meaningful clusters in which all the data objects are *unlabeled.* In many applications, there is no prior knowledge about the label information of any data object (the extreme case is not

knowing how many groups there should be), thus the clustering method is important for the data analysis.

With the advanced information technology, the size of data set becomes larger and larger. The definition of a large data set has evolved from several thousand data objects [69] to millions. It is impossible to manually cluster such a large data set, hence the challenge to researchers in data clustering is to develop more efficient and effective methods to retrieve information hidden in large data sets from disparate sources.

One important category of the clustering methods is the spectral graph partitioning method using a graph model. The spectral graph partitioning method takes an algebraic approach and has become popular since the early 1990s. In this thesis, we propose some novel spectral graph partitioning approaches for data clustering.

The following two sections discuss some background about the data clustering and spectral graph partitioning. Section 1.4 outlines the rest of the thesis.

1.2 Data clustering

Data clustering has been used extensively to explore the data patterns in the field such as information retrieval, data mining, image segmentation, handwriting recognition, and pattern classification. The purpose of data clustering is to discover the structure hidden in a data set. It is an unsupervised classification of the data objects into meaningful groups (or clusters) so that the data objects in the same group are more closely related to each other than those in different group. Data clustering is also used to discover whether there are distinct subgroups within the data set. That is equivalent to finding the dissimilarity among different data subgroups.

Naturally, to find the similarity/dissimilarity between data objects, the first important issue is how to measure the degree of similarity/dissimilarity between two data objects. This measure is subject to the real world applications and the viewpoints of individuals regarding the data set. The computation of similarity/dissimilarity can be based on the features (or attributes, variables) of the data objects or based on a single similarity/dissimilarity measure formed by combining individual features. The features are classified into three types: quantitative, ordinal, and categorical [27]. Different types of features should be combined in different ways. According to the different approaches of measuring the degree of similarity/dissimilarity, data clustering methods can be described as feature based or similarity based of the data objects.

Feature-based clustering methods group data objects with similar features or attributes. The data objects are expressed as the vectors of features. For example, in the problem of document clustering, the data object is a document with each word in the document as a feature of the document. Feature selection should be performed to eliminate from consideration those features that are irrelevant for the purpose of clustering. It is essential when the number of features are very large. The selection of the appropriate features is application dependent. Careful feature selection can improve the clustering result significantly, while poorly selected features can burden the task of interpretation of the clustering result. Feature selection is well established in statistical pattern recognition [26] but needs a trial-and-error process in the clustering context. Feature extraction, which extracts new features from the original data set, is also necessary. One of the most popular feature extraction techniques is Principle Component Analysis [51].

Similarity-based clustering methods group together data objects with high similarities. Careful selection of the similarity measurement is crucial to the success of the clustering methods. Similarity can be computed from the feature space. The most popular similarity measure is the *Euclidean distance* $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent two data objects in the feature space. To eliminate the problem of the feature vectors with large magnitude dominating the ones with small magnitude, the normalized form of the feature vectors should be used. One normalized form of similarity measurement is the cosine of two data objects that we used to compute the document similarity in our experiments. The cosine of two data objects $\mathbf{x}_i, \mathbf{x}_j$ is defined as

$$\frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2} \tag{1.1}$$

where $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$.

Based on the structures of the partition, there are two main techniques in cluster analysis: *hierarchical clustering method* and *partitional clustering method.* Hierarchical clustering method produces nested hierarchical structure of clusters. A tree diagram, or dendrogram, is used to represent the hierarchically nested set of partitions. *Agglomerative algorithm* and *divisive algorithm* are two main approaches of hierarchical method.

Partitional clustering method produces a partition of only one level structure. It partitions the entire data set into a pre-defined number of disjoint groups which optimize a specified criterion.

If we allow the partitions to be overlapped, namely, the partitions that have elements in common, then this clustering method is called the *clumping method.*

Hierarchical method, partitional method, and clumping method together are referred to as cluster analysis. In the real application, these three techniques can be nested to apply to the data set according to the practical problem. There exists a large amount of literature on clustering methods and algorithms [27, 29, 37]. For a complete survey of data clustering, see [50].

1.3 Spectral graph partition

Spectral graph partition is a very important research field dedicated to solving data clustering problems. In spectral graph partition, the original clustering problem is first transformed to a graph model; then, the graph is partitioned into subgraphs using a linear algebraic approach.

1.3.1 Graph model for data clustering

Graph partition has a very broad range of applications. At one end are the near-regular graphs, such as the mesh of a 2D surface of an airfoil or a 3D engine cylinder. Partitioning such a mesh into subdomains for distributed memory processors is a common task. Several popular software packages for this partitioning task have been developed [53, 44]. More applications of graph partition on parallel computing can be found in [45, 48, 61, 64]. At another end are the highly irregular or random graphs, such as the link graphs of the World Wide Web. The degrees of the vertices (web documents) in these graphs vary dramatically. Partitioning the graph is useful in automatically identifying topics from the retrieved web pages for a user query.

In the graph partitioning method, the partitioning problem is modeled as an undirected graph. For a feature-based clustering problem, the data objects and features are treated as the vertices of the graph. The edge exists only between two vertices representing the data object and feature, respectively. There is no edge linking two vertices with the same type, for example, two vertices representing two features. The weight of the edge is normally taken to be the number of occurrences of the feature in the data object. The graph formed this way is a bipartite graph $G(X, Y, E)$ where X, Y are the sets of data objects and features, respectively, and E is the edge set. Usually, the weight matrix W of the bipartite graph is a rectangular one with the rows denoting the features and columns denoting the data objects. This graph partitioning problem is called bipartite graph partition. Figure 1.1 illustrates a bipartite graph. In this figure, there are two types of data objects, r-type and c-type. After partitioning, r-type data set R is partitioned into R_1 and R_2, while c-type data set C is partitioned into C_1 and C_2, represented by the vertical dashed line. Clearly, R_1 is grouped with C_1 because of their large connection; the same is true for R_2 with C_2. Many data types arising from data mining applications can be modeled as bipartite graphs. Examples include terms and documents in a text corpus, customers and purchasing items in market basket analysis, and reviewers and movies in a movie recommender system.

For similarity-based clustering problems, the data objects consist of the vertex set V of the graph model $G(V, E)$ with the similarity of the data objects as the weight of the edge between corresponding vertices. In other words, the vertices of the graph are of the same type. The edge weight is taken to be the similarity between two data objects.

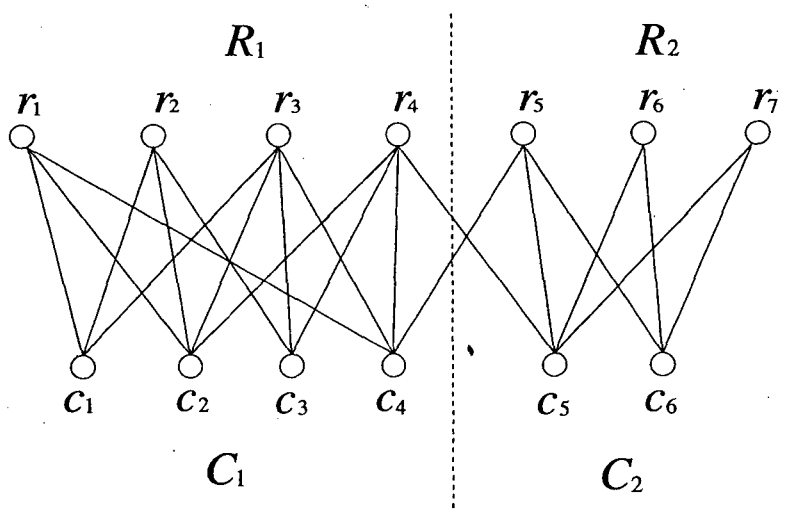

Fig. 1.1. Bipartite graph: There are two types of data objects, R and C.

The weight matrix W formed is therefore a symmetric square matrix. One important example is the load balance among parallel processors.

1.3.2 Spectral graph partitioning methods

The optimal solution to the graph partitioning problem is NP-hard because of the combinatoric nature of the problem. To reduce the NP-hard problem to a solvable one, an effective approach is to give a linear order to the data objects and then find an optimal cut point based on this linear order. One example is to compute a principal direction/component (principal eigenvector of the weight matrix) and find a cut point along this direction so that all points on one side belong to one subgraph and all points on the other side belong to another subgraph. This establishes a linear search order that effectively reduces the original NP hard problem to a linear solution. This example belongs to a large range of partitioning methods called spectral graph partitioning

methods. Spectral methods relax the discrete optimization problem into a continuous one which makes the partitioning problem tractable. It is an active research area of clustering methods. Spectral methods are based on the early work of Donath and Hoffman [22, 23], Fiedler [31, 32], and many other researchers (see a review by Mohar [65]). It was popularized by the work of Pothen, Simon, Liou [68], and many other follow-up studies [46, 74, 77] on graph bisection ($|A| = |B|$) cases where A and B are the sets of vertices for two respective subgraphs. Using indicator variables x_u,

$$x_u = \begin{cases} 1 & \text{if} \quad u \in S \\ -1 & \text{if} \quad u \in \bar{S}, \end{cases}$$

the cut size can be conveniently written as

$$\text{cut}(A, B) = \sum_{e_{uv} \in E} \frac{1}{4}(x_u - x_v)^2 w_{uv} = \frac{1}{2}\, \mathbf{x}^T (D - W)\mathbf{x}. \tag{1.2}$$

From (1.2), we see that to minimize the cut size, x_u and x_v should be close to each other if the value of w_{uv} is large. This implies that the minimization methods should assign these two vertices near each other when they have a large connection. Relaxing x_u from $\{1, -1\}$ to the continuous value in $[-1, 1]$ and assuming $\mathbf{x}$ is normalized, that is, $\|\mathbf{x}\|_2 = 1$, the minimization problem (1.2) is equivalent to minimizing

$$\frac{\mathbf{x}^T (D - W)\mathbf{x}}{\mathbf{x}^T \mathbf{x}},$$

which is a Rayleigh quotient. Hence minimizing (1.2) is reduced to solve the following eigensystem

$$(D - W)\mathbf{x} = \lambda \mathbf{x}. \tag{1.3}$$

where $W = (w_{ij})$ is the weight matrix of the graph, $D = \text{diag}(W\mathbf{e})$ and $\mathbf{e}$ is the vector of all 1's with appropriate dimension. $L = D - W$ is called the Laplacian matrix. Since the trivial eigenvector $\mathbf{x}_1 = \mathbf{e}/\sqrt{n}$, where n is the number of vertices, is associated with $\lambda_1 = 0$ which is the smallest eigenvalue of (1.3), the second smallest eigenvector $\mathbf{x}_2$ is the solution to the partitioning problem.

Spectral partitioning methods use eigenvectors of the Laplacian matrix $L = D - W$ to partition the graph. In particular, the eigenvector corresponding to the second smallest eigenvalue is most useful to partition the graph. This eigenvector is called the *Fiedler vector* in recognition of Fiedler's pioneering work.

The Fiedler vector provides a good linear order for searching for an optimal cut point. Naturally, the criteria used to find the cut point will largely determine the final quality of the partition results. In graph theory, a set of edges that separate the graph into two disconnected parts is called an edge-cut (edge separator) of the graph. The simplest objective function measuring the quality of the partition result is defined to be the cut size alone. This definition gives rise to standard *minimum cut* algorithm (Mincut). The expression of the Mincut criterion is given below:

$$\text{Mincut}(A, B) \equiv \text{cut}(A, B). \tag{1.4}$$

The Mincut often causes an unbalanced partition; it may cut a portion of a graph with a small number of vertices [13, 14]. In the context of graph clustering, this is, in general, not desirable. To avoid partitioning out a small part of a graph by using edge-cut alone, many criteria utilize various normalized forms of edge-cut, which are generally obtained by dividing the edge-cut by some measure of the size of the partitions.

One normalized form of the edge-cut is called the *ratio cut* (Rcut). The Rcut partitioning criterion is first used in the area of VLSI design [14, 78]. It is defined below.

$$\text{Rcut}(A, B) \equiv \frac{\text{cut}(A, B)}{|A|} + \frac{\text{cut}(A, B)}{|B|}. \tag{1.5}$$

Hagen and Kahng [41] remove the requirement $|A| = |B|$ imposed by Pothen, *et al*[68] in the bisection case. They show that the Fiedler vector obtained by solving (1.3) provides a good linear search order to the Rcut partitioning criterion. A K-way partitioning algorithm based on the Rcut criterion is proposed in [11].

The *normalized cut* (Ncut) is another such normalized measure that tends to minimize the objective function

$$\text{Ncut}(A, B) \equiv \frac{\text{cut}(A, B)}{\text{W}(A, V)} + \frac{\text{cut}(A, B)}{\text{W}(B, V)}, \tag{1.6}$$

where $\text{W}(A, V) = \sum_{i \in A, j \in V} w_{ij}$. The normalized cut criterion was first proposed by Shi and Malik and has been successfully used in image segmentation [71]. The theoretical attraction of the normalized cut lies in its analytical solution. The near-optimal solution

of the normalized cut can be obtained by solving the relaxed generalized eigensystem

$$(D - W)\mathbf{x} = \lambda D\mathbf{x}. \tag{1.7}$$

The second smallest eigenvector of this eigensystem is used to provide a linear search order for the optimal cut point. The normalized cut criterion can be applied to the symmetric similarity matrix representing a graph whose vertices are of the same type, as well as the rectangular matrix associated with a bipartite graph of different types of vertices. We apply the normalized cut method to the web graph partitioning problem, which is a very important data clustering task due to its large amount of information but unorganized structure. The clustering results are promising. We effectively locate all the topics in one large web graph, not just those dominant topics.

Although there was success in image segmentation and web graph partitioning, we found the normalized cut performed relatively poor in data clustering problems when the overlap between two clusters was large; it tended to cut off a small portion of the graph. This result prompted a search for better partition criteria which can effectively avoid this situation from happening. In the normalized cut form (1.6), the cut size $\mathrm{cut}(A, B)$ is weighed against the association between the subgraph and the entire graph. It is more meaningful to weigh the cut size against the self-association of the individual subgraph. This naturally leads to the following objective function:

$$\mathrm{Mcut}(A, B) \equiv \frac{\mathrm{cut}(A, B)}{\mathrm{W}(A, A)} + \frac{\mathrm{cut}(A, B)}{\mathrm{W}(B, B)} \tag{1.8}$$

where $W(A, A)$ is the self-association of the subgraph. This effort leads to the discovery of the spectral *min-max cut* (Mcut) criterion. The Mcut criterion is based on a min-max clustering principle: the similarity or association between two subgraphs (cut set) is minimized, while the similarity or association within each subgraph (summation of similarity between all pairs of nodes within a subgraph) is maximized. These two requirements can be satisfied simultaneously with a simple min-max cost function. The optimal solution of the min-max cut criterion can be approximated by solving the same generalized eigensystem as that for the normalized cut. The min-max cut criterion is proven to be less likely to cut off a small part of a graph than the normalized cut does and produces a more balanced partition.

Besides the spectral partitioning methods, other partitioning methods seek to minimize the sum of subgraph diameters [21] or k-center problem [1]. There are other clustering methods that use singular value decompositions [25]. For another view on segmentation problems, see [58].

1.4 Thesis outline

The remainder of the thesis is organized as follows. In §2, we discuss the popular spectral method, the *normalized cut*, and its application on web document clustering. We also give an insight into the relation between the normalized cut and the Cheeger constant and investigate the connection between the normalized cut and the K-means method.

In §3, we introduce the *min-max cut* criterion and provide the experimental results on the newsgroup data sets. Some refinement heuristics is also investigated in order to improve the initial partitioning results.

Many data mining applications can be modeled as bipartite graph partitioning problems. By minimizing either the normalized cut or min-max cut in the bipartite graph partitioning problems, the solution can be obtained by computing the singular vectors of the weight matrix associated with the bipartite graph. Taking this approach, we can cluster the data objects of the same type into groups while associating the group of one data type with one of the different data types. In §4, We apply both the normalized cut and the min-max cut criteria to a bipartite graph partitioning problem, attempting to group a set of news messages addressing the similar topics with corresponding words mostly used in these messages. The experimental results are presented to prove the effectiveness of this approach.

Based on the bipartite graph partition using the Mcut criterion, we investigate a new point of view on CA.

Correspondence analysis (CA) is a method in multivariate statistics to analyze contingency tables using a technique similar to principal component analysis (PCA). For most cases, CA projects the data into a $2 - dim$ space; the $2 - dim$ plot gives a clear graphical view of the data so that associations between different variables can be recognized easily.

Although CA can be derived from canonical correlations between two types of variables, the ordination point of view emphasized by Hill [47] based on gradient analysis

in ecology is interesting. But what is exactly the nature of the ordination on the same-type and different-type variables or objects?

In §5, we provide a new view of CA based on the cluster analysis of two-way contingency table data It is shown that CA is the direct results of clustering row and column objects simultaneously using a bi-clustering method based on the min-max cut graph partitioning algorithm.

Finally, we conclude the thesis and discuss the future research directions in §6.

Chapter 2

Normalized cut and web graph partition

Currently, the World Wide Web contains a huge number of documents and it is still growing rapidly. Finding the relevant documents to satisfy a user's information need is a very important and challenging task. Many commercial search engines have been developed and are used by millions of people all over the world. However, the relevancy of documents returned in search engine result sets is still lacking. Further research and development is needed to really make search engines an accurate information-seeking tool. The World Wide Web has a rich structure: it contains both textual web documents and the hyperlinks that connect them. The web documents and hyperlinks between them form a *directed* graph in which the web documents can be viewed as vertices and the hyperlinks as directed edges. Algorithms have been developed utilizing this directed graph to extract information contained in a collection of hyperlinked web documents. Kleinberg proposed the HITS algorithm based purely on hyperlink information to retrieve the most relevant information: *authority* and *hub* documents for a user query [56]. If the hypertext collection consists of several topics, however, authority and hub documents may only cover the most popular topics and leave out the less popular ones. One way to remedy this situation is to first partition the hypertext collection into topical groups, then present the search results as a list of topics to the user. This leads to the need to cluster the web documents based on both the textual and hyperlink information.

In this chapter, we apply a similarity-based clustering method to the problem of clustering web documents. It utilizes a graph-theoretic criterion called *normalized cut* which has its root in the study of graph isoperimetric problems [12, 15]. This method was proposed by Shi and Malik and has been successfully used in image segmentation [71].

Document clustering has been studied by many people [79]. Clustering retrieval results have been examined by Hearst and Paderson [42] based on textual information only, with emphasis on summarization. More recently, this approach is taken in the Grouper web interface [80]. In the context of clustering web documents, in addition to the textual contents of documents, many other sources of information can be effectively used to enhance clustering effectiveness, such as hyperlinks between documents, and co-citation (co-reference) patterns among documents. In our web document clustering approach, we incorporate information from hyperlink structure, co-citation patterns and textual contents of documents to construct a new similarity metric for measuring the topical homogeneity of web documents. Specifically, the hyperlink structure is used as the dominant factor in the similarity metric, and the textual contents are used to modulate the strength of each hyperlink. The similarity metric is used to construct a weighted graph which is then fed to the clustering method based on the normalized cut. This combination gives rise to a powerful method which effectively organizes the retrieved information against a user query and presents the organized information in a more accessible form for the user.

2.1 Organizing query result sets for search engines

The World Wide Web has grown to more than one billion unique web documents and continues to grow roughly at a rate of one million documents per day [6]. While the users are enjoying the large amount of information available on the World Wide Web, the sheer quantity of Web documents poses a problem to both the users and the search engines. When users submit a query, the search engine returns the retrieved web documents as the search result set. A typical query has 1-3 words with little query formulation [2, 16]. Sometimes the users are uncertain about their information needed when submitting the query [28], and for ordinary users, it is difficult to come up with query terms that accurately specify their information needs. These situations often result in broad-topic queries. Since many search engines retrieve the web documents based on the text similarity and link structures, after a broad-topic query, it is likely to produce a search result set with thousands, even up to millions of web documents. It is difficult to organize such a large number of documents in a manner that provides the information the user wants. It is often the case that for some broad-topic queries, the documents for the popular topic tend to dominate the result set. Under this circumstance, web document ranking algorithms such as HITS are unable to solve this problem. The authoritative documents returned to users by HITS may represent only the most popular topic. Documents related to under-represented topics have less chance to be returned to the users while perhaps they are what the users are expecting. To overcome this problem, before ranking the web documents and presenting them to users, it is necessary to group the

retrieved web documents into distinct topic areas, then return the ranked documents for each group according to their relevance.

The hyperlink structure of the World Wide Web provides us with rich information on web communities. It contains a lot of latent human annotation of the web society. This motivates us to cluster the web documents by partitioning the web link graph. If two web documents have very small text similarity, it is unlikely that they belong to the same topic, even if they are connected by a hyperlink. Therefore, to improve the quality of the graph representation, the text information can be incorporated into the link graph as a factor of edge weight. The co-citation is another relevance measure between two web documents based on how many other web documents create hyperlinks to both of them. For further improving the quality of the clustering results, combining the co-citation information into the link graph is also desirable.

For a clustering algorithm to be useful for the Web, it needs to be both effective and efficient. However, it is not necessary that the result sets be processed and the clusters be generated in real-time. Here, we are dealing with popular broad-topic queries, search engines can cache the result sets for those popular queries so that the processing can be done offline if necessary. We are currently investigating fast methods for spectral decomposition along the lines proposed in [34] to make our approach more suitable for real-time processing.

2.2 Similarity metric

One popular way for clustering data objects into subgroups is based on a similarity metric between objects, with the goal that objects within a subgroup are very similar,

and objects between different subgroups are less similar. In clustering a graph, similarity between nodes is represented by edge weight.

In the web documents clustering problem, we take a new approach in defining the similarity between two web documents. We incorporate link structure, textual information, and co-citation information into a similarity metric which gives rise to the weight matrix W. The link structure is the dominant factor, and the textual similarity is used to modulate the strength of each hyperlink. To further enhance the link structure, co-citation is also incorporated.

2.2.1 Hyperlink structure

The link information is obtained directly from the link graph. The method to form the link graph is introduced in section 2.6.1. Given a link graph $G = (V, E)$, which is directed, we define the matrix $A = (a_{ij})$ to be:

$$a_{ij} = \begin{cases} 1 & \text{if} \quad (i,j) \in E \quad \text{or} \quad (j,i) \in E \\ 0 & \text{otherwise.} \end{cases}$$

A is the adjacency matrix of the link graph where directionality of the hyperlinks is ignored. Link structure alone provides us with rich information on the topics of the document collection. By exploring the link structure, we are able to extract useful information from the web [9, 35, 10, 33, 59, 66, 60]. One of the most popular algorithms to extract document ranking information from the link structure is the HITS algorithm developed by Jon Kleinberg which will be briefly discussed in Appendix A.

2.2.2 Textual information

The textual information can be included to better cluster the web documents. Moreover, compared to printed literature, web documents reference each other more randomly. This is another reason that the text information is incorporated in order to regulate the influence of the document. One approach to incorporating the textual information is to measure the similarity between a user query and the anchor text (text between ⟨A HREF=...⟩ and ⟨/A⟩) as in [62, 10]. We experimented with this approach and found it did not work as effectively in our data sets as we had expected.

Here we use a new approach that (a) utilizes the entire text of a web document, not just the anchor text; (b) measures the textual similarity s_{ij} between two web documents i, j, instead of between the user query and the web document; (c) uses s_{ij} as the strength of the hyperlink between web documents i, j. The key observation here is that if two web documents have very little text similarity, it is unlikely that they belong to the same topic, even though they are connected by a hyperlink. Therefore s_{ij} properly gauges the extent or the importance of an individual hyperlink.

We represent each web document as a vector in the vector space model of IR (Information Retrieval) [76], then compute the *similarity* between them. The higher the similarity, the more likely the two documents deal with the same topic. For each element of the vector we use the standard $tf \cdot idf$ weighting: $tf(i,j) \cdot idf(i)$. $tf(i,j)$ is the Term Frequency of word i in document j, representing the number of occurrences of word i in

document j. *idf* is the Inverse Document Frequency corresponding to word i, defined as

$$\text{idf}(i) = \log(\frac{\text{no. of total docs}}{\text{no. of docs containing word } i}).$$

Some words appear too frequently in many documents. We assume these words are not very useful to identify the documents. Inverse Document Frequency can effectively decrease the influence of these words.

Since the term vector lengths of the documents vary, we use cosine normalization defined in (1.1) to compute similarity.

The similarities between documents form the similarity matrix S.

2.2.3 Co-citation patterns

Co-citation is another metric to measure the relevance of two web documents. If there are many documents pointing to both of them, then these two documents are likely to address a similar issue. The co-citation pattern is used by H. Small and others to trace and map scientific literatures [73]. The co-citation c_{ij} of documents i and j is the number of web documents pointing to both i and j.

Incorporating the above information into the similarity metric, we form the weight matrix of the graph:

$$W = \alpha\frac{A \otimes S}{\|A \otimes S\|_2} + (1-\alpha)\frac{C}{\|C\|_2}, \tag{2.1}$$

where A is the adjacency matrix of the link graph. S is the similarity matrix. C is the co-citation matrix and $(A \otimes S)_{ij} = a_{ij}s_{ij}$. The meaning of notation $\otimes$ applies to the

rest of this chapter. α is a real value between 0 and 1. The algorithms we introduce will be applied to matrix W.

The link and text information is retrieved *a priori*. The adjacency matrix A is formed by sweeping all the link graph edges once, requiring $O(|E|)$ time. Since each web document has an average out-degree (the number of links originated from it) around 8 [57, 59], on average there are 28 pairs of documents pointed to by a web document. It takes about $28|V|$ operations to compute the co-citation matrix C, requiring $O(|V|)$ time. Note that we only need the similarity of two documents which have a link between them. Suppose the average number of words in each document is N_w, then the running time to compute the similarity matrix S is $O(N_w|E|)$. Thus the total running time for computing the weight matrix W is $O(N_w|E| + |V|)$.

2.3 Partitioning method based on normalized cut

After forming the weight matrix W representing the web graph as in section 2.2, we can cluster the web documents into distinct topic areas by applying some existing graph partitioning methods on W. The criteria used to measure the goodness of the partition will largely determine the final quality of the partition results. In graph theory, a set of edges that separates the graph into two disconnected parts is called an edge-cut (edge separator) of the graph. The standard minimum cut algorithms minimize the size of edge-cut alone. This often causes an unbalanced partition: it may cut a portion of a graph with a small number of vertices. In the context of graph clustering, this is in general not desirable. To avoid partitioning out a small part of a graph G by using edge-cut alone, many criteria utilize various normalized forms of edge-cut which are generally

obtained by dividing the edge-cut by some measure of the size of the partitions. The *normalized cut* is one such measure.

2.3.1 Normalized cut

Given an undirected graph $G = (V, E)$ with edge weight matrix W, let S be a subset of V and $\bar{S} = V - S$, the complement of S in V. The normalized cut (1.6) can be rewritten as:

$$\text{Ncut}(S, \bar{S}) = \frac{\text{cut}(S, \bar{S})}{\text{W}(S, V)} + \frac{\text{cut}(S, \bar{S})}{\text{W}(\bar{S}, V)} \tag{2.2}$$

where $\text{cut}(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} w_{ij}$, $\text{W}(S, V) = \sum_{i \in S, j \in V} w_{ij}$. The partitioning problem here is to find the partitions that minimize the normalized cut $\text{Ncut}(S, \bar{S})$. If we let D be the diagonal matrix with $d_{ii} = \sum_{k=1}^{n} w_{ik}, i = 1, \cdots, n$, the minimization problem associated with normalized cut has been shown to be equivalent to the following discrete minimization problem [71]:

$$\text{Ncut}(\mathbf{x}) = \frac{\mathbf{x}^T (D - W)\mathbf{x}}{\mathbf{x}^T D \mathbf{x}}, \tag{2.3}$$

subject to the constraint that $x_i \in \{1, -b\}$ and $\mathbf{x}^T D\mathbf{e} = 0$ where $\mathbf{e}$ is a vector with all elements equal to 1, and b is positive. It is the Rayleigh quotient associated with the generalized eigensystem (1.7). Relaxing the condition $x_i \in \{1, -b\}$ and allowing the elements of $\mathbf{x}$ to take any real values, we can easily see that (2.3) can be minimized by the second smallest eigenvector of (1.7).

With the introduction of the normalized-cut criterion, we can easily apply it to measure web graph partitioning problems. It effectively avoids the problem of cutting a small part of the graph. Notice that letting $\mathbf{y} = D^{1/2}\mathbf{x}$, (1.7) can be transformed to

$$D^{-1/2}WD^{-1/2}\mathbf{y} = (1-\lambda)\mathbf{y}. \tag{2.4}$$

Therefore, we just need to compute the second largest eigenvector of $D^{-1/2}WD^{-1/2}$.

The normalized cut method is a further development in the class of spectral graph partitioning methods.

We use the second smallest eigenvector of the generalized eigensystem (1.7) to partition the graph. Clearly the normalized cut method is a spectral graph partitioning method with the removal of the constraint $|S| = |\bar{S}|$. Here we call the second smallest eigenvector of (1.7) the scaled Fiedler vector. One interesting property of the scaled Fiedler vector is its relation to algebraic connectivity: [1]

THEOREM 2.1. *Given connected graph $G(V, E)$. Let $\mathbf{f}$ be the eigenvector associated with the second smallest eigenvalue of (1.7). Define $V_1 = \{v \in V : f_v \leq 0\}$, then the subgraph induced by V_1 is connected. Similarly, define $V_2 = \{v \in V : f_v \geq 0\}$, then the subgraph induced by V_2 is also connected.*

REMARK. The above theorem implies that if the original graph is connected, after partitioning using the scaled Fiedler vector, the subgraphs obtained are also connected if the scaled Fiedler vector has no zero entries. But in general this conclusion is not true for the K-means method which we will discuss later.

[1] The proof of the theorem follows similar arguments as in Theorem 2.1 in [68].

Now we give an illustrative example of using the scaled Fiedler vector to partition the graph. Figure 2.1(a) is a weight (adjacency) matrix generated by Matlab. The first diagonal block is 100-by-100, and the second diagonal block is 200-by-200. These two blocks represent two highly connected subgraphs. The off-diagonal blocks represent sparse connectivity between the two subgraphs. The adjacency matrix is symmetric with diagonal elements set to zero. Figure 2.1(b) shows the plot of the scaled Fiedler vector of the generalized eigensystem corresponding to this matrix. From the plot, we can easily see that the scaled Fiedler vector cuts the nodes of the matrix into two distinct parts, if we choose zero as the cutting point.

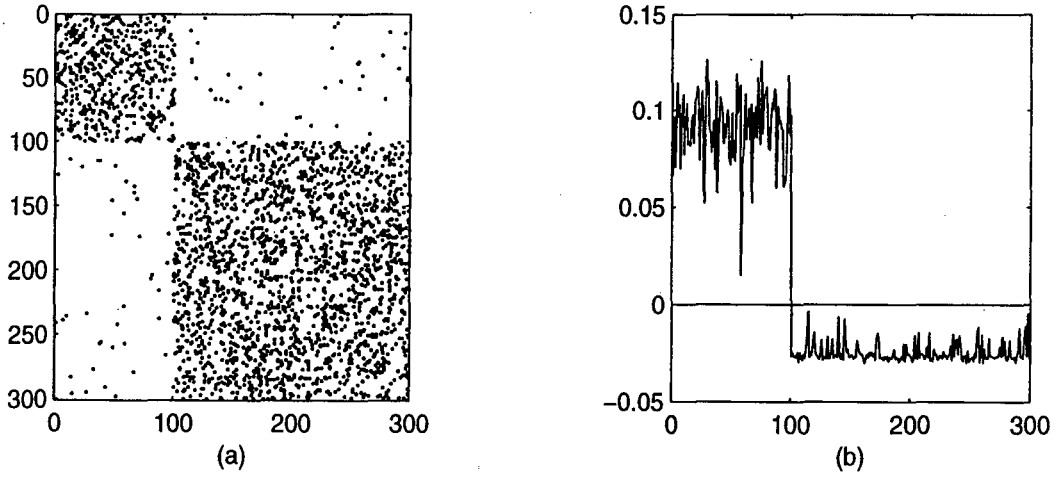

Fig. 2.1. (a) The weight matrix of a graph. (b) The scaled Fiedler vector values on each node.

2.3.2 Hierarchical divisive clustering

In this algorithm, we partition the weighted graph with the scaled Fiedler vector and normalized cut. After the scaled Fiedler vector $\mathbf{f}$ of the generalized eigensystem (1.7) is available, we sort the vertices of G based on their corresponding values in $\mathbf{f}$ and check all possible cutting points for the best partitioning which we define as the one with the smallest normalized cut. We then decide whether to accept or reject the partitioning based on the smallest normalized cut and pre-defined threshold. If the smallest normalized cut computed is below the threshold, we accept the partitioning. If the smallest normalized cut is above the threshold, which means there are more connections between these two subgraphs, then we reject the partitioning and consider them as one cluster. This algorithm is shown below:

Given a weighted graph $G = (V, E)$, its weight matrix is W.

1. $\mathbf{d} = W\mathbf{e}$, where $\mathbf{e} = (1, 1, \cdots, 1)^T$. $D = \text{Diag}(\mathbf{d})$.

2. Solve the generalized eigensystem (1.7) for the scaled Fiedler vector $\mathbf{f}$.

3. Sort the vertices of G based on their corresponding values in $\mathbf{f}$.

4. Check every possible cutting point in the order of the sorted vertices, find the one with the smallest normalized cut.

5. If the normalized cut is below a certain threshold, accept the partition and recursively partition the subgraphs. Otherwise, stop.

Computational complexity. The running time for this algorithm depends on the number of iterations. Here we analyze the running time for one iteration only. To

simplify the notation, we still use E and V to represent the edge set and node set of the graph to be partitioned in each recursion step. Because the running time is dominated by the eigensolver and the sorting procedure, here we consider the running time of steps 2 and 3 only. Using Lanczos method to compute the scaled Fiedler vector $\mathbf{f}$ of (1.7) has complexity roughly proportional to $\text{nnz}(D - W)$, the number of non-zero elements of $D - W$, which is $O(|E|)$. If the number of Lanczos iteration steps is N_r, then the running time for step 2 is $O(N_r|E|)$. In step 3, the sorting of $\mathbf{f}$ takes $O(|V|log(|V|))$. Thus the running time for one recursion is $O(N_r|E| + |V|log(|V|))$. Note that $|V|$ and $|E|$ become smaller in each subgraph, so the running time is smaller for each recursion as the recursion level goes deeper.

The number of recursion is bounded by the number of clusters we finally obtained, which is very small comparing with $|E|$ and $|V|$, hence the total time for the partition is $O(N_r|E| + |V|log(|V|))$.

2.4 Connection with the Cheeger constant

Rewrite (2.2) as

$$\begin{aligned} \text{Ncut}(S, \bar{S}) &= \frac{\text{cut}(S, \bar{S})}{\min(\text{W}(S, V), \text{W}(\bar{S}, V))} + \frac{\text{cut}(S, \bar{S})}{\max(\text{W}(S, V), \text{W}(\bar{S}, V))} \\ &= h_G(S) + \frac{\text{cut}(S, \bar{S})}{\max(\text{W}(S, V), \text{W}(\bar{S}, V))} \end{aligned}$$

where

$$h_G(S) = \frac{\text{cut}(S, \bar{S})}{\min(\text{W}(S, V), \text{W}(\bar{S}, V))}.$$

The *Cheeger constant* h_G of graph G is defined as:

$$h_G = \min_S h_G(S).$$

The Cheeger constant is one of the earliest normalized forms of edge-cut measuring the quality of the graph partition. It was studied by J. Cheeger in early 1970s [12]. The normalized cut can be regarded as a slight variation of the Cheeger constant.

Defining a constant h_{ncut} for the normalized cut cost function (2.2) as $h_{ncut} = \min_S \mathrm{Ncut}(S, \bar{S})$, we show that the upper and lower bounds for the Cheeger constant h_G can be expressed in terms of h_{ncut}:

$$h_{ncut} \geq h_G \geq \frac{h_{ncut}}{2}. \tag{2.5}$$

Clearly we have $h_{ncut} \geq h_G$. On the other hand,

$$h_G(S) \geq \frac{1}{2}\left(\frac{\mathrm{cut}(S, \bar{S})}{\mathrm{W}(S, V)} + \frac{\mathrm{cut}(S, \bar{S})}{\mathrm{W}(\bar{S}, V)}\right) = \mathrm{Ncut}(S, \bar{S})/2,$$

which implies $h_G \geq \frac{h_{ncut}}{2}$. The inequality (2.5) shows the close relationship between the Cheeger constant and the normalized cut.

2.5 Connection with the K-means method

In this section, we explore the connection of the normalized-cut based clustering method and the popular K-means clustering method. For ease of discussion, we will only

consider the case where the weight matrix W satisfies $w_{ij} \in \{0, 1\}$. Similar conclusions can be drawn for the more general cases.

2.5.1 K-means method

The general idea of the K-means method is the following: 1) partition the nodes into two arbitrary clusters; 2) for each node re-assign it to the cluster that the node is in some sense closest to; repeat the above two steps until convergence, i.e., until when the cluster membership of no node will change by running through the two steps. One natural and reasonable heuristics for measuring the closeness of a node to a cluster is the following: for a node k, we examine all the nodes that are adjacent to k, if there are more neighbors of k belonging to cluster one, we then re-assign i to cluster one, otherwise we re-assign k to cluster two. It is easy to see that this approach can be generalized to the multiple cluster case: we re-assign i to the cluster in which i has the biggest number of neighbors. If we denote the cluster assignment of k as $x_k \in \{-1, 1\}$, then

$$\sum_{j=1}^{n} w_{kj} x_j$$

is the difference in the numbers of neighbors of k belonging to the two clusters. Therefore, the re-assignment of k can be computed as

$$x_k^{(i)} = \text{sign}(\sum_{j=1}^{n} w_{kj} x_j^{(i-1)}),$$

where i denotes the iteration step. Rewriting in matrix-vector format, we have

$$\mathbf{x}^{(i)} = \text{sign}(W\mathbf{x}^{(i-1)}).$$

As will be discussed in more detail later, the above iteration can be considered as a variation of the power method for computing the largest eigenpair of a symmetric matrix [36], the difference being that a different normalization process is used here. Restricting $x_k \in \{-1, 1\}$ corresponds to hard-clustering: a node either belongs to cluster one or cluster two. For soft-clustering, i.e., we can consider $x_k \in [0, 1]$ as the probability of node k belonging to cluster one, and use normalization

$$\mathbf{y}^{(i)} = W\mathbf{x}^{(i-1)}, \quad \mathbf{x}^{(i)} = \mathbf{y}^{(i)}/\|\mathbf{y}^{(i)}\|_2.$$

This corresponds exactly to the power method for W. In the above discussion we only considered the neighbors of a node k. We can also turn the heuristic argument around: for a node k, we examine all the nodes that are *not* adjacent to k, if there are more of these belonging to cluster one, we then re-assign i to cluster two, otherwise we re-assign k to cluster one. These two arguments can certainly be combined, and we obtain the following re-assignment rule

$$x_k^{(i)} = \text{sign}(\sum_{j\sim k}^{n} x_j^{(i-1)} - \alpha \sum_{j!\sim k}^{n} x_j^{(i-1)}) \tag{2.6}$$

where $\alpha > 0$ is a balancing factor. Here we also use $j \sim k$ to denote that node j is adjacent to node k and $j! \sim k$ to denote that node j is not adjacent to node k.

2.5.2 Connection between normalized cut and K-means method

The scaled Fiedler vector $\mathbf{f}$ of (1.7) can be obtained by first computing the second largest eigenvector $\mathbf{y}_2$ of (2.4). We know that $D^{1/2}\mathbf{e}$ is the eigenvector of $D^{-1/2}WD^{-1/2}$ corresponding to its largest eigenvalue $\lambda_1 = 1$. [2] The second largest eigenvector $\mathbf{y}_2$ can be obtained by computing the largest eigenpair of

$$\hat{W} = D^{-1/2}WD^{-1/2} - \frac{D^{1/2}\mathbf{e}\mathbf{e}^T D^{1/2}}{\mathbf{e}^T D\mathbf{e}}. \tag{2.7}$$

One approach for computing the largest eigenpair of a symmetric matrix $\hat{W}$ is the power method [36]:

> Start with a unit vector $\mathbf{x}^{(0)}$. For $i = 1, 2, \ldots$, until the result converges
>
> $\mathbf{y}^{(i)} = \hat{W}\mathbf{x}^{(i-1)}$
>
> $\mathbf{x}^{(i)} = \mathbf{y}^{(i)}/\|\mathbf{y}^{(i)}\|_2$
>
> At the end of convergence, an approximate eigenvalue can be
>
> computed as $\hat{\lambda} = (\mathbf{x}^{(i)})^T \hat{W}\mathbf{x}^{(i)}$.

Applying the power method on $\hat{W}$, we get the largest eigenvector $\mathbf{y}_2$ of $\hat{W}$ which is the second largest eigenvector of $D^{-1/2}WD^{-1/2}$. Then the scaled Fiedler vector $\mathbf{f}$ of (1.7) is obtained by $\mathbf{f} = D^{-1/2}\mathbf{y}_2$.

---

[2] It can be readily proved that all the eigenvalues of $D^{-1/2}WD^{-1/2}$ are at most 1. For the proof, see Appendix C.(1)

Apply the power method to the matrix in (2.7), and let the k-th entry of $\mathbf{x}^{(i)}$ be $x_k^{(i)}$. We have

$$y_k^{(i)} = \sum_{j=1}^{n} \frac{w_{kj} x_j^{(i)}}{\sqrt{d_k d_j}} - \frac{\sqrt{d_k d_j} x_j^{(i)}}{d_s},$$

where $d_s = \mathbf{e}^T \mathbf{d}$, the sum of the diagonal elements of D. We can rewrite the above equation as

$$y_k^{(i)} = \sum_{j \sim k} \left(\frac{x_j^{(i)}}{\sqrt{d_k d_j}} - \frac{\sqrt{d_k d_j} x_j^{(i)}}{d_s} \right) - \sum_{j ! \sim k} \frac{\sqrt{d_k d_j} x_j^{(i)}}{d_s}.$$

Now comparing the above with the K-means iteration in (2.6), we notice the following: for the neighbors of node k, the normalized-cut method uses the modified weight

$$\frac{1}{\sqrt{d_k d_j}} - \frac{\sqrt{d_k d_j}}{d_s},$$

and for those nodes not adjacent to k, normalized-cut uses

$$\frac{\sqrt{d_k d_j}}{d_s}$$

as the weight. Therefore, the normalized-cut method can be considered as a variation of the K-means method.

2.6 Experiments

We test the algorithm on the link graphs of queries *amazon, star* and *apple.* We choose these three query terms because they all have multiple distinct meanings. *Amazon* has at least three meanings. One is related to amazon.com, one of the largest on-line shopping web sites. Another is the famous rain forest in South America. The third is the name of ancient female warriors from Alecto, a female ruled monarchy. For the query *star*, we can think about a natural luminous body visible in the sky, a movie star, a famous athlete and the movie *Star Wars.* Mentioned *apple*, we will associate it with a kind of fruit or the apple computer.

We then apply our clustering algorithm on the data sets. Since each cluster often has a large number of web documents, we choose only the most important web documents among a cluster. The most important or authoritative web documents in each cluster are determined using the HITS algorithm introduced in Appendix A.

We also compare the results obtained from our algorithm to the results by recursively applying simple K-means based algorithm. As our results will show, the normalized-cut based technique performs better that the K-means based algorithm. The stop criteria for the K-means algorithm can be either the maximal iteration number reached, the result is stable, or the cluster size is below a certain threshold. In our implementation, we choose a threshold on maximal iteration number to be our stop criteria. The K-means method is applied recursively to each subgraph obtained until the size of the subgraph is below certain threshold.

2.6.1 Data preparation

To obtain the link graphs, we first provide a text-based search engine *hotbot* with query terms. *Hotbot* returns as the query result a list of URLs with highest ranked web documents. We limit the number of returned URLs to be 120 which form the root set. By limiting the number of returned URLs to be 120, we can keep the overall data set within a reasonable size. Then we expand the root set by including all web documents that point to a web document in the root set and those pointed to by a web document in the root set. This is a level one expansion. The link graph can be viewed as a directed graph. It's easy to convert it to the adjacency matrix of an undirected graph.

After the full list of URLs is available, we run a web crawler to get the text information of these web documents. The text of a document is obtained using a web crawler that we write in *Perl*. To accommodate the vast differences in web document lengths, we limit the length of each document to be 500 words. The rest of the document is discarded if it has more than 500 words. Stopwords (such as *I, is, a*, etc.) are discarded using a standard list. Words are stemmed using Porter stemming [67], so the words *linking* and *linked* are both stemmed to the same root *link*. Once the text information of all documents is available, we compute the document similarity as in section 2.2.2.

At this point, we have the necessary data for the experiment. In our experiment, we set α in expression (2.1) to be 0.5.

2.6.2 Clustering results

We apply our algorithm on the data sets of three query terms with the threshold of normalized cut set to 0.06, then run the HITS algorithm on each cluster obtained.

The top authorities of each significant clusters are listed. By significant we mean the size of the cluster is not too small.

The order of the clusters has no special meanings. It is merely the sequence by which we process the clusters.

2.6.2.1 Query term: *amazon*

There are a total of 2294 URLs in this data set. Applying our algorithm, we obtain the following significant clusters. The clusters with small size are not counted. Some small clusters exist as the result of the algorithm because they form the connected components of the link graph. It shows that our algorithm has the capability to find the cluster of small size but tightly connected.

Cluster 1:

- *www.amazon.com/*
- *www.amazon.co.uk/*
- *www.amazon.de*

These three web documents are the home pages of amazon.com, one of the most famous shopping companies in the world. The first website is located in the USA, the second in the UK and the third in Germany. It makes sense to cluster them together. We list only three authorities here because the rest of the documents ranked among the top 10 have very very low authority weights compared with the first three, so we don't list them here.

Cluster 2:

- *www.amazoncity.com/*
- *www.amazoncityradio.com/*
- *www.amazoncity.com/spiderwoman/*
- *radio.amazoncity.com/*
- *www.wired.com/news/news/culture/story/6751.html*

This cluster is about the women warriors.

Cluster 3:

- *www.amazon.org/*
- *www.amazonfembks.com/*
- *www.igc.apc.org/women/bookstores/*
- *www.teleport.com/~rocky/queer.shtml*
- *www.advocate.com/html/gaylinks/resources.html*

The topic of this cluster is on women's issues: bi-sexuality and lesbian books.

Cluster 4:

- *sothebys.amazon.com/exec/varzea/tg/special-sales/...*
- *sothebys.amazon.com/exec/varzea/subst/home/sothebys.html/...*
- *sothebys.amazon.com/exec/varzea/tg/special-sales/...*
- *s1.amazon.com/exec/varzea/subst/home/home.html/...*

- *sothebys.amazon.com/exec/varzea/subst/home/sothebys.html/...*

All five authorities listed here are web documents of a large on-line auction company formed by Sothebys and amazon.com. It's not clustered in to Cluster 1 because there are relatively few links between them.

Cluster 5:

- *www.swalliance.com/*
- *timeline.echostation.com*
- *www.echostation.com:8080/~1*
- *downtime.echostation.com*
- *rpg.echostation.com/*

The topic of this cluster is about *Star Wars*, surprisingly. But there are a total of 68 documents on this topic, most created by *Star Wars* fans. Some are on-line shopping companies selling goods related to the movie *Star Wars.*

There are two clusters, but for each of them, the web documents are from the same site: *www.langenberg.com* and *www.latingrocer.com*, respectively. These two clusters are:

Cluster 6:

- *misc.langenberg.com/*
- *cooking.langenberg.com/*
- *shipping.langenberg.com/*
- *money.langenberg.com/*

- *weather.langenberg.com/*

and

Cluster 7:

- *www.latingrocer2.com*
- *www.latingrocer.com*
- *www.latingrocer.com/Pages/customer.html*
- *www.latingrocer.com/Pages/privacy.html*
- *www.latingrocer.com/Pages/contact.html*

Cluster 8:

- *www.internext.com.br/ariau*

This cluster has only one authority. Other web documents in this cluster all point to it. It's a website of a hotel in the Amazon River valley.

After applying the K-means based algorithm on the data, we also get several clusters. Cluster 1 is separated into three clusters, that is, the websites in the three different countries form three clusters under this algorithm. Second, there are more clusters with small size than the result from our algorithm. Many of them should belong to larger clusters, but were partitioned incorrectly.

By adjusting the threshold, the algorithm can group or separate the following web documents:

- *www.amazon.com/*

- *www.amazon.co.uk/* and
- *www.amazon.de*

while the K-means based algorithm can only partition them into different clusters. This verifies that our algorithm is not only more flexible than the other, but also clusters more reasonably.

From the clusters we obtained, we found that, unlike what we had expected, no cluster has a focused topic on the rain forest while Cluster 8 does have something to do with the rain forest in Brazil. Checking the entire data set, we only found a couple of web documents that mention the rain forest. They don't form a cluster with significant size. If we return to *hotbot* and enter the query *amazon*, the web sites presented are dominantly related to amazon.com. The documents about the rain forest are not among the highest-ranked list returned by *hotbot*. It means the number of web documents on amazon the rain forest is small, which makes the web documents about this topic have low rank weight. That's the reason that we can't have a cluster on this topic.

As for the third meaning of amazon, although female warrior dose not directly appear as a distinct topic in any cluster, Cluster 3 focused on women's issues, or even the bi-sexual issue. By examining the content of these documents, we are sure that these issues are the extent of the original meaning of amazon as female warriors.

There are clusters with small size that apparently have nothing to do with *amazon*, such as the clusters about websites langenberg.com and latingrocer.com. They are explored as separate clusters because the web documents on the same site point to each other, raising the importance themselves. To avoid such situations, before applying any

clustering algorithm, we can coarsen the link graph first, so that the web documents from the same site collapse to one node in the graph.

2.6.2.2 Query term: *star*

We have 3504 URLs for this query. Setting the threshold to be 0.06, we run our algorithm on the data set of query term *star*. The authorities of each cluster are listed below:

Cluster 1:

- *www.starwars.com/*
- *www.lucasarts.com/*
- *www.sirstevesguide.com/*
- *www.jediknight.net/*
- *www.surfthe.net/swma/*

This cluster is focused on *star wars*.

Cluster 2:

- *www.kcstar.com/*
- *www.dailystarnews.com/*
- *www.kansascity.com/*
- *www.starbulletin.com/*
- *www.trib.com/*

This cluster includes the web documents of some news media with the word *star* as part of their names.

Cluster 3:

- *www.weatherpoint.com/starnews*
- *www.starnews.com/digest/sports.html*
- *www.starnews.com/digest/citystate.html*
- *www.indy.com*
- *speednet.starnews.com/*

The top authorities are all web documents of starnews.com in this cluster.

Cluster 4:

- *www.state.mn.us/mainmenu.html*
- *www.mda.state.mn.us/*
- *www.doli.state.mn.us/*
- *www.legalethics.com/pa/states/state/mn.htm*
- *www.exploreminnesota.com*

This cluster's topic is the state of Minnesota. The reason that Minnesota is a topic of the cluster under query *star* is because the official State of Minnesota web site is called *North Star*, which is named second among all government web sites.

Cluster 5:

- *www.star-telegram.com/*
- *www.dfw.com/*
- *www.virtualtexan.com/*
- *marketplace.dfw.com*
- *www.star-telegram.com/advertise/vshops/*

A cluster of star-telegram.com located in Texas.

Cluster 6:

- *www.starpages.net/*

There is only one authority in this cluster. This authority is a web site introducing preeminent persons in various fields, such as movie star, sport stars, etc. This cluster is what we have expected to be obtained after partitioning.

Cluster 7:

- *www.aavso.org/*
- *www.astro.wisc.edu/ dolan/constellations/*
- *ourworld.compuserve.com/homepages/rawhide_home_page*
- *adc.gsfc.nasa.gov/adc/adc_amateurs.html*
- *heasarc.gsfc.nasa.gov/docs/www_info/webstars.html*

This cluster talks about space and astronomy. It's also what we have expected. Now all three meanings of *star* are found to be topics of separate clusters.

Clusters 2, 3, 5 are all web documents of news media. They are partitioned to be different clusters since there is no link among the three clusters.

Using the K-means based algorithm, we obtained many more clusters than obtained by our algorithm. The same as the result of the query *amazon* under the K-means based algorithm, many of them are not really clusters with focused topics. On the other hand, Clusters 2 and 4 are grouped together, although there is no link between them. This is because the K-means based algorithm is not global. It's easy to be trapped to a local minimum while the first algorithm avoids this situation by using the scaled Fiedler vector to partition.

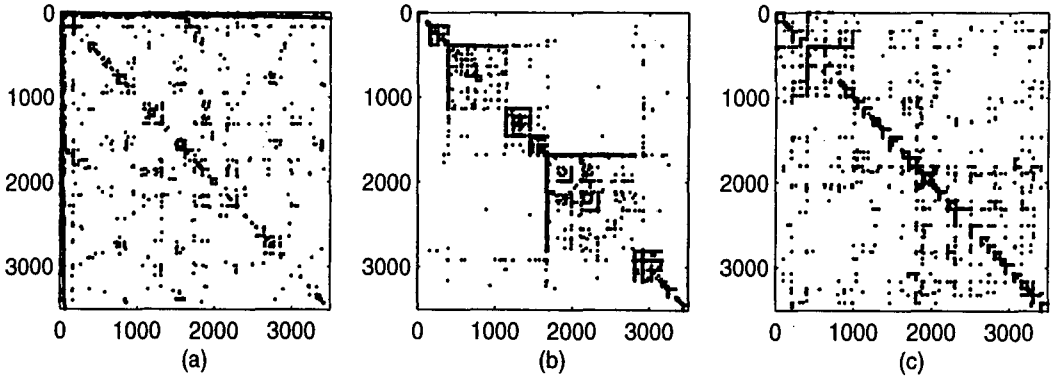

Fig. 2.2. Clustering result of query *star*: (a) weighted link graph of *star*; (b) clustering result using our algorithm, with threshold 0.06; (c) clustering result using K-means based algorithm.

In Figure 2.2, graph (a) is the original weighted link graph of data for the query *star*. (b) is the re-ordered graph after clustering using our algorithm. (c) is the re-ordered

graph after clustering using the K-means based algorithm. In graph (b) and (c), each diagonal block corresponds to a resulting cluster. Comparing graphs (b) and (c), we can see that, in graph (b), the off-diagonal blocks are much sparser than off-diagonal blocks in graph (c). This means our algorithm creates clusters with much fewer connections between them than the K-means based algorithm does.

In Figure 2.3, graph (a) is the original weighted link graph of data for the query *star*. (b) is the re-ordered graph after clustering using our algorithm with the threshold equal to 0.06. (c) is the re-ordered graph after clustering using our algorithm with the threshold equal to 0.1. (d) is the re-ordered graph after clustering using our algorithm with the threshold equal to 0.2. The graph clearly shows that by increasing the threshold, we can obtain more clusters with the size of each cluster smaller. In this way, we can control the clustering result by changing the value of the threshold in the algorithm. It makes this algorithm more flexible than the K-means based one.

2.6.2.3 Query term: *apple*

In this data set, there are 2757 URLs returned by the search engine. The threshold is still 0.06. The authorities of each cluster are listed below after running the algorithm on the data of the query term *apple*:

Cluster 1:

- *www.apple.com/*
- *www.apple.com/support/*
- *www.apple.com/education/*

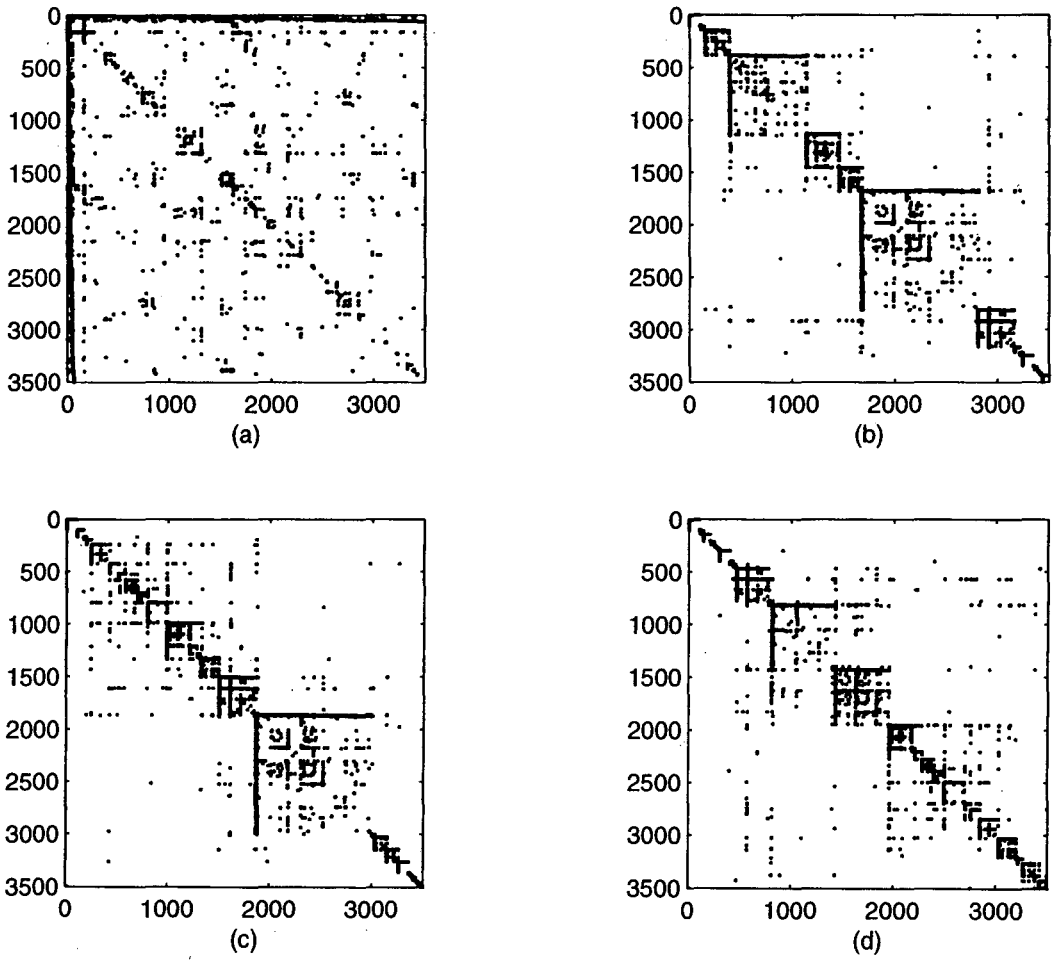

Fig. 2.3. Clustering result of query *star* using our algorithm: (a) weighted link graph of *star*; (b) clustering result with threshold 0.06; (c) clustering result with threshold 0.1; (d) clustering result with threshold 0.2.

- *www.apple.com/quicktime/*
- *www.apple.com/hotnews/*

Here all top authorities are from the same web site: *www.apple.com*. This cluster is dominant in this query. Most URLs belong to it. After running the HITS algorithm with the site information considered, that is, the URLs from the same web site are collapsed to one node, we obtain different top authorities:

- *www.apple.com/powermac/server/*
- *www.claris.com/*
- *www.apple.ru/hardware/displays*
- *www.cs.brandeis.edu/~xray/oldmac.html*
- *www.next.com/*

The second URL in this cluster is the website of a computer software company. It produces software used for the Macintosh. The other four URLs are all about the apple computer. This list of authorities provides more useful information than the previous one does.

Cluster 2:

- *www.yabloko.ru/*
- *www.cityline.ru/politika/*
- *www.russ.ru/*

- *www.forum.msk.ru/*
- *www.novayagazeta.ru*

All web documents in this cluster are written in Russian.

Cluster 3:

- *www.michiganapples.com/*

This cluster has only one authority. It's related to apple, the fruit.

The following cluster is formed in this query simply because its name happens to contain the query term *apple*:

Cluster 4:

- *www.ci.apple-valley.mn.us/*

which is the website of the city of Apple Valley, MN.

Cluster 5:

- *www.valleyweb.com/*

This is the website of Annapolis Valley in Canada where there is the Apple Blossom Festival in the spring celebrating the traditions and agricultural heritage.

There are not many clusters formed by the algorithm, nor do we find very interesting topics which are beyond our expectation.

The performance of the K-means algorithm is mixed. On one hand it doesn't form a cluster as large as Cluster 1. For example, it forms a new cluster around *www.france.euro.apple.com/* which belongs to Cluster 1 obtained with our algorithm. On the other hand, it groups

totally different web documents together. For instance, the top authorities of one cluster are:

- *www.jokewallpaper.com/*
- *the-tech.mit.edu/Macmade/*
- *www.geocities.com/SiliconValley/Vista/7184/guitool.html*

But *www.jokewallpaper.com/* and *the-tech.mit.edu/Macmade/* mention totally different things.

2.7 Further discussion

2.7.1 Importance of text information

Intuitively, the links in the webgraph can not be regarded as equally important, specifically many web links are created much less carefully than the references in scientific literature. The text similarity between two web documents provide us with a useful metric to address this issue. Without the similarity as the measure of the link strength, that is, if we form the weight matrix as:

$$W = \alpha \frac{A}{\|A\|_2} + (1-\alpha)\frac{C}{\|C\|_2},$$

then we unduly raise the strength of some links which connect two web documents with little in common. When applied to the data set *amazon*, our algorithm groups Clusters 1, 3 and 4 into one single cluster, using the same threshold as the stopping criterion, namely 0.06 in our previous experiments. Cluster 3 addresses the female issues and

Clusters 1 and 4 are related to *amazon.com*. Apparently this clustering result is not a good one. This result justifies our choice of incorporating the text information into the weight metric.

Recall that our weight matrix is formed as:

$$W = \alpha \frac{A \otimes S}{\|A \otimes S\|_2} + (1 - \alpha) \frac{C}{\|C\|_2}.$$

If the value of α changes, the weight matrix will change accordingly. Then we can set different thresholds in the algorithm to have three authoritative web documents in Cluster 1 of the data set *amazon* separated into different clusters. Table 2.1 lists the α value and the corresponding threshold T such that these three web documents are separated by the algorithm. From the (α, T) pairs in the table, we see clearly that when the value of α increases, to partition these three web documents into different clusters, the threshold decreases monotonically.

α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
T	1.1	1.0	0.8	0.7	0.6	0.5	0.5	0.4	0.3	0.3

Table 2.1.
(α, T) pairs that separate three authoritative web documents in Cluster 1 of query *amazon*.

2.7.2 Scaled Fiedler vector

Now we use a concrete example to show that partitioning a graph with the scaled Fiedler vector and normalized cut is feasible. Here we use the subgraph which includes URLs in Clusters 1, 3 and 4 of the data set *amazon*. There are a total of 1839 web documents in this subgraph. From the discussion above, we know that this subgraph contains multiple topics and can be produced by using the weight matrix without the text information. Figure 2.4(top) is the sorted scaled Fiedler vector of the subg raph. The horizontal axis is the node index and the vertical axis is the corresponding entry of the scaled Fiedler vector. Figure 2.4(bottom) is the node index and corresponding normalized cut value. We see that the sorted scaled Fiedler vector behaves like step function. Each segment takes nearly constant value. The jumping points can be used as the cutting points because they correspond to small normalized cut values. See Figure 2.4(bottom) for their values. After we have determined the threshold, we can obtain the cutting points, and partition the graph into clusters. If we set the threshold to be 0.06, we obtain the following cutting points:

(57 58)

(106 107 108 109 110 111 112 113 121 124 125 127 128 129 130 131)

(1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1731 1732 1733 1734 1735 1739 1740 1741 1742)

(1820 1823)

The numbers in the same pair of () are almost consecutive, we can choose any one of them as the cutting point. This won't affect the resulting clusters significantly since

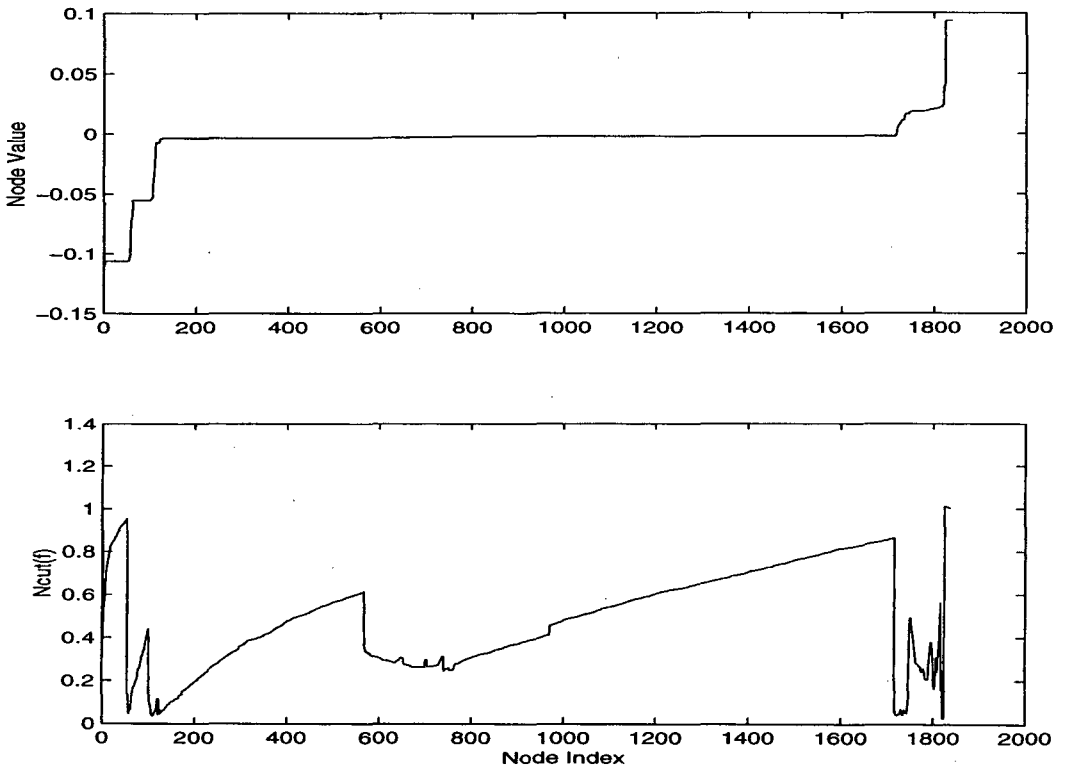

Fig. 2.4. The sorted scaled Fiedler vector value (top) and corresponding normalized cut (bottom).

these nodes correspond to URLs which are not important, i.e., not good authorities nor good hubs.

If we enlarge Figure 2.4 to include only the first 200 nodes, we can see the result more clearly in Figure 2.5.

For convenience, we choose the cutting points which correspond to the smallest normalized cut in each group: 58 110 1721 1820. This gives us 5 groups:

1. 1 – 57: with top authority *amazon.org*
2. 58 – 109: with top authority *fembks.com*
3. 110 – 1720: corresponds to Cluster 1 of *amazon*
4. 1721 – 1819: corresponds to Cluster 4 of *amazon*
5. 1820 – 1839: with top authority *ethnobotany.org*

Groups 1 and 2 correspond to Cluster 3. They are separated here because we compute the normalized cut value at point 58 with respect to the whole subgraph of 1839 nodes. The value is below the threshold 0.06. While in our algorithm we first partition this subgraph and obtain a smaller subgraph with nodes from 1 to 109, then continue to partition this subgraph. At this point, the smallest normalized cut value is still obtained at node 58, but with the value of 0.0715 which is above the threshold. The partition is rejected. So groups 1 and 2 form one cluster in our algorithm. The result here indicates that our algorithm may be improved by checking the points with the normalized cut values below the threshold and partition the graph into several subgraphs at the same time.

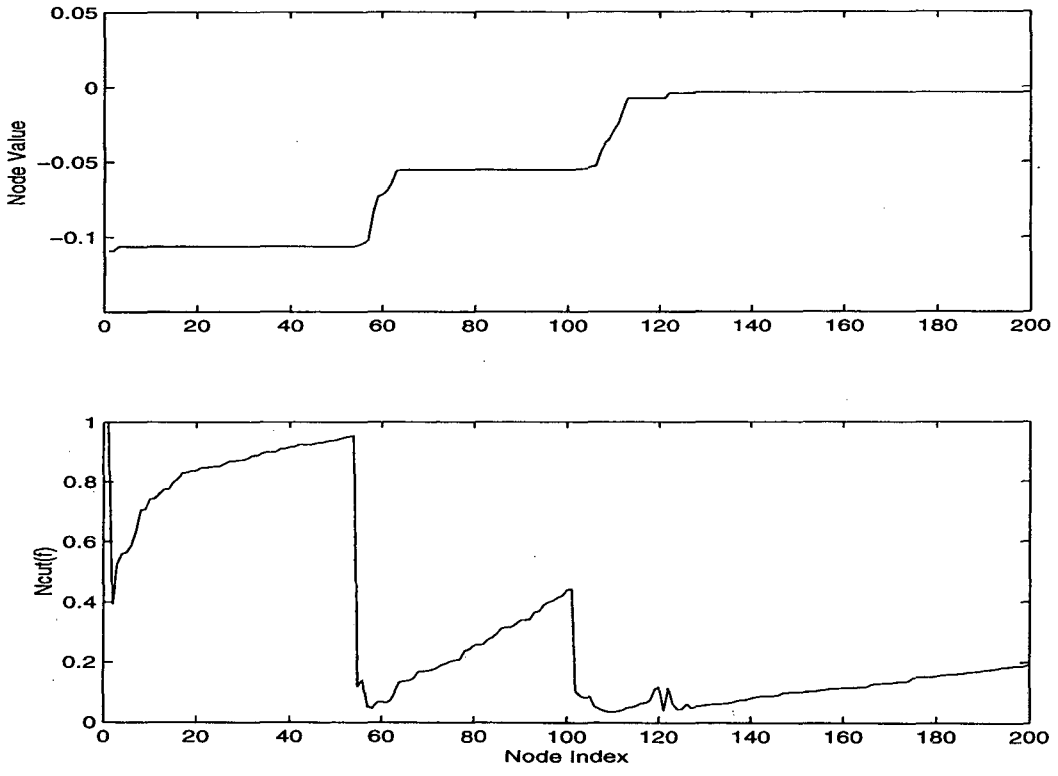

Fig. 2.5. The Sorted scaled Fiedler vector value (top) and corresponding normalized cut (bottom) for the first 200 nodes.

Our algorithm does find the cluster corresponding to group 5. Since its size is too small, we didn't list it in our results.

2.7.3 Robustness

Figure 2.6 is the plot of the α value in (2.1) and the corresponding average normalized cut value. The graph shows that for the query term *amazon*(top), the smallest average normalized cut value corresponds to $\alpha = 0.7$ while for the query term *star*(bottom), $\alpha = 0.1$ leads to the smallest average normalized cut value. From this figure, we know that we cannot choose an α value or a range of α values that can minimize the average normalized cut in all cases.

For the query *amazon*, $\alpha = 0.7$ gives the best average normalized cut. We choose this α value to test the average normalized cut under different threshold. In case of the query *star*, it is $\alpha = 0.1$ that gives the best average normalized cut. But we can't make α too small, since doing so will lower the importance of text similarity. So we choose $\alpha = 0.4$ to test because it is a local minimum. From Figure 2.7 we see that, as the threshold increases, the average normalized cut value increases, too. The relationship between them are almost linear.

2.7.4 Multi-level partitioning method

We also applied multi-level K-way partitioning method METIS [52]. The result is not quite as good. (In fact, we use normalized cut method after working with METIS for a while.) METIS tends to produce subgraphs with nearly equal sizes, while sizes of different clusters are generally different in clustering, especially for web graph. More

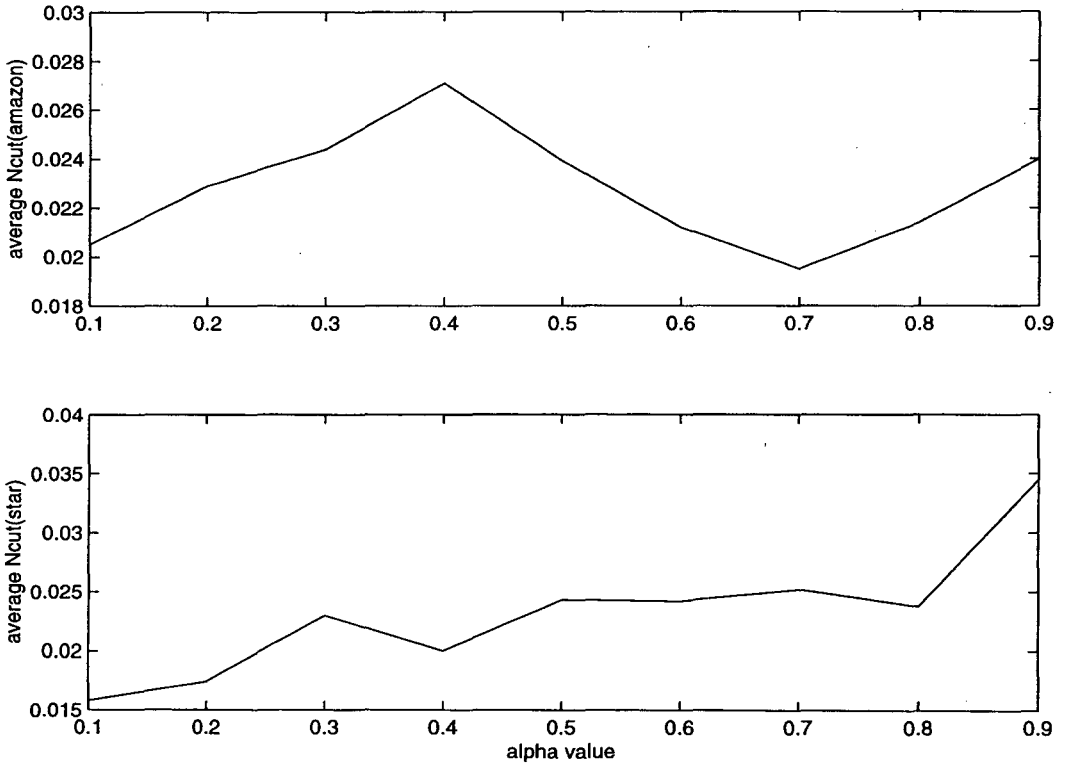

Fig. 2.6. α value and corresponding average normalized cut. Top: *amazon*, Bottom: *star*.

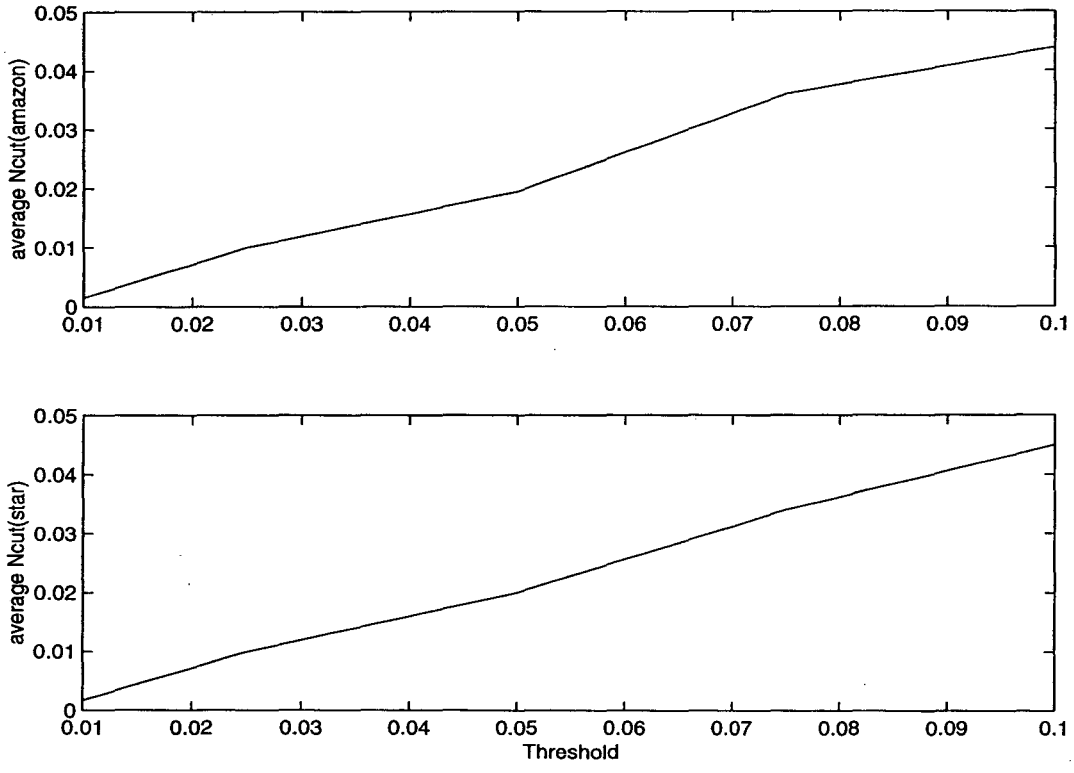

Fig. 2.7. Threshold and corresponding average normalized cut. Top: *amazon*, $\alpha = 0.7$; Bottom: *star*,$\alpha = 0.4$.

fundamentally, it appears that the node contraction procedure in METIS is not suitable for highly random graphs like webgraphs.

2.8 Concluding remarks

In this chapter, we present an algorithm to solve the web document clustering problem. We treat the problem as a graph partitioning problem, measure the partitioning result using the *normalized cut* criterion which was first proposed in the field of image segmentation. Combining normalized cut and the scaled Fiedler vector together, this approach forms a global, unbiased algorithm which can effectively extract different topics contained in the webgraph. Compared with the K-means based algorithm, it avoids creating clusters with small size by controlling the threshold on the value of the normalized cut. The clusters obtained have high similarity within clusters and dissimilarity between clusters. In our experiment, after choosing suitable threshold, we obtain the clusters, each with distinct topics.

Chapter 3

Min-max cut

In §2, the experimental results show that the normalized cut method (Ncut) works well for the web graph partitioning problem. When we apply this method to the similarity matrix formed from the newsgroup data sets, however, the clustering results are not as good as we had expected. After thorough investigation, we find that the problem is caused by the relatively larger overlap between two newsgroups.

Under the condition that two clusters have a small overlap, i.e., when the cut size is small compared to the association within each cluster, the Ncut can effectively avoid cutting off a small set of data objects. Whereas if the overlap is large, the chances that a small set of data objects is cut are increasing. Following we show that the Ncut tends to cut a small part of a graph if the overlap is large. Assuming the sum of weights in cluster A is roughly equal to that in cluster B. Let

$$\mathrm{cut}(A,B) \simeq \frac{1}{n}\mathrm{W}(A,A) \simeq \frac{1}{n}\mathrm{W}(B,B) \tag{3.1}$$

where n is a positive value larger than 1 and $\mathrm{W}(A,A) = \sum_{i,j\in A} a_{ij}$. The overlap is larger when n is smaller.

In the case when the partition is optimal, A and B are the exact partitioning results. The corresponding optimal Ncut value is

$$\begin{aligned} \text{Ncut}_0 &= \frac{\text{cut}(A,B)}{\text{W}(A,A)+\text{cut}(A,B)} + \frac{\text{cut}(A,B)}{\text{W}(B,B)+\text{cut}(A,B)} \\ &\simeq \frac{\frac{1}{n}}{1+\frac{1}{n}} + \frac{\frac{1}{n}}{1+\frac{1}{n}} \\ &\simeq \frac{2}{1+n}. \end{aligned}$$

For a skewed partition A_1 and B_1, assume $\text{W}(A_1,A_1) \ll \text{W}(B_1,B_1)$, then $\text{cut}(A_1,B_1) \ll \text{W}(B_1,B_1)$. The corresponding Ncut value is

$$\begin{aligned} \text{Ncut}_1 &= \frac{\text{cut}(A_1,B_1)}{\text{W}(A_1,A_1)+\text{cut}(A_1,B_1)} + \frac{\text{cut}(A_1,B_1)}{\text{W}(B_1,B_1)+\text{cut}(A_1,B_1)} \\ &\simeq \frac{\text{cut}(A_1,B_1)}{\text{W}(A_1,A_1)+\text{cut}(A_1,B_1)}. \end{aligned}$$

For a skewed cut to happen, the condition $\text{Ncut}_1 < \text{Ncut}_0$ must hold, which leads to

$$\text{W}(A_1,A_1) > \frac{n-1}{2}\text{cut}(A_1,B_1). \tag{3.2}$$

This condition is easy to satisfy if n is small, i.e., when the overlap is large. Note that the above derivation does not eliminate the possibility that no small set is cut in some large overlap cases, or the small set is still created in some small overlap cases.

In this chapter, we propose a new graph partitioning method based on a min-max clustering principle: the similarity or association between two subgraphs (cut size) is minimized, while the similarity or association within each subgraph (summation of similarity

between all pairs of nodes within a subgraph) is maximized. These two requirements can be satisfied simultaneously with a simple min-max cut function. Compared with the normalized cut, this method can produce less skewed cut.

The relaxed version of the min-max cut function optimization leads to a generalized eigensystem problem. The second lowest eigenvector, the scaled Fiedler vector, provides a linear search order. Thus the min-max cut algorithm provides both a well-defined objective and a clear procedure for searching for the optimal solution. We tested the algorithm on a number of newsgroup text data sets and compared it with some current methods. The min-max cut algorithm always outperformed them.

In spectral graph partition, it is generally believed that the search order provided by the (scaled) Fiedler vector gives the best linear search order for the optimal cut point [53, 44, 71], but it may not be the perfect one. As a partitioning result, we hope the node (we use node here to denote each data object) will have higher linkage with the cluster it belongs to. The experimental results show, however, this is not always the case. We find many nodes have higher linkage with another cluster than the one they are currently assigned to. This observation inspires us to investigate some refinement heuristic to improve the initial partitioning results. In this chapter, we introduce a linkage difference metric (called linkage differential order) that effectively identifies the nodes needed to be re-assigned. In our experiments, swapping them to the cluster with higher linkage, the objective function is reduced and the clustering accuracy is improved substantially. We should point out that the linkage differential ordering can start from any existing clustering results and improve the ordering and therefore the clustering results.

3.1 Min-max cut (Mcut)

Given a weighted graph $G = G(V, E)$ with weight matrix W, we wish to partition G into two subgraphs A and B using the min-max clustering principle: minimize similarity between clusters and maximize similarity within a cluster. This is a sound principle well established in the areas of statistics, data mining and machine learning. The min-max clustering principle requires we minimize $\text{cut}(A, B)$ while maximizing $\text{W}(A, A)$ and $\text{W}(B, B)$ at the same time. Both requirements can be simultaneously satisfied by the objective function (1.8). We call (1.8) the min-max cut function, or Mcut for short. The Mcut is inspired by the previous work on spectral graph partition [41, 68, 71].

Without loss of generality, let

$$W = \begin{pmatrix} W_A & W_{A,B} \\ W_{B,A} & W_B \end{pmatrix}.$$

Assume $\mathbf{x}$ and $\mathbf{y}$ are the vectors conformably partitioned with A and B, i.e., $\mathbf{x} = (1 \cdots 1, 0 \cdots 0)^T$, $\mathbf{y} = (0 \cdots 0, 1 \cdots 1)^T$, then

$$\begin{aligned} \text{cut}(A, B) &= \mathbf{x}^T(D - W)\mathbf{x} = \mathbf{y}^T(D - W)\mathbf{y}, \\ \text{W}(A, A) &= \mathbf{x}^T W \mathbf{x}, \quad \text{W}(B, B) = \mathbf{y}^T W \mathbf{y}. \end{aligned}$$

Hence the objective function (1.8) can be rewritten as

$$\text{Mcut}(A, B) = \frac{\mathbf{x}^T(D - W)\mathbf{x}}{\mathbf{x}^T W \mathbf{x}} + \frac{\mathbf{y}^T(D - W)\mathbf{y}}{\mathbf{y}^T W \mathbf{y}}. \tag{3.3}$$

Observe that in (3.3), the Mcut is invariant under the change of $\|\mathbf{x}\|_2$ and $\|\mathbf{y}\|_2$, and

$$\mathbf{x}^T D\mathbf{y} = 0 \quad \text{and} \quad \mathbf{x}^T W\mathbf{x} > 0, \quad \mathbf{y}^T W\mathbf{y} > 0.$$

The problem (3.3) can be relaxed into the following optimization problem

$$\min \quad \frac{\widehat{\mathbf{x}}^T(I-\widehat{W})\widehat{\mathbf{x}}}{\widehat{\mathbf{x}}^T\widehat{W}\widehat{\mathbf{x}}} + \frac{\widehat{\mathbf{y}}^T(I-\widehat{W})\widehat{\mathbf{y}}}{\widehat{\mathbf{y}}^T\widehat{W}\widehat{\mathbf{y}}}, \tag{3.4}$$

subject to $\|\widehat{\mathbf{x}}\|_2 = \|\widehat{\mathbf{y}}\|_2 = 1$, $\widehat{\mathbf{x}}^T\widehat{\mathbf{y}} = 0, \widehat{\mathbf{x}}^T\widehat{W}\widehat{\mathbf{x}} > 0, \widehat{\mathbf{y}}^T\widehat{W}\widehat{\mathbf{y}} > 0$, where $\widehat{\mathbf{x}} = D^{1/2}\mathbf{x}, \widehat{\mathbf{y}} = D^{1/2}\mathbf{y}$ and $\widehat{W} = D^{-1/2}WD^{-1/2}$. The conditions that $\widehat{\mathbf{x}}^T\widehat{W}\widehat{\mathbf{x}} > 0$ and $\widehat{\mathbf{y}}^T\widehat{W}\widehat{\mathbf{y}} > 0$ are necessary since $\widehat{W}$ in general is an indefinite matrix. Let the largest 2 eigenvalues of $\widehat{W}$ be $\widehat{\lambda}_1, \widehat{\lambda}_2$. $\widehat{\lambda}_1 = 1$ by construction. We have the following theorem.

THEOREM 3.1. *Assume that* $\widehat{\lambda}_1 + \widehat{\lambda}_2 > 0$. *Let vectors* $\widehat{\mathbf{x}}$ *and* $\widehat{\mathbf{y}}$ *solve problem (3.4). Choose* $\widehat{U}$ *to be any column orthogonal matrix such that* $\widehat{Q} = \left(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}, \widehat{U}\right)$ *is an* $n \times n$ *orthogonal matrix. Then*

$$\widehat{Q}^T\widehat{W}\widehat{Q} = \begin{pmatrix} \begin{pmatrix} \alpha & \gamma \\ \gamma & \alpha \end{pmatrix} & \mathbf{0} \\ \mathbf{0} & \widetilde{W} \end{pmatrix}$$

where $\alpha = (\widehat{\lambda}_1 + \widehat{\lambda}_2)/2$.

See Appendix B for the proof.

It follows from Theorem 3.1 that both ratios in (3.4) are equal at the optimal solution:

$$\frac{\widehat{\mathbf{x}}^T(I-\widehat{W})\widehat{\mathbf{x}}}{\widehat{\mathbf{x}}^T\widehat{W}\widehat{\mathbf{x}}} = \frac{\widehat{\mathbf{y}}^T(I-\widehat{W})\widehat{\mathbf{y}}}{\widehat{\mathbf{y}}^T\widehat{W}\widehat{\mathbf{y}}} = \frac{2}{\widehat{\lambda}_1+\widehat{\lambda}_2} - 1$$

and the optimal value is $\text{Mcut} = 4/(\widehat{\lambda}_1+\widehat{\lambda}_2) - 2$, which is equivalent to

$$\frac{1}{2}\text{Mcut}(\mathbf{x}) = \frac{\mathbf{x}^T(D-W)\mathbf{x}}{\mathbf{x}^TW\mathbf{x}} = \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^TW\mathbf{y}} \tag{3.5}$$

at the optimal solution. Equality (3.5) suggests that the resulting clusters tend to have similar weights and are thus well balanced. This fact makes the Mcut a much desired objective function for data clustering.

Let

$$J_M(\mathbf{x}) = \frac{\mathbf{x}^T(D-W)\mathbf{x}}{\mathbf{x}^TW\mathbf{x}}, \qquad J_N(\mathbf{x}) = \text{Ncut}(\mathbf{x}) = \frac{\mathbf{x}^T(D-W)\mathbf{x}}{\mathbf{x}^TD\mathbf{x}}.$$

It is easy to show that

$$J_M(\mathbf{x}) = \frac{J_N(\mathbf{x})}{1-J_N(\mathbf{x})}$$

which means $\mathbf{x}_0 = \text{argmin}_{\mathbf{x}} J_N(\mathbf{x}) = \text{argmin}_{\mathbf{x}} J_M(\mathbf{x})$.

(3.5) also implies $\mathbf{x}_0 = \text{argmin}_{\mathbf{x}} J_M(\mathbf{x}) = \text{argmin}_{\mathbf{x}} \text{Mcut}(\mathbf{x})$.

The above derivation leads to

$$\mathbf{x}_0 = \text{argmin}_{\mathbf{x}} J_N(\mathbf{x}) = \text{argmin}_{\mathbf{x}} \text{Mcut}(\mathbf{x}).$$

Note that in the relaxed form, the solution to $J_N(\mathbf{x})$ is exactly the solution to the normalized cut problem introduced in §2. Therefore, we have

$$\text{argmin}_{\mathbf{x}} \text{Ncut}(\mathbf{x}) = \text{argmin}_{\mathbf{x}} \text{Mcut}(\mathbf{x}), \tag{3.6}$$

and solving the min-max problem is similar to solving the normalized cut problem. The only difference is the different criteria used to find the optimal cut point.

3.1.1 Mcut algorithm

The algorithm for partitioning a graph into two subgraphs using the Mcut criterion is the following.

1. Compute the scaled Fiedler vector $\mathbf{f}$ of (1.7).

2. Sort $\mathbf{f}$ to obtain the Fiedler order.

3. Search for the optimal cut point corresponding to the lowest Mcut value based on the Fiedler order.

The running time for this algorithm is similar to that for the normalized cut algorithm discussed in section 2.3.2.

3.1.2 Experiments

3.1.2.1 Newsgroup article clustering

Now we apply the Mcut method to the newsgroup article data set. [1] This data set contains about 20,000 articles (email messages) evenly divided among 20 newsgroups. We list the names of the newsgroups together with the associated group labels.

NG1:	alt.atheism	NG2:	comp.graphics
NG3:	comp.os.ms-windows.misc	NG4:	comp.sys.ibm.pc.hardware
NG5:	comp.sys.mac.hardware	NG6:	comp.windows.x
NG7:	misc.forsale	NG8:	rec.autos
NG9:	rec.motorcycles	NG10:	rec.sport.baseball
NG11:	rec.sport.hockey	NG12:	sci.crypt
NG13:	sci.electronics	NG14:	sci.med
NG15:	sci.space	NG16:	soc.religion.christian
NG17:	talk.politics.guns	NG18:	talk.politics.mideast
NG19:	talk.politics.misc	NG20:	talk.religion.misc

We use the *bow* toolkit to construct the term-document matrix for this data set. Specifically we use the tokenization option so that the UseNet headers are stripped, and we also apply stemming [63]. Some of the newsgroups have large overlaps, such as the five newsgroups comp.* about computers. In fact several articles are posted to multiple newsgroups.

[1] The newsgroup data set together with the bow toolkit for processing it can be downloaded from http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html.

We focus on 3 two-cluster cases. These three data sets are

(1) NG1: alt.atheism + NG2: comp.graphics

(2) NG10: rec.sport.baseball + NG11: rec.sport.hockey

(3) NG18: talk.politics.mideast + NG19: talk.politics.misc

First, the term-document matrix M is constructed as follows. 2000 terms (words) are selected according to the mutual information between the terms and documents:

$$I(t) = \sum_d p(t,d) \log_2[p(t,d)/p(t)p(d)], \tag{3.7}$$

where t represents a term and d represents a document. $p(t,d)$ is the joint probability of t and d. The standard $tf \cdot idf$ scheme is used to calculate the term weighting, and each document is normalized to form the column of M. This is the vector space model of information retrieval [70]. Then the document-document similarities are calculated as $W = M^T M$. W is the weight/affinity matrix of the undirected graph and the Mcut algorithm is applied to this similarity matrix.

For comparison purpose, we consider two other clustering methods: the normalized cut and principle direction divisive partitioning (PDDP) [7]. PDDP is based on the idea of principle component analysis (PCA) applied to the vector space model, and has been shown to outperform several standard clustering methods such as hierarchical agglomerative algorithm [7]. In PDDP, each document is considered as a multivariate data point. The set of documents is normalized to have unit Euclidean length and then centered, i,e., the term-document matrix M is normalized first, then the average of each row (a term) is subtracted, and the first principle component is computed. The loadings

of each document (the projection of each document on the principle axis) form a $1-dim$ linear search order. This provides a heuristic similar to the linear search order provided by the Fiedler vector. Instead of searching through to find a minimum based on some objective functions, PDDP simply cuts data into two parts at the center of mass.

To increase statistics, we perform these two-cluster experiments in a way similar to cross-validation. We divide one newsgroup A randomly into K_1 subgroups and the other newsgroup B into K_2 subgroups. Then one subgroup of A is mixed with one subgroup of B to produce a data set S. The graph partitioning methods are run on S to produce two clusters. Since the true label of each news article is known, we use the accuracy, percentage of the news articles correctly clustered, as a measure of success. This is repeated for all $K_1 \cdot K_2$ pairs between A and B, and the mean and standard deviation of the accuracy are calculated.

	Mcut	Ncut	PDDP
NG1/NG2	97.2 ± 1.1 %	97.2 ± 0.8 %	96.4 ± 1.2%
NG10/NG11	79.5 ± 11.0 %	74.4 ± 20.4 %	89.1 ± 4.7%
NG18/NG19	83.6 ± 2.5 %	57.5 ± 0.9 %	71.9 ± 5.4%

Table 3.1.
Accuracy of clustering experiments using Mcut, Ncut and PDDP. Each test set is a mixture of 400 news articles, 200 from each newsgroup.

The clustering results are listed in Table 3.1 for balanced cases, i.e., both subgroups contain about 200 news articles. The Mcut performs about the same as the Ncut

for the mixture NG1/NG2, where the cluster overlap is small. The Mcut performs substantially better than the Ncut for the mixtures NG10/NG11 and NG18/NG19, where the cluster overlaps are large. Generally, the Mcut performs slightly better than PDDP.

	Mcut	Ncut	PDDP
NG1/NG2	97.6 ± 0.8 %	97.2 ± 0.8 %	90.6 ± 2.1%
NG10/NG11	85.7 ± 8.3 %	73.8 ± 16.6 %	87.4 ± 2.6%
NG18/NG19	78.8 ± 4.5 %	65.7 ± 0.5 %	59.6 ± 2.4%

Table 3.2.
Accuracy of clustering experiments using Mcut, Ncut and PDDP. Each test set is a mixture of 500 news articles, 200 from one newsgroup and 300 from the other newsgroup.

The results are listed in Table 3.2 for unbalanced cases, i.e., one subgroup contains about 300 news articles and another contains about 200. This is generally a harder problem due to the unbalanced prior distributions. In this case, both the Mcut and Ncut perform reasonably well. No clear deterioration is seen, while the performance of PDDP clearly drops. This indicates the strength of the Mcut method using the graph model. The Mcut consistently outperforms the Ncut for cases where the overlaps are large.

3.1.2.2 Web document clustering

We test our Mcut algorithm on the web documents, trying to partition the web link graph in order to create clusters with distinct topics. We use the same data sets as

in §2. The threshold on the Mcut value is 0.06, the same as in the Ncut case, though this threshold can be set to different values. The only difference from the Ncut algorithm is the objective function used.

Query term: *amazon*

Applying the Mcut algorithm, we obtain the significant clusters similar to the ones presented in §2 with the following exceptions.

Cluster 2 of the Ncut results is further separated into following two clusters:

www.amazoncity.com/

www.amazoncityradio.com/

www.etsu.edu/cas/history/women.htm

www.sfweekly.com/extra/femizines.html

roterommet.org/linker/kvinnelinker.htm

which is related to the women's issues, and

radio.amazoncity.com/

www.amazoncity.com/central/cityhall/press.html

www.trabanino.com/dothat/radio.htm

www.amazoncityradio.com/bodypeace

www.theonestopwebshop.co.za/info.html

which concentrates on the radio about the women's issues.

Cluster 3 of the Ncut results is separated into

www.amazon.org/

www.dartmouth.edu/~glbprog/links.html

www.teleport.com/~rocky/queer.shtml

www.fc.net/~zarathus/links.html

www.advocate.com/html/gaylinks/resources.html

and

www.amazonfembks.com/

www.igc.apc.org/women/bookstores/

www.wrc.bc.ca/femlinks.htm

www.pressgang.bc.ca/links.htm

search.jassan.com/Hobbies_and_Lifestyle/Women

The former talks about the bi-sexual issues and the latter mentions the books about the women's issues.

Query terms: *star* and *apple*

For these two data sets, the Mcut method generates the same clusters as the Ncut does.

From the clustering results, we see that these two methods, the Ncut and Mcut, produce the results not differing too much from each other. There are two possible reasons for that: 1) the linkage between the web communities discovered by these two methods is relatively small; 2) by adjusting the threshold, the cluster already formed can be further partitioned by either method.

3.2 Skewed cut

Now we study the reasons that the Mcut consistently outperforms the Ncut in large overlap cases. The most important reason is that the Ncut often cuts out a very small set, i.e., a skewed cut. Two specific cases are shown in Figure 3.1. The cut points

for the Mcut and Ncut and the relevant quantities are listed in Table 3.3. We see that the Ncut has two pronounced valleys, and produces a skewed cut in both cases, while the Mcut has a very flat valley and gives balanced cuts. Further examination shows that in both cases, the cut sizes in the Ncut are equal or bigger than the smaller self-similarities, as listed in Table 3.3. Taking the left panels in Figure 3.1 as example, the Ncut produces a cut size of 262.7, much larger than the self-similarity $\mathrm{W}(B,B) = 169$. Clearly, the Ncut graph partitioning method is not appropriate for these cases. For the Mcut method, the cut size is absent in the denominators; this provides a balanced cut. These case studies provide some insights to the graph partitioning methods.

	i_{cut}	$\mathrm{cut}(A,B)$	$\mathrm{W}(A,A)$	$\mathrm{W}(B,B)$
Ncut	364	262.7	5312.6	169.0
Mcut	141	1026.6	1488.9	2464.7
Ncut	31	213.5	172.4	5167.9
Mcut	202	1082.1	1670.0	1933.0

Table 3.3.
cut point i_{cut}, cut size $\mathrm{cut}(A,B)$, and self-similarities $\mathrm{W}(A,A)$ and $\mathrm{W}(B,B)$ for the two cases in Figure 3.1: top two lines for the left three panels, and bottom two lines for the right three panels.

Prompted by these case studies, we now analyze the situation under which a skewed cut will occur. Consider the balanced cases in which $\mathrm{W}(A,A) \simeq \mathrm{W}(B,B)$ and

$\text{cut}(A,B)$ satisfies (3.1). When the partition is optimal, A and B are the exact partitioning results. The corresponding Mcut value is

$$\text{Mcut}_0 \quad = \quad \frac{\text{cut}(A,B)}{\text{W}(A,A)} + \frac{\text{cut}(A,B)}{\text{W}(B,B)} \quad \simeq \quad \frac{2}{n}.$$

For a skewed partition A_1, B_1, assume $\text{W}(A_1,A_1) \ll \text{W}(B_1,B_1)$, hence $\text{cut}(A_1,B_1) \ll \text{W}(B_1,B_1)$. The corresponding Mcut value is

$$\text{Mcut}_1 \simeq \frac{\text{cut}(A_1,B_1)}{\text{W}(A_1,A_1)}.$$

A skewed cut occurs if $\text{Mcut}_1 < \text{Mcut}_0$, which means

$$\text{W}(A_1,A_1) > \frac{n}{2}\text{cut}(A_1,B_1). \tag{3.8}$$

When the overlap is small, i.e., n is large, (3.2) and (3.8) are not much different, as we have expected from the definition of the objective functions. When n is small, condition (3.8) is less possible to be satisfied, hence the Mcut tends to create less skewed cut than the Ncut.

3.2.1 Random graph model

Perhaps the most important feature of the Mcut method is that it tends to produce balanced cut, i.e., the resulting clusters have similar sizes. In this section, we use the random graph model [8, 14] to illustrate this point. We consider four partitioning methods: the Mincut, Rcut, Ncut and Mcut.

THEOREM 3.2. *For random graphs, the Mincut favors highly skewed cuts, i.e., very uneven sizes. The Mcut favors balanced cut, i.e., both subgraphs have the same sizes. The Rcut and Ncut show no preferences.*

Proof. We partition graph G into two subgraphs A and B. Assume two nodes are connected by an edge with probability p, and $n = |G|$, the number of nodes in G. Note that the average number of edges between A and B is $p|A||B|$.

From (1.4), we have

$$\text{Mincut}(A, B) = p|A||B|.$$

For the Rcut (1.5), we have

$$\text{Rcut}(A, B) = \frac{p|A||B|}{|A|} + \frac{p|A||B|}{|B|} = p(|A| + |B|) = np.$$

For the Ncut (1.6), since all nodes have the same degree $(n-1)p$,

$$\text{Ncut}(A, B) = \frac{p|A||B|}{p|A|(n-1)} + \frac{p|A||B|}{p|B|(n-1)} = n/(n-1).$$

For the Mcut (1.8), we have

$$\begin{aligned} \text{Mcut}(A, B) &= \frac{p|A||B|}{p|A|(|A|-1)} + \frac{p|A||B|}{p|B|(|B|-1)} \\ &= \frac{|B|}{|A|-1} + \frac{|A|}{|B|-1}. \end{aligned}$$

Now we minimize these objectives. Clearly, the Mincut favors $|A| = n-1$ and $|B| = 1$, or $|B| = n-1$ and $|A| = 1$, both of which are skewed cuts. Minimizing $\text{Mcut}(A, B)$, we

obtain a balanced cut: $|A| = |B|$, a highly desirable property for clustering. The Rcut and Ncut objectives have no size dependency and no size preference, which also implies possible unstable results.

3.3 Linkage-based refinements

The linear search order provided by the scaled Fiedler vector is generally a good heuristic, as the results shown above. Nevertheless, it may not necessarily be the perfect one. Here we explore this subtle point and investigate some effective refinement methods which can improve the quality of graph partition substantially.

The linear search order provided by sorting the scaled Fiedler vector $\mathbf{f}$ implies that the nodes on one side of the cut point must belong to the same cluster. In other words, if $f_u \geq f_v \geq f_w$, the linear search will not allow the situation to happen that u, w belong to one cluster and v belongs to the other cluster. Such a strict order is not necessarily followed by the nodes near the cut point. In fact, in large overlap cases, we may expect that some nodes could be moved to the other side of the cut point while lowering the overall objective function value.

How to identify those nodes near the cut point? To answer this question, we define the linkage ℓ, a closeness or similarity measure between a node and a cluster:

$$\ell(u, A) = \mathrm{W}(u, A)/\mathrm{W}(A, A).$$

Here $W(A, A)$ is for the normalization purpose so that a large subgraph will not necessarily has a large similarity with a node. The linkage between two groups, such as the single linkage, double linkage, are often used in hierarchical agglomerative clustering.

With the definition of linkage, we can identify the nodes near the cut point. If a node u is well inside a cluster, u will have a large linkage with the cluster, and small linkage with the other cluster. One the other hand, if u is near the cut point, it will have similar linkage values with both clusters. Therefore, we define the linkage difference

$$\Delta\ell(u) = \ell(u, A) - \ell(u, B).$$

A node with small $\Delta\ell$ should be close to the cut point and is a possible candidate to be moved to the other cluster.

Figure 3.2 shows the linkage difference $\Delta\ell$ for all nodes. The vertical line represents the cut point. It is interesting to observe that, not only many nodes have small $\Delta\ell$, but quite a few nodes have wrong signs of $\Delta\ell$ (i.e., $\Delta\ell(u) < 0$ while $u \in A$, or $\Delta\ell(v) > 0$ while $v \in B$). For example, node #62 has a relatively large negative $\Delta\ell$, which implies node #62 has a larger linkage with cluster B even though it is located to the left of the cut point. Indeed, if we move node #62 to cluster B, the objective function value (1.8) is reduced. This way, we find a better solution to the graph partitioning problem (based on the Mcut criterion) than the one obtained by the Fiedler order.

In practice, we can try to move to cluster B all the nodes in A with negative $\Delta\ell$, by testing if the objective function values are lowered. Similarly, we can try to move to A all the nodes in B with positive $\Delta\ell$. This procedure of swapping the nodes

with wrong $\Delta\ell$ signs to the opposite cluster is called the "linkage-based swap". This swap reduces the objective function value and increases the partitioning quality. The effect on the clustering accuracy due to the swap is listed in Table 3.4. In all cases, the accuracy increases. Note that in the large overlap cases, NG9/NG10 and NG18/NG19, the accuracy increases by nearly 10% over the Mcut alone. This is our first refinement over the initial Mcut results.

	Mcut	Mcut+Swap	Mcut+Swap+Move
NG1/NG2	97.2 ± 1.1%	97.5 ± 0.8 %	97.8 ±0.7%
NG10/NG11	79.5 ±11.0%	85.0 ± 8.9 %	92.8 ±6.3%
NG18/NG19	83.6 ± 2.5%	87.8 ± 2.0 %	89.5 ±2.2%

Table 3.4.
Accuracy improvement due to linkage-based refinement for Mcut, Mcut + swap, and Mcut + swap + move over 5% smallest $\Delta\ell$ on both sides of the cut point.

Next, we sort the remaining nodes in a cluster according to $\Delta\ell$ and select the lowest 5% of them as the candidates, then move those which reduce the Mcut objective to the other cluster. This is done for both A and B. This procedure is called "linkage-based move". Again, these moves reduce the Mcut objective and therefore improve clustering results. Their effect on improving clustering accuracy is shown in Table 3.4. Adding together, the linkage based refinements improve the accuracy by 20%. The final Mcut results are about 30-50% better than the Ncut and about 6-25% better than PDDP.

Note that our refinement is similar to the Fiduccia and Mattheyses (FM) heuristic algorithm [30] which is a modification of Kernighan-Lin algorithm [55]. The priority list

based on the linkage difference is similar to the list based on the *gain* in the FM algorithm. We do greedy movement in one pass, whereas the FM algorithm allows non-greedy moves.

3.4 Linkage differential linear order

It is generally believed that the Fiedler order given by the (scaled) Fiedler vector provides the best known linearized order for searching for the optimal cut, although counter examples with high symmetry exist [40, 74]. Is there a linear search order better than the Fiedler order?

Our analysis in the previous sections suggests a new linear search order. Given the linkage difference in Figure 3.2, we see that quite a few nodes far away from the cut point have wrong $\Delta\ell$ signs, which means they should belong to the other cluster. This strongly suggests that the Fiedler order is not necessarily the best linear search order. In fact, we can sort the linkage difference $\Delta\ell$ to obtain a new linear order, referred to as the *linkage differential* (LD) order. The search for finding the best cut point based on this new LD order represents another improvement over the original Mcut method.

	Acc(F)	Acc(LD)	Min(F)	Min(LD)
NG1/NG2	97.2 ± 1.1%	97.6±0.8 %	0.698	0.694
NG10/NG11	79.5 ±11.0%	87.2±8.0 %	1.186	1.087
NG18/NG19	83.6 ± 2.5%	89.2±1.8 %	1.126	1.057

Table 3.5.
Accuracy improvement based on the linkage differential (LD) order: clustering accuracy obtained on the Fiedler order (2nd column) and LD order (3rd column), minimum Mcut values obtained on the Fiedler order (4th column) and LD order (5th column).

The refinement results are given in Table 3.5. The Mcut values obtained on this new order are generally lower than those based on the Fiedler order. Compared to that based on the Fiedler order, the clustering accuracy increases substantially, and the clustering results based on the LD order is slightly better than those obtained by using Mcut+Swap in Table 3.4. Therefore, we find a new linear order that leads to better graph partitioning results.

Note that the LD order does not depend on the Fiedler order. Given any initial clustering results of two clusters, we can always calculate $\Delta\ell$ and sort it to obtain the LD order. For instance, we can obtain LD order based on the PDDP results. Furthermore, the LD order can be applied recursively to the clustering results obtained from previous LD order to further improve the results.

So far in this chapter, we have focused on bisectioning a graph into two subgraphs. If more subgraphs or clusters are expected, the Mcut and related refinement can be applied recursively to each subgraph, until certain stopping criteria is met, either the desired number of clusters is reached or the minimum Mcut value is above a certain pre-defined value.

Once the recursive division is stopped, some refinements discussed in section 3.3 should be applied. This is because even if all the nodes are optimally partitioned during each bisection step, the final partition is not necessarily the optimal, since they are not directly obtained from the k-way Mcut objective

$$\text{Mcut}_K = \frac{\text{cut}_1}{\text{W}(G_1)} + \frac{\text{cut}_2}{\text{W}(G_2)} + \cdots + \frac{\text{cut}_K}{\text{W}(G_K)}, \tag{3.9}$$

assuming G is partitioned into K subgraphs $G_1, \cdots, G_K$, $\text{cut}_i = \text{cut}(G_i, \bar{G}_i)$, and $\text{W}(G_i) = \text{W}(G_i, G_i)$, $i = 1, \cdots, K$. Suppose node u currently belongs to cluster G_i. The linkage difference $\Delta\ell_{ij}(u) = \ell(u, G_j) - \ell(u, G_i)$ for all other $K-1$ clusters should be computed, and u is moved to the cluster with the smallest $\Delta\ell_{ij}(u)$, provided that the the overall objective Mcut_K is reduced.

Define the k-way Ncut objective as

$$\text{Ncut}_K = \frac{\text{cut}_1}{\text{W}(G_1) + \text{cut}_1} + \frac{\text{cut}_2}{\text{W}(G_2) + \text{cut}_2} + \cdots + \frac{\text{cut}_K}{\text{W}(G_K) + \text{cut}_K}.$$

For $K > 2$, the average ratio $\text{cut}(G_1, \bar{G}_1)/\text{W}(G_1) = \sum_{i=2}^{K} \text{cut}(G_1, G_i)/\text{W}(G_1)$ will be larger than that when $K = 2$. Then the Mcut_K will differ from the Ncut_K much more than in the $K = 2$ case [cf. (2.2)]. From the previous analysis, the Ncut is more likely to produce a skewed cut. Therefore, the Mcut is essential in $K-$way partitions.

3.5 Concluding remarks

In this chapter, following the clustering principle, we introduce a new partitioning criterion, the min-max (Mcut) criterion, for graph partition. Compared to the Ncut algorithm which produces many skewed cuts in cases of large cluster overlap, the Mcut method produces more balanced partition. This result is preferable in many applications. We also propose a heuristic, the linkage difference metric, which effectively identifies those nodes near the cut point as the possible candidates to be moved to the other cluster; this leads to some effective refinement procedures. The new linkage differential order is shown to provide a better linear search order than the best known Fiedler order.

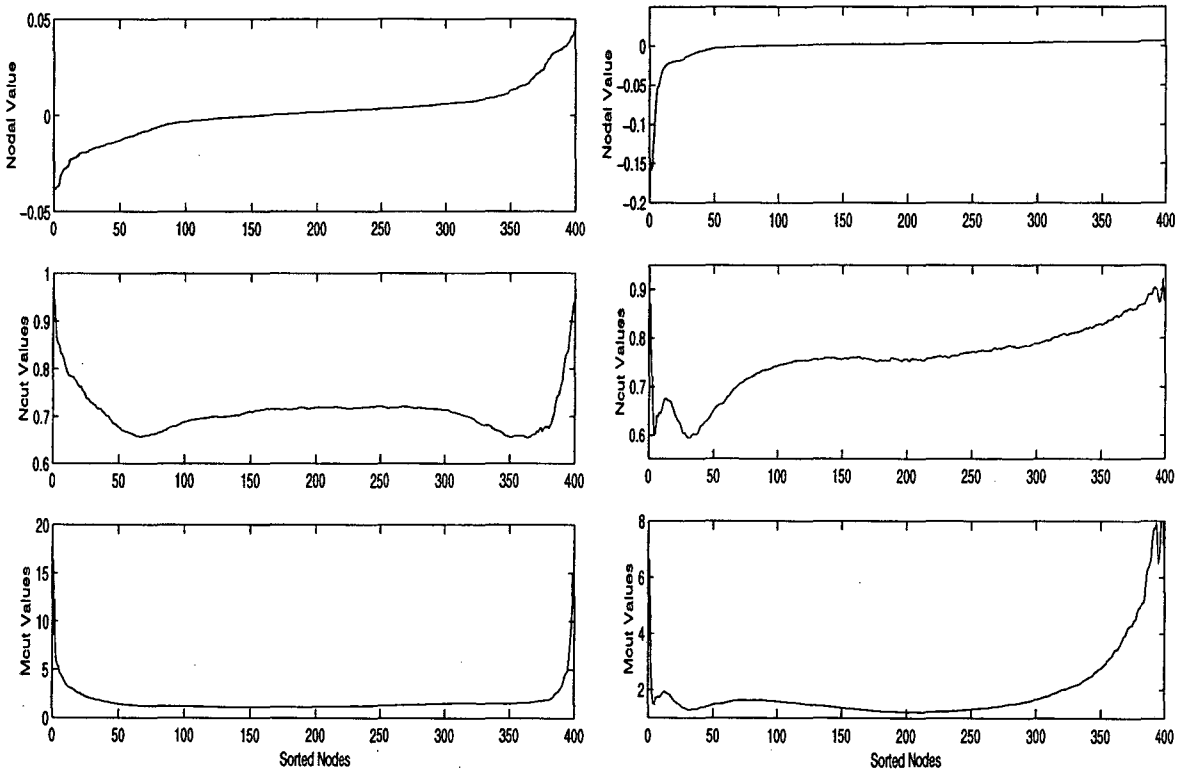

Fig. 3.1. Two cases in NG18/NG19 in Table 3.1 (left 3 and right 3 panels): sorted scaled Fiedler vector (top), Ncut values (middle) and Mcut values (bottom).

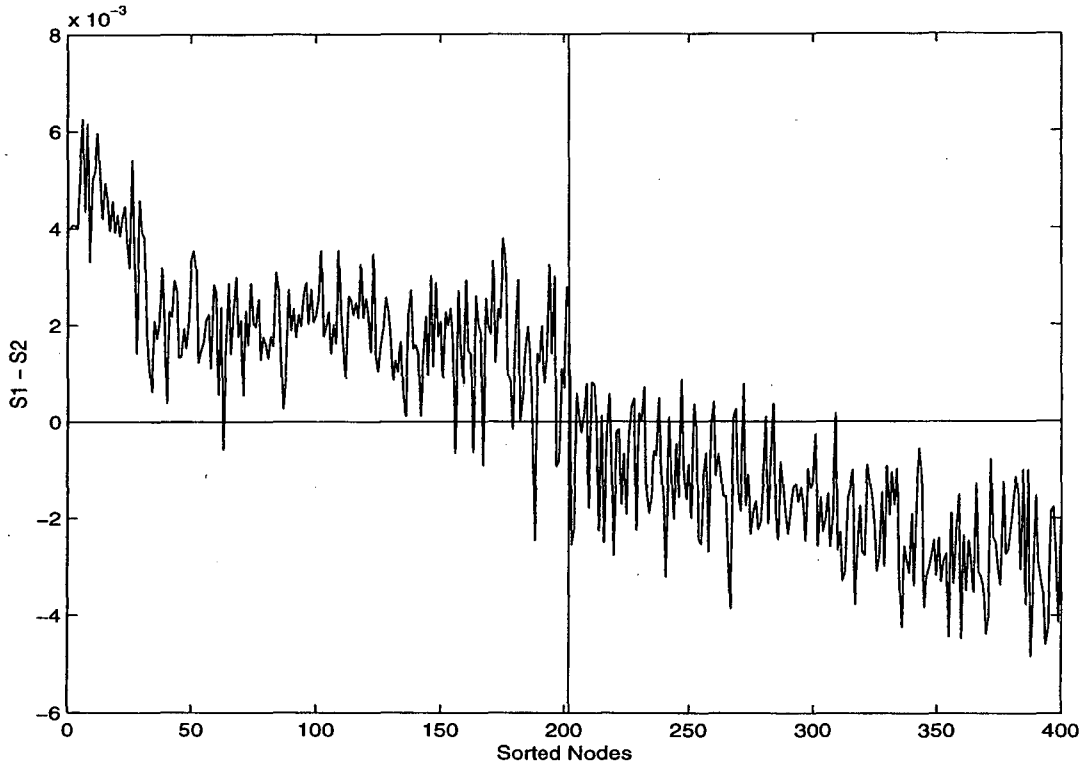

Fig. 3.2. Linkage difference of the nodes. The vertical line represents the cut point using the Mcut. Left-hand side of the cut point is Cluster A, and right-hand side is Cluster B.

Chapter 4

Bipartite graph partition using Ncut and Mcut

Many clustering algorithms are based on the assumptions that the given data set consists of covariate information (or attributes) for each individual data object, and cluster analysis can be cast as a problem of grouping a set of $n - dim$ vectors, each representing a data object in the data set. Such a data clustering problem can be modeled as the bipartite graph partition. Examples include terms and documents in a text corpus, customers and purchasing items in the market basket analysis, and reviewers and movies in a movie recommending system. Take the document clustering using the vector space model [3] as an example. Here each document is represented by an $n - dim$ vector, and each coordinate of the vector corresponds to a term in a vocabulary of size n. This formulation leads to the so-called term-document matrix $M = (m_{ij})$ for the representation of the collection of documents, where m_{ij} is the so-called term frequency, i.e., the number of times term i occurs in document j. In this vector space model, the terms and documents are treated asymmetrically with the terms considered as the covariates or attributes of the documents. It is also possible to treat both the terms and documents as the first-class citizens in a symmetric fashion, and consider m_{ij} as the frequency of co-occurrence of term i and document j, as is done, for example, in probabilistic latent semantic indexing [49]. In this chapter, we follow this basic principle and propose a new approach to model the term and document as vertices in a bipartite

graph with the edge of the graph indicating the co-occurrence of the term and document. In addition, we can optionally use the edge weight to indicate the frequency of this co-occurrence. Cluster analysis for document collections in this context is based on a very intuitive notion: documents are grouped by topics. On one hand, the documents in a topic tend to use more heavily the same subset of terms which form a term cluster, and on the other hand, a topic is usually characterized by a subset of terms and the documents heavily using those terms tend to address that particular topic. It is this interplay of the terms and documents that gives rise to what we call bi-clustering by which the terms and documents are simultaneously grouped into *semantically coherent* clusters.

Many criteria have been proposed to measure the quality of graph partition result of an undirected graph. We will show, in this chapter, how to adapt the *normalized cut* (Ncut) criterion introduced in §2 and the *min-max cut* (Mcut) criterion in §3 to the bi-clustering problems which lead to a minimization problem by computing the partial singular value decomposition (SVD) of the associated edge weight matrix of the bipartite graph.

4.1 Bipartite graph partitioning

Denote a graph by $G(V, E)$, where V is the vertex set and E is the edge set of the graph. A graph $G(V, E)$ is *bipartite* with two vertex classes X and Y if $V = X \cup Y$ with $X \cap Y = \emptyset$, and each edge in E has one endpoint in X and the other in Y. We consider the weighted bipartite graph $G(X, Y, W)$ with $W = (w_{ij})$ where $w_{ij} > 0$ denotes the weight of the edge between vertex i and j. $w_{ij} = 0$ if there is no edge between i and j.

In the context of document clustering, X represents the set of terms and Y represents the set of documents, and w_{ij} can be used to denote the number of times term i occurs in document j. A vertex partition of $G(X,Y,W)$ denoted by $\Pi(A,B)$ is defined by a partition of the vertex sets X and Y, respectively: $X = A \cup A^c$, and $Y = B \cup B^c$, where for set S, S^c denotes its compliment. By convention, we pair A with B, and A^c with B^c. We say that a pair of vertices $x \in X$ and $y \in Y$ are *matched* with respect to a partition $\Pi(A,B)$ if there is an edge between x and y, and either $x \in A$ and $y \in B$ or $x \in A^c$ and $y \in B^c$. In the context of cluster analysis, the edge weight measures the similarity between two data objects. To partition the data objects into clusters, we seek a partition of $G(X,Y,W)$ such that the association (similarity) between unmatched vertices is as small as possible. One possibility is to consider the following quantity

$$\begin{aligned} \text{cut}(A,B) &\equiv \text{W}(A,B^c) + \text{W}(A^c,B) \\ &= \textstyle\sum_{i\in A, j\in B^c} w_{ij} + \sum_{i\in A^c, j\in B} w_{ij}. \end{aligned} \tag{4.1}$$

Intuitively, choosing $\Pi(A,B)$ to minimize $\text{cut}(A,B)$ will give rise to a partition that minimizes the sum of all the edge weights between unmatched vertices. In the context of document clustering, we try to find two document clusters B and B^c which have few terms in common, and the documents in B mostly use the terms in A and those in B^c use the terms in A^c. Unfortunately, choosing a partition based entirely on $\text{cut}(A,B)$ tends to produce unbalanced clusters, i.e., the sizes of A and/or B or their compliments tend to be small. Inspired by the work in [15, 24, 71] and the work in §3, we propose

the following two normalized variants of the edge cut in (4.1)

$$\mathrm{Ncut}(A,B) \equiv \frac{\mathrm{cut}(A,B)}{\mathrm{W}(A,Y)+\mathrm{W}(X,B)} + \frac{\mathrm{cut}(A^c,B^c)}{\mathrm{W}(A^c,Y)+\mathrm{W}(X,B^c)},$$

and

$$\mathrm{Mcut}(A,B) \equiv \frac{\mathrm{cut}(A,B)}{\mathrm{W}(A,B)} + \frac{\mathrm{cut}(A^c,B^c)}{\mathrm{W}(A^c,B^c)}.$$

Above are the normalized cut and min-max cut criteria we discussed in the previous two chapters. The intuition behind these two criteria is that, we want not only a partition with a small edge cut, but also wthe two subgraphs formed by the matched vertices to be as dense as possible. The latter requirement is partially satisfied by introducing the normalizing denominators in the above equations. Our bi-clustering problem is now equivalent to the following optimization problem

$$\min_{\Pi(A,B)} \mathrm{Ncut}(A,B) \qquad \text{or} \qquad \min_{\Pi(A,B)} \mathrm{Mcut}(A,B),$$

i.e., finding a partition of the vertex sets X and Y so as to minimize $\mathrm{Ncut}(A,B)$ or $\mathrm{Mcut}(A,B)$ of the bipartite graph $G(X,Y,W)$.

4.2 Approximate solutions using singular vectors

Given a bipartite graph $G(X,Y,W)$ and the associated partition $\Pi(A,B)$, reorder the vertices of X and Y so that the vertices in A and B are ordered before those in A^c

and B^c, respectively. The weight matrix W can be written as

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}, \tag{4.2}$$

i.e., the rows of W_{11} correspond to the vertices in A and the columns of W_{11} correspond to those in B. Therefore $G(A, B, W_{11})$ denotes the weighted bipartite graph corresponding to the vertex sets A and B. For any m-by-n matrix $H = (h_{ij})$, define

$$\mathrm{s}(H) = \sum_{i=1}^{m} \sum_{j=1}^{n} h_{ij},$$

i.e., $\mathrm{s}(H)$ is the sum of all the elements of H.

Now we begin with the solution to the Ncut problem.

4.2.1 Bipartite clustering using Ncut

It is easy to see that

$$\mathrm{Ncut}(A, B) = \frac{\mathrm{s}(W_{12}) + \mathrm{s}(W_{21})}{2\mathrm{s}(W_{11}) + \mathrm{s}(W_{12}) + \mathrm{s}(W_{21})} + \frac{\mathrm{s}(W_{12}) + \mathrm{s}(W_{21})}{2\mathrm{s}(W_{22}) + \mathrm{s}(W_{12}) + \mathrm{s}(W_{21})}.$$

In order to make connections to SVD problems, we first consider the case when W is symmetric.[1] It is easy to see that with W symmetric (denoting $\mathrm{Ncut}(A, A)$ by $\mathrm{Ncut}(A)$),

[1] A different proof for the symmetric case was first derived in [71]. Our derivation, however, is simpler and more transparent, and leads naturally to the SVD problems for the rectangular case.

we have

$$\mathrm{Ncut}(A) = \frac{\mathrm{s}(W_{12})}{\mathrm{s}(W_{11}) + \mathrm{s}(W_{12})} + \frac{\mathrm{s}(W_{12})}{\mathrm{s}(W_{22}) + \mathrm{s}(W_{12})}. \tag{4.3}$$

Let $\mathbf{e}$ be the vector of all 1's, and $D = \mathrm{diag}(W\mathbf{e})$, then $(D - W)\mathbf{e} = 0$. Let $\mathbf{x} = (x_i)$ be the vector with

$$x_i = \begin{cases} 1, & i \in A, \\ -1, & i \in A^c. \end{cases}$$

It is easy to verify that

$$\mathrm{s}(W_{12}) = \mathbf{x}^T(D - W)\mathbf{x}/4.$$

Define

$$p \equiv \frac{\mathrm{s}(W_{11}) + \mathrm{s}(W_{12})}{\mathrm{s}(W_{11}) + 2\mathrm{s}(W_{12}) + \mathrm{s}(W_{22})} = \frac{\mathrm{s}(W_{11}) + \mathrm{s}(W_{12})}{\mathbf{e}^T D\mathbf{e}},$$

then

$$\mathrm{s}(W_{11}) + \mathrm{s}(W_{12}) = p\mathbf{e}^T D\mathbf{e},$$

$$\mathrm{s}(W_{22}) + \mathrm{s}(W_{12}) = (1-p)\mathbf{e}^T D\mathbf{e},$$

and

$$\text{Ncut}(A) = \frac{\mathbf{x}^T(D-W)\mathbf{x}}{4p(1-p)\mathbf{e}^T D\mathbf{e}}. \tag{4.4}$$

Note that $(D-W)\mathbf{e} = 0$, then for any scalar s, we have

$$(s\mathbf{e}+\mathbf{x})^T(D-W)(s\mathbf{e}+\mathbf{x}) = \mathbf{x}^T(D-W)\mathbf{x}.$$

To cast (4.4) into the form of a Rayleigh quotient, we need to find s such that

$$(s\mathbf{e}+\mathbf{x})^T D(s\mathbf{e}+\mathbf{x}) = 4p(1-p)\mathbf{e}^T D\mathbf{e}.$$

Since $\mathbf{x}^T D\mathbf{x} = \mathbf{e}^T D\mathbf{e}$, it follows from the above equation that $s = 1-2p$. Now let $\mathbf{y} = (1-2p)\mathbf{e}+\mathbf{x}$. It is easy to see that $\mathbf{y}^T D\mathbf{e} = ((1-2p)\mathbf{e}+\mathbf{x})^T D\mathbf{e} = 0$, and

$$y_i = \begin{cases} 2(1-p) > 0, & i \in A, \\ -2p < 0, & i \in A^c. \end{cases}$$

Thus

$$\min_A \text{Ncut}(A) = \min\left\{ \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \mid \mathbf{y} \in S \right\},$$

where

$$S = \{\mathbf{y} \mid \mathbf{y}^T D\mathbf{e} = 0, y_i \in \{2(1-p), -2p\}\}.$$

If we drop the constraints $y_i \in \{2(1-p), -2p\}$ and let the elements of $\mathbf{y}$ take arbitrary continuous values, then the optimal $\mathbf{y}$ can be approximated by the following relaxed *continuous* minimization problem,

$$\min \left\{ \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} \mid \mathbf{y}^T D\mathbf{e} = 0 \right\}. \tag{4.5}$$

Note that it follows from $W\mathbf{e} = D\mathbf{e}$ that

$$D^{-1/2}WD^{-1/2}(D^{1/2}\mathbf{e}) = D^{1/2}\mathbf{e},$$

and therefore $D^{1/2}\mathbf{e}$ is an eigenvector of $D^{-1/2}WD^{-1/2}$ corresponding to the eigenvalue 1. It is easy to show that all the eigenvalues of $D^{-1/2}WD^{-1/2}$ have absolute value at most 1 (See Appendix C.(1)). Thus the optimal $\mathbf{y}$ in (4.5) can be computed as $\mathbf{y} = D^{1/2}\hat{\mathbf{y}}$, where $\hat{\mathbf{y}}$ is the *second* largest eigenvector of $D^{-1/2}WD^{-1/2}$.

Now we return to the rectangular case for the weight matrix W, and let D_X and D_Y be diagonal matrices such that

$$W\mathbf{e} = D_X\mathbf{e}, \quad W^T\mathbf{e} = D_Y\mathbf{e}. \tag{4.6}$$

Consider a partition $\Pi(A, B)$, and define

$$u_i = \begin{cases} 1, & i \in A \\ -1, & i \in A^c \end{cases}, \quad v_i = \begin{cases} 1, & i \in B \\ -1, & i \in B^c \end{cases}.$$

Let W have the block form as in (4.2), and consider the augmented symmetric matrix[2]

$$\hat{W} = \begin{bmatrix} 0 & W \\ W^T & 0 \end{bmatrix} = \left[\begin{array}{cc|cc} 0 & 0 & W_{11} & W_{12} \\ 0 & 0 & W_{21} & W_{22} \\ \hline W_{11}^T & W_{21}^T & 0 & 0 \\ W_{12}^T & W_{22}^T & 0 & 0 \end{array}\right].$$

Interchanging the second and third block rows and columns of the above matrix, we obtain

$$\left[\begin{array}{cc|cc} 0 & W_{11} & 0 & W_{12} \\ W_{11}^T & 0 & W_{21}^T & 0 \\ \hline 0 & W_{21} & 0 & W_{22} \\ W_{12}^T & 0 & W_{22}^T & 0 \end{array}\right] \equiv \begin{bmatrix} \hat{W}_{11} & \hat{W}_{12} \\ \hat{W}_{12}^T & \hat{W}_{22} \end{bmatrix},$$

and the normalized cut can be written as

$$\mathrm{Ncut}(A,B) = \frac{\mathrm{s}(\hat{W}_{12})}{\mathrm{s}(\hat{W}_{11}) + \mathrm{s}(\hat{W}_{12})} + \frac{\mathrm{s}(\hat{W}_{12})}{\mathrm{s}(\hat{W}_{22}) + \mathrm{s}(\hat{W}_{12})}, \tag{4.7}$$

a form that resembles the symmetric case (4.3). Define

$$q = \frac{2\mathrm{s}(W_{11}) + \mathrm{s}(W_{12}) + \mathrm{s}(W_{21})}{\mathbf{e}^T D_X \mathbf{e} + \mathbf{e}^T D_Y \mathbf{e}},$$

[2]In [43], the Laplacian of $\hat{W}$ is used for partitioning a rectangular matrix in the context of designing load-balanced matrix-vector multiplication algorithms for parallel computation. However, the eigenvalue problem of the Laplacian of $\hat{W}$ does not lead to a simpler singular value problem.

then we have

$$\begin{aligned} \mathrm{Ncut}(A,B) &= \frac{-2\mathbf{x}^T W\mathbf{y} + \mathbf{x}^T D_X\mathbf{x} + \mathbf{y}^T D_Y\mathbf{y}}{\mathbf{x}^T D_X\mathbf{x} + \mathbf{y}^T D_Y\mathbf{y}} \\ &= 1 - \frac{2\mathbf{x}^T W\mathbf{y}}{\mathbf{x}^T D_X\mathbf{x} + \mathbf{y}^T D_Y\mathbf{y}}, \end{aligned}$$

where $\mathbf{x} = (1-2p)\mathbf{e} + \mathbf{u}, \mathbf{y} = (1-2p)\mathbf{e} + \mathbf{v}$. It is also easy to see that

$$\mathbf{x}^T D_X\mathbf{e} + \mathbf{y}^T D_Y\mathbf{e} = 0, \quad x_i, y_i \in \{2(1-q), -2q\}. \tag{4.8}$$

Therefore,

$$\min_{\Pi(A,B)} \mathrm{Ncut}(A,B) = 1 - \max_{\substack{\mathbf{x}\neq 0 \\ \mathbf{y}\neq 0}} \left\{ \frac{2\mathbf{x}^T W\mathbf{y}}{\mathbf{x}^T D_X\mathbf{x} + \mathbf{y}^T D_Y\mathbf{y}} \mid \mathbf{x},\mathbf{y} \text{ satisfy (4.8)} \right\}.$$

Ignoring the discrete constraints on the elements of $\mathbf{x}$ and $\mathbf{y}$, we have the following continuous maximization problem,

$$\max_{\substack{\mathbf{x}\neq 0 \\ \mathbf{y}\neq 0}} \left\{ \frac{2\mathbf{x}^T W\mathbf{y}}{\mathbf{x}^T D_X\mathbf{x} + \mathbf{y}^T D_Y\mathbf{y}} \mid \mathbf{x}^T D_X\mathbf{e} + \mathbf{y}^T D_Y\mathbf{e} = 0 \right\}. \tag{4.9}$$

Without the constraints $\mathbf{x}^T D_X\mathbf{e} + \mathbf{y}^T D_Y\mathbf{e} = 0$, the above problem is equivalent to computing the largest singular triplet of $D_X^{-1/2} W D_Y^{-1/2}$ (Appendix C.(2)). From (4.6),

we have

$$D_X^{-1/2} W D_Y^{-1/2} (D_Y^{1/2}\mathbf{e}) = D_X^{1/2}\mathbf{e}$$
$$(D_X^{-1/2} W D_Y^{-1/2})^T (D_X^{1/2}\mathbf{e}) = D_Y^{1/2}\mathbf{e}.$$

Similarly to the symmetric case, it is easy to show that all the singular values of $D_X^{-1/2} W D_Y^{-1/2}$ are at most 1. Therefore, an optimal pair $\{\mathbf{x}, \mathbf{y}\}$ for (4.9) can be computed as $\mathbf{x} = D_X^{-1/2}\hat{\mathbf{x}}$ and $\mathbf{y} = D_Y^{-1/2}\hat{\mathbf{y}}$, where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the *second* largest left and right singular vectors of $D_X^{-1/2} W D_Y^{-1/2}$, respectively (Appendix C.(3)). With the above discussion, we now summarize our basic approach for bipartite graph clustering incorporating a recursive procedure.

ALGORITHM. Spectral Recursive Embedding (SRE)

Given a weighted bipartite graph $G = (X, Y, E)$ with edge weight matrix W:

1. Compute D_X and D_Y and form the scaled weight matrix $\hat{W} = D_X^{-1/2} W D_Y^{-1/2}$.
2. Compute the *second* largest left and right singular vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ of $\hat{W}$.
3. Find cut points c_x and c_y for $\mathbf{x} = D_X^{-1/2}\hat{\mathbf{x}}$ and $\mathbf{y} = D_Y^{-1/2}\hat{\mathbf{y}}$, respectively.
4. Form partitions $A = \{i \mid x_i \geq c_x\}$ and $A^c = \{i \mid x_i < c_x\}$ for vertex set X, and $B = \{j \mid y_j \geq c_y\}$ and $B^c = \{j \mid y_j < c_y\}$ for vertex set Y.
5. Recursively partition the subgraphs $G(A, B)$ and $G(A^c, B^c)$ if necessary.

Two basic strategies can be used for selecting the cut points c_x and c_y. The simplest strategy is to set $c_x = 0$ and $c_y = 0$. A more computing-intensive approach is to base the selection on the Ncut: check N equally spaced splitting points of $\mathbf{x}$ and $\mathbf{y}$, respectively; find the cut points c_x and c_y with the smallest Ncut.

Computational complexity. The major computational cost of SRE is in step 2 for computing the left and right singular vectors, which can be obtained either by power method or more robustly by the Lanczos bidiagonalization process [36, Chapter 9]. The Lanczos method is an iterative process for computing partial SVDs in which each iterative step involves the computation of two matrix-vector multiplications $\hat{W}\mathbf{u}$ and $\hat{W}^T\mathbf{v}$ for some vectors $\mathbf{u}$ and $\mathbf{v}$. The computational cost of these is roughly proportional to $\text{nnz}(\hat{W})$, the number of non-zero elements of $\hat{W}$. The total computational cost of SRE is $O(c_{\text{sre}} k_{\text{svd}} \text{nnz}(\hat{W}))$, where c_{sre} the the level of recursion and k_{svd} is the number of Lanczos iteration steps. In general, k_{svd} depends on the singular value gaps of $\hat{W}$. Also notice that $\text{nnz}(\hat{W}) = n_w n$, where n_w is the average number of terms per document and n is the total number of documents. Therefore, the total cost of SRE is in general linear in the number of documents to be clustered.

4.2.2 Bipartite clustering using Mcut

Note that in (4.7), the expression for the Ncut is indeed derived from the square weight matrix case. We have proven that, in the square weight matrix case, the same eigenvector used to minimize the Ncut value is exactly the same eigenvector that minimizes the Mcut value (cf. (3.6)). Therefore if the singular vectors $\mathbf{x}$ and $\mathbf{y}$ minimize the

Ncut value in (4.7), they can equivalently be used to minimize the following Mcut value

$$\text{Mcut}(A,B) = \frac{\text{s}(\hat{W}_{12})}{\text{s}(\hat{W}_{11})} + \frac{\text{s}(\hat{W}_{12})}{\text{s}(\hat{W}_{22})}.$$

Then the SRE algorithm and other analyses are still valid for the Mcut.

For a more detailed proof, see section 5.2.

4.3 Connections to correspondence analysis

In its basic form, correspondence analysis is applied to an m-by-n two-way table of counts W [38, 75, 4]. Let $w = \text{s}(W)$, the sum of all the elements of W. Correspondence analysis seeks to compute the largest singular triplets of the matrix $Z = (z_{ij}) \in \mathcal{R}^{m \times n}$ with

$$z_{ij} = \frac{w_{ij}/w - (D_X(i,i)/w)(D_Y(j,j)/w)}{\sqrt{(D_X(i,i)/w)(D_Y(j,j)/w)}}.$$

The matrix Z can be considered as the correlation matrix of two group indicator matrices for the original W [75]. Now we show that the SVD of Z is closely related to the SVD of $\hat{W} \equiv D_X^{-1/2} W D_Y^{-1/2}$. In fact, in section 4.2, we showed that $D_X^{1/2}\mathbf{e}$ and $D_Y^{1/2}\mathbf{e}$ are the left and right singular vectors of $\hat{W}$ corresponding to the singular value one, and it is also easy to show that all the singular values of $\hat{W}$ are at most 1. Therefore, the rest of the singular values and singular vectors of $\hat{W}$ can be found by computing the SVD of

the following rank-one modification of $\hat{W}$

$$D_X^{-1/2} W D_Y^{-1/2} - \frac{D_X^{1/2}\mathbf{e}\mathbf{e}^T D_Y^{1/2}}{\|D_X^{1/2}\mathbf{e}\|_2 \|D_Y^{1/2}\mathbf{e}\|_2}$$

which has the (i,j)th element

$$\frac{w_{ij}}{\sqrt{D_X(i,i)D_Y(j,j)}} - \frac{\sqrt{D_X(i,i)D_Y(j,j)}}{w} = w^2 z_{ij},$$

and is a constant multiple of the the (i,j)th element of Z. Therefore, the Ncut/Mcut based cluster analysis and correspondence analysis arrive at the same SVD problems even though they start with completely different principles. It is worthwhile to explore more deeply the interplay between these two different points of views and approaches, for example, using the statistical analysis of correspondence analysis to provide better strategy for selecting cut points and estimating the number of clusters.

4.4 Partitions with overlaps

So far in our discussion, we have only looked at *hard* clustering, i.e., a data object belongs to one and only one cluster. In many situations, especially when there is a large overlap between the clusters, it is more advantageous to allow data objects to belong to different clusters. For example, in document clustering, certain groups of words can be shared by two clusters. Is it possible to model this overlap using our bipartite graph model and yet find efficient approximate solutions? The answer seems to be yes, but our results at this point are rather preliminary and we will only illustrate the possibilities.

Our basic idea is that when computing $\mathrm{Ncut}(A, B)$, we should disregard the contributions of the set of vertices that is in the overlap. More specifically, let $X = A \cup O_X \cup \bar{A}$ and $Y = B \cup O_Y \cup \bar{B}$, where O_X denotes the overlap between the vertex subsets $A \cup O_X$ and $\bar{A} \cup O_X$, and O_Y the overlap between $B \cup O_Y$ and $\bar{B} \cup O_Y$. Compute

$$\mathrm{Ncut}(A, B, \bar{A}, \bar{B}) = \frac{\mathrm{cut}(A, B)}{\mathrm{W}(A, Y) + \mathrm{W}(X, B)} + \frac{\mathrm{cut}(\bar{A}, \bar{B})}{\mathrm{W}(\bar{A}, Y) + \mathrm{W}(X, \bar{B})}.$$

However, we can make $\mathrm{Ncut}(A, B, \bar{A}, \bar{B})$ smaller simply by putting more vertices in the overlap. Therefore, we need to balance these two competing quantities: the size of the overlap and the modified normalized cut $\mathrm{Ncut}(A, B, \bar{A}, \bar{B})$ by minimizing

$$\mathrm{Ncut}(A, B, \bar{A}, \bar{B}) + \alpha(|O_X| + |O_Y|),$$

where α is a regularization parameter. The Mcut case can be treated in the same way. How to find an efficient method for computing the (approximate) optimal solution to the above minimization problem still needs to be investigated. We close this section by presenting an illustrative example showing that, in some situations, the singular vectors already automatically separating the overlap sets while giving the coordinates for carrying out clustering.

EXAMPLE 1. We construct a sparse m-by-n rectangular matrix

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$$

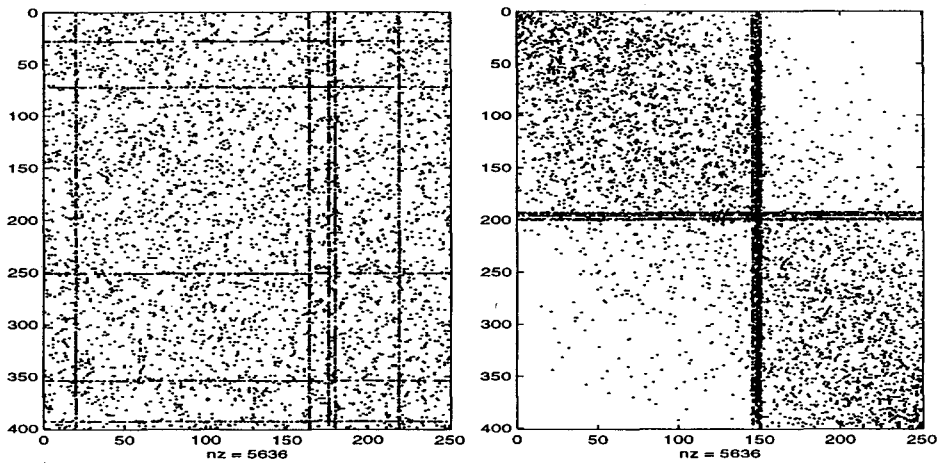

Fig. 4.1. Sparsity patterns of a test matrix before clustering (left) and after clustering (right).

so that W_{11} and W_{22} are relatively denser than W_{12} and W_{21}. We also add some dense rows and columns to the matrix W to represent row and column overlaps. The left panel of Figure 4.1 shows the sparsity pattern of $\bar{W}$, a matrix obtained by randomly permuting the rows and columns of W. We then compute the second largest left and right singular vectors of $D_X^{-1/2}\bar{W}D_Y^{-1/2}$, say $\mathbf{x}$ and $\mathbf{y}$, and sort the rows and columns of $\bar{W}$ according to the values of the entries in $D_X^{-1/2}\mathbf{x}$ and $D_Y^{-1/2}\mathbf{y}$, respectively. The sparsity pattern of this permuted $\bar{W}$ is shown on the right panel of Figure 4.1. As can be seen that the singular vectors not only do the job of clustering, but also concentrate the dense rows and columns at the boundary of the two clusters simultaneously.

4.5 Experiments

In this section we present our experimental results on clustering a data set of newsgroup articles also used in chapter §3. Before applying the clustering algorithms to

the data set, several preprocessing steps need to be considered. Two standard steps are weighting and feature selection. For weighting, we consider a variant of $tf \cdot idf$ weighting scheme, $tf \cdot \log_2(n/df)$, and several other variations listed in [3]. For feature selection, we look at three approaches 1) deleting the terms that occur less than a certain number of times in the data set; 2) deleting the terms that occur in less than a certain number of documents in the data set; 3) selecting the terms according to mutual information of the terms and documents defined in (3.7). In general, we find that the traditional $tf \cdot idf$ based weighting schemes do not improve the performance for SRE. One possible explanation comes from the connection with correspondence analysis. The raw frequencies are the samples of co-occurrence probabilities, and the pre- and post-multiplication by $D_X^{-1/2}$ and $D_Y^{-1/2}$ in $D_X^{-1/2}(D-W)D_Y^{-1/2}$ *automatically* take into account of weighting. We do, however, find that trimming the raw frequencies can sometimes improve performance for SRE, especially for the anomalous cases where some words occur in certain documents an unusual number of times, skewing the clustering process.

We test three variations of SRE methods: 1) SRE(Cut0), which chooses $c_x = 0$ and $c_y = 0$ as the cut points; 2) SRE(Ncut), which computes the cut points using the Ncut method; and 3) SRE(Mcut), which computes the cut points using the Mcut method.

For the purpose of comparison, we consider two other clustering methods: 1) the K-means method [37]; 2) principal direction divisive partition (PDDP) method [7]. The K-means method is a widely used cluster analysis tool. The variant we used employs the Euclidean distance when comparing the similarity between two documents, as is done in (1.1). We also tried the K-means without document length normalization, and the

results are far worse. Therefore we will not report the corresponding results. Since the K-means method is an iterative method, we need to specify a stopping criterion. For the variant we use, we compare the centroids between two consecutive iterations, and stop when the difference is smaller than a pre-defined tolerance.

PDDP is another clustering method that utilizes the singular vectors. In PDDP, the set of documents is normalized and centered first, i.e., let W be the term-document matrix, and $\mathbf{w}$ be the average of the columns of W, then compute the largest singular value triplet $\{\mathbf{u}, \sigma, \mathbf{v}\}$ of $W - \mathbf{we}^T$. The set of documents are partitioned based on the corresponding values of $\mathbf{v} = (v_i)$: one simple scheme is to let those with positive v_i go into one cluster and the rest into another cluster. The whole process is repeated on the term-document matrices of the two sub-clusters, respectively. Although both SRE and PDDP make use of the singular vectors of some versions of the term-document matrices, they are derived from fundamentally different principles: PDDP is a feature-based clustering method, projecting all the data points to a one-dimensional subspace spanned by the first principal axis; SRE is a similarity-based clustering method with two co-occurring variables (terms and documents in the context of document clustering) clustered simultaneously. Unlike SRE, PDDP does not have a well-defined objective function for minimization. It only partitions the columns of the term-document matrices while SRE partitions both of its rows and columns. This will have significant impact on the computational costs. PDDP, however, has an advantage that it can be applied to a data set with both positive and negative values while SRE can only be applied to that with non-negative data values.

Mixture	SRE(Cut0)	SRE(Ncut)	SRE(Mcut)	PDDP	K-means
50/50	92.12 ± 3.52	93.92 ± 4.94	94.85 ± 2.37	91.90 ± 3.19	76.93 ± 14.42
50/100	90.57 ± 3.11	95.86 ± 2.07	95.28 ± 2.51	86.11 ± 3.94	76.74 ± 14.01
50/150	88.04 ± 3.90	96.18 ± 2.98	95.15 ± 3.24	78.60 ± 5.03	68.80 ± 13.55
50/200	82.77 ± 5.24	96.53 ± 2.42	94.14 ± 5.84	70.43 ± 6.04	69.22 ± 12.34

Table 4.1.
Comparison of SRE, PDDP, and K-means (NG1/NG2). Accuracy is in percentage(%).

Mixture	SRE(Cut0)	SRE(Ncut)	SRE(Mcut)	PDDP	K-means
50/50	74.56 ± 8.93	71.36 ± 11.68	73.14 ± 12.11	73.40 ± 10.07	61.61 ± 8.77
50/100	67.13 ± 7.17	66.61 ± 10.25	69.04 ± 10.47	67.10 ± 10.20	64.40 ± 9.37
50/150	58.30 ± 5.99	69.69 ± 6.82	68.68 ± 7.10	58.72 ± 7.48	62.53 ± 8.20
50/200	57.55 ± 5.69	74.26 ± 5.16	71.12 ± 8.29	56.63 ± 4.84	60.82 ± 7.54

Table 4.2.
Comparison of SRE, PDDP, and K-means (NG10/NG11). Accuracy is in percentage(%).

Mixture	SRE(Cut0)	SRE(Ncut)	SRE(Mcut)	PDDP	K-means
50/50	73.66 ± 10.53	61.94 ± 6.89	66.92 ± 8.60	69.52 ± 12.83	62.25 ± 9.94
50/100	67.23 ± 7.84	64.48 ± 9.88	71.44 ± 11.53	67.84 ± 7.30	60.91 ± 7.92
50/150	65.83 ± 12.79	70.34 ± 8.74	74.39 ± 9.39	60.37 ± 9.85	63.32 ± 8.26
50/200	61.23 ± 9.88	72.27 ± 7.53	73.50 ± 8.09	60.76 ± 5.55	64.50 ± 7.58

Table 4.3.
Comparison of SRE, PDDP, and K-means (NG18/NG19). Accuracy is in percentage(%).

	mideast	graphics	space	baseball	motorcycles
cluster 1	87	0	0	2	0
cluster 2	7	90	7	6	7
cluster 3	3	9	84	1	1
cluster 4	0	0	1	88	0
cluster 5	3	1	8	3	92

Table 4.4.
Confusion matrix for newsgroups {NG2, NG9, NG10, NG15, NG18}.

EXAMPLE 2. In this example, we examine binary clustering with both even and uneven clusters. We consider three pairs of newsgroups: NG1 and NG2 which are well-separated; NG10 and NG11; and NG18 and NG19. The latter two have a large overlap. We use document frequency as the feature selection criterion and delete the words that occur in less than 5 documents in each data sets we use. For both the K-means and PDDP, we apply $tf \cdot idf$ weighting scheme together with document length normalization so that each document vector will have Euclidean norm one. For SRE we trim the raw frequency so that the maximum is 10. For each newsgroup pair, we select four types of mixture of articles: m/n indicates that m articles are from the first group and n from the second group. The results are listed in Tables 4.1, 4.2, and 4.3. We randomly sample 100 times and list the means and standard deviations. We should emphasize that the K-means method can only find local minimum, and the results highly depend on the initial values and stopping criteria. This is also reflected by the large standard deviations associated with the K-means method. From the three tests we conclude that both SRE and PDDP outperform the K-means method. The performance of SRE and

PDDP are similar in the balanced mixtures, but SRE is superior to PDDP in the skewed mixtures, especially for SRE(Ncut) and SRE(Mcut) where the results are much better at the expense of higher running time.

EXAMPLE 3. In this example, we consider an easy multi-cluster case. We examine five newsgroups NG2, NG9, NG10, NG15, NG18 which were also considered in [72]. We sample 100 articles from each newsgroups and use mutual information for feature selection. We use the minimum normalized cut to search for the cut points in each recursion. Table 4.4 gives the confusion matrix for one sample. The accuracy for this sample is 88.2%. We also test two other samples with accuracy 85.4% and 81.2%, which compare favorably with those obtained for three samples with accuracy 59%, 58% and 53% reported in [72]. In the following, we list the top few words for each clusters computed by mutual information.

```
Cluster 1:
     armenian israel arab palestinian peopl jew isra
     iran muslim kill turkis war greek iraqi adl call
Cluster 2:
     imag file bit green gif mail graphic colour
     group version comput jpeg blue xv ftp ac uk list
Cluster 3:
     univers space nasa theori system mission henri
     moon cost sky launch orbit shuttl physic work
Cluster 4:
```

```
    clutch year game gant player team hirschbeck
    basebal won hi lost ball defens base run win
Cluster 5:
    bike dog lock ride don wave drive black
    articl write apr motorcycl ca turn dod insur
```

4.6 Concluding remarks

In this chapter, we formulate a class of clustering problems as bipartite graph partitioning problems, and we show that efficient optimal solutions can be found by computing the partial singular value decomposition of some scaled edge weight matrices. We have also shown, however, that there still remain many challenging problems. One area that needs further investigation is the selection of cut points and number of clusters using multiple left and right singular vectors and the possibility of adding local refinements to improve clustering quality. It will be difficult to use local refinement for PDDP because it does not have a global objective function for minimization. Another area is finding efficient algorithms for handling overlapping clusters. Finally, the treatment of missing data under our bipartite graph model, especially when we apply our spectral clustering methods to the problem of data analysis of recommender systems, also deserves further investigation.

Chapter 5

Correspondence analysis as result of spectral cluster analysis

Correspondence analysis (CA) is a method in multivariate statistics to analyze the relationship between two different types of data objects. This technique applies the singular value decomposition (SVD) to the contingency table similar to principal component analysis (PCA). For most cases, CA projects the data objects into a $2-dim$ space. This plot in the reduced dimension space gives a clear graphical view of the data with higher dimensions so that the spatial relationship between different variables can be recognized much more easily than in the original space.

Compared to PCA, however, CA has received much less attention. While PCA is a major topic in almost every book on multivariate statistics, CA is simply ignored in many books. It appears that the standard descriptions of CA emphasize the geometric interpretations whose statistical significances are not immediately clear, while the differences or the advantages of CA over PCA have not been effectively articulated.

In this chapter, we provide a new view of CA based on the cluster analysis of two-way contingency table data. It is shown that CA is the direct results of clustering row and column objects simultaneously using a bi-clustering method (§4) based on the Mcut method (§3).

This new bi-clustering point of view leads to the following important results. 1) It emphasizes the "correspondence" nature of the two-way contingency data, the

relationship between row objects defined or characterized by their representations in column objects, and vice versa. This "duality" relationship between columns and rows (or those between words and documents) has been emphasized in information retrieval [70, 19]. 2) A similarity measure between the same-type objects is naturally defined based on probabilistic associations of the non-negative contingency table data. It is this similarity that defines the clusters that the clustering procedure is trying to discover. 3) It gives a precise explanation of the ordination between the same-type and between the different-type objects. 4) It provides a concrete and effective procedure for clustering.

5.1 CA — a slight variation of PCA?

Given a contingency table P with m rows and n columns, we can calculate the principal components, i.e., singular value decomposition, as

$$P = \sum_{k=1}^{h} \mathbf{u}_k \lambda_k \mathbf{v}_k^T$$

where h is the rank of matrix P, $h \leq \min(m,n)$. Note here, we did not center the data (i.e., subtract the mean) because typically the table entries are the counts of occurrences; $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_h$ are principal components for row objects, and $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_h$ are for column objects.

Correspondence analysis of P proceeds considerably the same way, although typical descriptions of CA [38, 4] are somewhat complicated so that the main points are not

explained very clearly. Write

$$P = D_r^{1/2}(D_r^{-1/2}PD_c^{-1/2})D_c^{1/2} = D_r^{1/2}\left(\sum_{k=1}^{h}\hat{\mathbf{u}}_k\hat{\lambda}_k\hat{\mathbf{v}}_k^T\right)D_c^{1/2}$$
$$= \sum_{k=1}^{h}(D_r^{1/2}\hat{\mathbf{u}}_k)\hat{\lambda}_k(D_c^{1/2}\hat{\mathbf{v}}_k)^T$$

where $D_r = \text{diag}(\mathbf{r})$, $\mathbf{r} = (p_{.1}, p_{.2}, \cdots, p_{.n})^T$, $p_{.i} = \sum_i p_{ij}$, and $D_c = \text{diag}(\mathbf{c})$, $\mathbf{c} = (p_{1.}, p_{2.}, \cdots, p_{m.})^T$, $p_{i.} = \sum_j p_{ij}$. Then CA is the same as PCA, except in CA, the SVD is applied to the scaled contingency table

$$\widehat{P} = D_r^{-1/2}PD_c^{-1/2} = \sum_{k=1}^{h}\hat{\mathbf{u}}_k\hat{\lambda}_k\hat{\mathbf{v}}_k^T, \tag{5.1}$$

instead on P directly. The largest singular value of $\widehat{P}$ and its corresponding singular vectors are

$$\hat{\lambda}_1 = 1,\ \hat{\mathbf{u}}_1 = D_r^{1/2}\mathbf{e}_m/p_{..}^{1/2},\ \hat{\mathbf{v}}_1 = D_c^{1/2}\mathbf{e}_n/p_{..}^{1/2} \tag{5.2}$$

where $p_{..} = \sum_{ij} p_{ij} = ||D_r^{1/2}||_2^2 = ||D_c^{1/2}||_2^2$, and $\mathbf{e}_m$ is a vector of all 1's with dimension m. Now, we have the familiar form

$$P = \mathbf{r}\mathbf{c}^T/p_{..} + \sum_{k=2}^{h} D_r(D_r^{-1/2}\hat{\mathbf{u}}_k)\hat{\lambda}_k(D_c^{-1/2}\hat{\mathbf{v}}_k)^T D_c.$$

Note that, in standard CA, the contingency table is scaled such that $p_{..} = 1$. Here we leave open the option that one directly works on the contingency table without such an

overall scaling. Everything works equally well. The key point here is that $\widehat{P}$ is invariant with regard to this overall scaling.

In CA, an important difference from PCA is that the "principal components" are $D_r^{-1/2}\widehat{\mathbf{u}}_k$ and $D_c^{-1/2}\widehat{\mathbf{v}}_k$. In the $2-dim$ CA plot, column objects are represented by the principal coordinates $(\widehat{\lambda}_2 D_c^{-1/2}\widehat{\mathbf{v}}_2, \quad \widehat{\lambda}_3 D_c^{-1/2}\widehat{\mathbf{v}}_3)$, and row objects are represented by the principal coordinates $(\widehat{\lambda}_2 D_r^{-1/2}\widehat{\mathbf{u}}_2, \quad \widehat{\lambda}_3 D_r^{-1/2}\widehat{\mathbf{u}}_3)$. Here the scale of each dimension is set to be the corresponding singular value $\widehat{\lambda}_k$, the same as in PCA. Note that we use $k = 2, 3, \cdots, h$ (instead of $k = 1, 2, \cdots, h-1$ in usual CA formulations) both to emphasize the connection to PCA and to be consistent with the clustering framework discussed later in this chapter.

Two critical questions regarding CA are: 1) why use the scaled contingency table $\widehat{P}$, in stead of the original one? 2) why use $D_r^{-1/2}\widehat{\mathbf{u}}_k$ and $D_c^{-1/2}\widehat{\mathbf{v}}_k$ as the "principal components", instead of the seemingly more direct $D_r^{1/2}\widehat{\mathbf{u}}_k$ and $D_c^{1/2}\widehat{\mathbf{v}}_k$? The main contribution of this chapter is to show that: 1) $\widehat{P}$ defines a more consistent similarity metric, and it arises naturally in the clustering procedure; 2) $D_r^{-1/2}\widehat{\mathbf{u}}_k$ and $D_c^{-1/2}\widehat{\mathbf{v}}_k$ are the better components (indicator vectors) for the clustering.

5.1.1 Scaled principal components

The above scaled contingency table approach is in fact a generalization of a *scaled principal components analysis* (SPCA) approach. For a symmetric similarity matrix S with all non-negative elements, we perform the following expansion

$$S = D^{1/2}(D^{-1/2}SD^{-1/2})D^{1/2} = D^{1/2}\left(\sum_{k=1}^{h}\widehat{\mathbf{u}}_k\widehat{\lambda}_k\widehat{\mathbf{u}}_k^T\right)D^{1/2}$$

where $D = \text{diag}(\mathbf{d})$ and $\mathbf{d} = (s_{.1}, s_{.2}, \cdots, s_{.m})^T$. In SPCA, the SVD is applied to the scaled covariance matrix $\widehat{S} = D^{-1/2}SD^{-1/2} = \sum_{k=1}^{h} \widehat{\mathbf{u}}_k \widehat{\lambda}_k \widehat{\mathbf{u}}_k^T$, instead on S directly. The largest eigenvalue of $\widehat{S}$ and its corresponding eigenvector are $\widehat{\lambda}_1 = 1$ and $\widehat{\mathbf{u}}_1 = D^{1/2}\mathbf{e}_m / s_{..}^{1/2}$, respectively, where $s_{..} = \sum_{ij} s_{ij}$. Then we have the standard SPCA form

$$S = \mathbf{d}\mathbf{d}^T / s_{..} + \sum_{k=2}^{h} D(D^{-1/2}\widehat{\mathbf{u}}_k)\widehat{\lambda}_k (D^{-1/2}\widehat{\mathbf{u}}_k)^T D.$$

In SPCA, the "principal components" are $D^{-1/2}\widehat{\mathbf{u}}_k$. Following we will see that $D^{-1/2}\widehat{\mathbf{u}}_k$ are actually the indicator vectors used to identify the clusters in a cluster method.

5.1.2 Mcut and SPCA

The Mcut method introduced in §3 is used to partition the weighted graph $G(V, E)$ with weight matrix W, such that the Mcut value (1.8) is minimized. This minimization problem is equivalent to minimizing

$$J_M = \frac{\mathbf{q}^T(D - W)\mathbf{q}}{\mathbf{q}^T W \mathbf{q}}, \tag{5.3}$$

where $D = \text{diag}(W\mathbf{e})$. Relaxing the indicator vector $\mathbf{q}$ from two discrete values $\{a, -b\}$ to the continuous values in $[-1, 1]$, and maintaining the positivity of $\mathbf{q}^T W \mathbf{q}$, the solution to the minimization problem can be transformed to

$$(D - W)\mathbf{q} = \zeta W \mathbf{q}. \tag{5.4}$$

Clearly, the eigenvector $\mathbf{q}_1 = \mathbf{e}$ associated with $\zeta_1 = 0$ is a trivial solution and does not satisfy our requirement. The eigenvector $\mathbf{q}_2$ associated with the second smallest eigenvalue ζ_2 satisfies the orthogonal condition $\mathbf{q}_2^T W\mathbf{e} = \mathbf{q}_2^T D\mathbf{e} = 0$, hence provides a good linear search heuristic.

(5.4) can be rewritten as the generalized eigensystem

$$(D - W)\mathbf{q} = \frac{\zeta}{1+\zeta} D\mathbf{q} \tag{5.5}$$

or

$$\widehat{W}\widehat{\mathbf{z}} = (D^{-1/2} W D^{-1/2})\widehat{\mathbf{z}} = \lambda\widehat{\mathbf{z}}, \qquad \widehat{\mathbf{z}} = D^{1/2}\mathbf{q} \quad \text{and} \quad \lambda = 1/(1+\zeta). \tag{5.6}$$

This equation can be further written as $(I - \widehat{W})\widehat{\mathbf{z}} = (1-\lambda)\widehat{\mathbf{z}}$, and $I - \widehat{W}$ is called the normalized Laplacian matrix of a weighted graph.

Now the connection to SPCA is clear. Replacing the weight matrix W by the similarity or covariance matrix S, $\widehat{\mathbf{z}}_i = \widehat{\mathbf{u}}_i$ are exactly the eigenvectors of $\widehat{S}$, and the indicator vectors for the clustering are $\mathbf{q} = D^{-1/2}\widehat{\mathbf{u}}_i$, the principal components in SPCA.

Following we prove that CA is a natural generalization of the SPCA framework to asymmetric matrix, the two-way contingency table. We begin by introducing bi-clustering using the Mcut method.

5.2 Bi-clustering using Mcut

An $m \times n$ contingency table can be represented by a weighted bipartite graph $G(R, C, E)$ shown in Figure 1.1. Each row object denotes an r-type data object and

each column object denotes a c-type data object. (Usually a row stands for a sample, an event, and a column stands for a variable, an attribute.) A non-zero entry p_{ij} in the contingency table records the count of co-occurrences of row object r_i and column object c_j, and is represented by a weighted edge between r_i and c_j in the bipartite graph. There is no edge between the same-type data objects. Since the table entries are integers, An incidence matrix can be generated based on the bipartite graph, as is done in [54, 47].

Assume there are two types of data objects, R and C. Bi-clustering (also called co-clustering [18]) on a bipartite graph $G(R, C, E)$ is a method to cluster two types of data objects simultaneously, based on the relationship provided by the 2-way contingency table, such that the cluster of one data type has larger connection with the cluster of another data type. Here we apply the Mcut method to the bipartite graph and show that the solution to this graph partitioning problem is exactly those indicator vectors in CA.

For this purpose, the indicator variables are represented by two sets: $\mathbf{f}$ and $\mathbf{g}$. $\mathbf{f}$ determines the split of R into R_1 and R_2, and $\mathbf{g}$ determines the split of C into C_1 and C_2. Suppose we have found the optimal splits based on $\mathbf{f}$ and $\mathbf{g}$, then we can write the contingency table as

$$P = \begin{pmatrix} P_{R_1,C_1} & P_{R_1,C_2} \\ P_{R_2,C_1} & P_{R_2,C_2} \end{pmatrix}. \tag{5.7}$$

The objective function (1.8) becomes

$$\mathrm{Mcut}(C_1, C_2; R_1, R_2) = \frac{W(P_{R_1,C_2}) + W(P_{R_2,C_1})}{W(P_{R_1,C_1})} + \frac{W(P_{R_1,C_2}) + W(P_{R_2,C_1})}{W(P_{R_2,C_2})}, \tag{5.8}$$

where $W(P_{R_1,C_2}) = \sum_{i \in R_1, k \in C_2} p_{ik}$ is the cut size between R_1 and C_2, and $W(P_{R_2,C_1})$ is the cut size between R_2 and C_1. $W(P_{R_1,C_1})$, $W(P_{R_2,C_2})$ are the similarities within the two clusters (Figure 1.1). Note that the minimization of the Mcut value implies a smaller value for J_M in (5.3), which in turn implies a smaller value for ζ and a large value for λ.

Now we wish to solve the eigensystem (5.6). We need to construct a symmetric weight matrix W from the rectangle matrix P. The standard construction [36, 46, 82, 18] is

$$W = \begin{pmatrix} 0 & P \\ P^T & 0 \end{pmatrix}. \tag{5.9}$$

Then $D = \mathrm{diag}(D_r, D_c)$. $\widehat{\mathbf{z}}$ can be represented by the indicator vectors $\mathbf{f}, \mathbf{g}$ as

$$\widehat{\mathbf{z}} = \begin{pmatrix} \widehat{\mathbf{x}} \\ \widehat{\mathbf{y}} \end{pmatrix} = \begin{pmatrix} D_r^{1/2}\mathbf{f} \\ D_c^{1/2}\mathbf{g} \end{pmatrix}. \tag{5.10}$$

Putting all these into (5.6), we have

$$\widehat{P}\widehat{\mathbf{y}} = \lambda \widehat{\mathbf{x}}, \quad \widehat{P}^T \widehat{\mathbf{x}} = \lambda \widehat{\mathbf{y}}. \tag{5.11}$$

The solutions to (5.11) are the SVD of $\widehat{P}$ in (5.1), with

$$\widehat{\mathbf{x}} = \widehat{\mathbf{u}}_k, \quad \widehat{\mathbf{y}} = \widehat{\mathbf{v}}_k, \quad \lambda_k = \widehat{\lambda}_k, \qquad k = 1, 2, \cdots, h.$$

Another set of solutions is $\widehat{\mathbf{x}} = \widehat{\mathbf{u}}_k, \quad \widehat{\mathbf{y}} = -\widehat{\mathbf{v}}_k, \quad \lambda_k = -\widehat{\lambda}_k$. Since $\widehat{\lambda}_k \geq 0$, $\zeta = 1/\lambda - 1$ is negative, i.e., the minimum value of J_M is negative. Therefore this set of solutions is not useful. It is easy to prove that the singular values λ_k of $D_r^{-1/2} P D_c^{-1/2}$ are less than or equal to 1. The largest singular triplet is given in (5.2). They are simply the centroids of row and column objects. From (5.10), the optimal indicator vectors for the cluster analysis are

$$\mathbf{f}_k = D_r^{-1/2} \widehat{\mathbf{u}}_k, \quad \mathbf{g}_k = D_c^{-1/2} \widehat{\mathbf{v}}_k, \tag{5.12}$$

which are the exact solutions to CA.

Once $\mathbf{f}_k, \mathbf{g}_k$ are available, we form a single indicator vector $\mathbf{q}_k = \begin{pmatrix} \mathbf{f}_k \\ \mathbf{g}_k \end{pmatrix}$, then sort $\mathbf{q}_2$ to provide a linear order for both r-type and c-type objects. Using the Mcut algorithm, we choose as the cut point i_{cut} the one at which the minimum Mcut value is obtained. The data objects on one side of the cut point belong to one cluster and those on the other side belong to another cluster. This automatically splits the c-type objects into two parts

$$C_1 = \{c_i \mid i \leq i_{cut}\}, \quad C_2 = \{c_i \mid i > i_{cut}\},$$

and the r-type objects into two parts as well

$$R_1 = \{r_i \mid i \leq i_{cut}\} \quad R_2 = \{r_i \mid i > i_{cut}\}.$$

This gives the optimal bi-clustering result of the contingency table data objects.

5.3 Correspondence analysis

Now we can answer the critical questions raised in section 5.1. First, $\widehat{P}$ arises naturally from the clustering procedures (5.7, 5.9). Second, from (5.12), we see that it is $\mathbf{f}_k = D_r^{-1/2}\widehat{\mathbf{u}}_k$ and $\mathbf{g}_k = D_c^{-1/2}\widehat{\mathbf{v}}_k$, not $D_r^{1/2}\widehat{\mathbf{u}}_k$ and $D_c^{1/2}\widehat{\mathbf{v}}_k$ that determine the optimal clustering of row and column objects. This is the main advantage of CA over PCA.

It is important to note that, both the scales and signs of $\mathbf{f}_k$ and $\mathbf{g}_k$ are automatically matched because they are the integral parts of the solution to one equation. That is the reason why we can combine $\mathbf{f}_k$ and $\mathbf{g}_k$ to form a single vector $\mathbf{q}_k$, and plot $(\mathbf{q}_2, \mathbf{q}_3)$ in a $2 - dim$ space.

Sorting $\mathbf{f}_2$ provides an optimal order for the r-type objects, and sorting $\mathbf{g}_2$ provides an optimal order for the c-type objects. The exact meaning of "optimal order" will be explored in section 5.4.1. With these optimal orders, the rows and columns of the contingency table are re-arranged, which essentially groups the data objects with high associations.

Let $U = (\mathbf{u}_2, \mathbf{u}_3, \cdots, \mathbf{u}_h)$ and $V = (\mathbf{v}_2, \mathbf{v}_3, \cdots, \mathbf{v}_h)$. The orthonormal properties of SVD give $U^T U = I_{h-1}$ and $V^T V = I_{h-1}$. Now let $A = D_r^{1/2} U$ and $B = D_c^{1/2} V$.

We have

$$A^T D_r^{-1} A = I_{h-1}, \quad B^T D_c^{-1} B = I_{h-1},$$

the same as (4.1.9) in [38]. Let $F = (\mathbf{f}_2, \mathbf{f}_3, \cdots, \mathbf{f}_h) = D_r^{-1/2} U$ and $G = (\mathbf{g}_2, \mathbf{g}_3, \cdots, \mathbf{g}_h) = D_c^{-1/2} V$. We have

$$F^T D_r F = I_{h-1}, \quad G^T D_r G = I_{h-1},$$

the same as (4.1.18) in [38]. (Note that our $F\Lambda$ and $G\Lambda$ correspond to F and G in [38], where $\Lambda = \mathrm{diag}(\lambda_2, \cdots, \lambda_h)$, the same as D_μ in [38]). With these notations, we rewrite the CA in a simple form

$$P - \mathbf{r}\mathbf{c}^T / p_{..} = \sum_{k=2}^{h} D_r \mathbf{f}_k \lambda_k \mathbf{g}_k^T D_c = D_r F \Lambda G^T D_c. \tag{5.13}$$

An interesting and useful known result in CA is Pearson's chi-square test, $\chi = \sum_{ij} (p_{ij} - e_{ij})^2 / e_{ij}$, where e_{ij} is the expected count assuming the rows and columns are independently distributed, i.e., $e_{ij} = p_{i.} p_{.j} / p_{..} = r_i c_j / p_{..}$. Using (5.13), we have

$$\begin{aligned}
\chi^2 &= p_{..} \sum_{ij} [r_i^{1/2} (F\Lambda G^T)_{ij} c_j^{1/2}]^2 \\
&= p_{..} Tr[(D_r^{1/2} F \Lambda G^T D_c^{1/2})^T (D_r^{1/2} F \Lambda G^T D_c^{1/2})] \\
&= p_{..} \sum_{k=2}^{h} \lambda_k^2.
\end{aligned}$$

For the contingency table, the number of degrees of freedom is $(m-1)(n-1)$, which can be used to judge if the row and column distributions are independent. Since larger λ_k implies smaller ζ_k (cf. (5.6)), hence better clustering results, the good clustering results imply large deviation from the row-column independence. Random distributions have no interesting patterns and the good clustering results imply highly correlation patterns. These intuitions are made precise and quantitative in our cluster analysis.

5.3.1 Reciprocal averaging

Hill [47] emphasized the ordination (also called seriation) point of view of CA by using reciprocal averaging. The basic idea was first developed in plant ecology and archaeology, the so called "gradient analysis" that aimed at determining an order of plant preference to certain conditions. It is an iterative procedure: start with an initial order $\mathbf{g}^{(0)}$, usually an "educated guess", for row objects, and calculate an order $\mathbf{f}^{(0)}$ for column objects, which in turn generates a better order $\mathbf{g}^{(1)}$ for row objects. This procedure can be expressed as

$$f_i^{(t)} = \lambda^{-1} \sum_j (p_{ij}/p_{i\cdot}) g_j^{(t)}, \quad g_j^{(t+1)} = \lambda^{-1} \sum_i (p_{ij}/p_{\cdot j}) f_i^{(t)},$$

where t is the iteration step. When the convergence is reached, we have

$$\mathbf{f} = \lambda^{-1} D_r^{-1} P \mathbf{g}, \quad \mathbf{g} = \lambda^{-1} D_c^{-1} P^T \mathbf{f},$$

which can be equivalently written as

$$(D_r^{1/2}\mathbf{f}) = \lambda^{-1}\widehat{P}(D_c^{1/2}\mathbf{g}), \quad (D_c^{1/2}\mathbf{g}) = \lambda^{-1}\widehat{P}^T(D_r^{1/2}\mathbf{f}). \tag{5.14}$$

(5.14) is identical to (5.10, 5.11). Therefore the ordination or seriation scores computed in reciprocal averaging are precisely the "principal components" of CA.

5.4 Ordination of objects

5.4.1 Ordination of objects with different types

But what is exactly the nature of the ordination behind the gradient analysis? Although full of ecologist intuitions, a precise definition or explanation is still lacking. Using the cluster analysis described here, we can give a concrete explanation of the nature of the ordination. As noted earlier, in CA, the principal components $\mathbf{f}_k$ and $\mathbf{g}_k$ have the same scales and signs, therefore can be mixed to determine the ordination. Second, note that $\sum_{ij}(q_i - q_j)^2 w_{ij} = \sum_{ij} 2(q_i^2 - q_i q_j)w_{ij} = \sum_i 2q_i^2 \sum_j w_{ij} - \sum_i \sum_j 2q_i q_j w_{ij} = 2\mathbf{q}^T(D - W)\mathbf{q}$. The solution to (1.8) is equivalent to minimizing the following function

$$J(\mathbf{q}) = \frac{\mathbf{q}^T(D-W)\mathbf{q}}{\mathbf{q}^T D\mathbf{q}} = \frac{\sum_{ij}(q_i - q_j)^2 w_{ij}}{2\mathbf{q}^T D\mathbf{q}}. \tag{5.15}$$

For a large w_{ij}, $(q_i - q_j)^2$ should be small in order to minimize $J(\mathbf{q})$. Hence the data objects q_i, q_j with large w_{ij} will be close to each other in the sorted order. This is the nature of the ordination.

Now $\mathbf{q}$ has two types of variables, $\mathbf{f}$ and $\mathbf{g}$, and

$$J(\mathbf{q}) = \frac{\sum_{ij}(f_i - g_j)^2 p_{ij}}{\mathbf{f}^T D_r \mathbf{f} + \mathbf{g}^T D_c \mathbf{g}},$$

which means there exists an ordination between the r-type and c-type variables. When p_{ij} is large, the corresponding r_i, c_j values are close, i.e., they are close in the $2-dim$ CA plot. It shows that correspondence analysis captures the essential correlations between r-type and c-type variables, reflecting and quantifying our intuition that a large p_{ij} implies a large joint probability between r_i and c_j.

5.4.2 Ordination of objects with the same type

What is the ordination between the same type of objects, say two r-type objects? Their ordering is determined by their relative ordination with the c-type variables. Before proceeding, we define the similarity between two data objects of the same type.

5.4.2.1 Similarity metrics

The similarity or association between two row objects r_i, r_j is defined as the probability accumulation. We interpret p_{ik} as the probability of the association between row object r_i and column object c_k (joint probability of occurrence of r_i and c_k). The probability of the association between c_k and r_j is p_{jk}. Thus, the probability of the association between r_i and r_j through c_k is $p_{ik}p_{jk}$. The total probability of the association is the

sum over all column objects,

$$\text{sim}(r_i, r_j) = \sum_{k=1}^{n} p_{ik} p_{jk} / p_{\cdot k} = (P \underset{c}{D^{-1}} P^T)_{ij}, \tag{5.16}$$

where each column is inversely weighted by its marginal probability $p_{\cdot k}$. If the entries of a particular column are very large, this column will unduly dominate the summation. The inverse $p_{\cdot k}$ weight is introduced to prevent this situation from happening. $S^{rr} = P \underset{c}{D^{-1}} P^T$ is the row-row similarity matrix with each entry as the similarity of corresponding row data objects. The similarity or association between two column objects c_k, c_l can be similarly defined as

$$\text{sim}(c_k, c_l) = \sum_{i=1}^{m} p_{ik} p_{il} / p_{i\cdot} = (P^T \underset{r}{D^{-1}} P)_{kl}, \tag{5.17}$$

where each row is inversely weighted by its marginal probability $p_{i\cdot}$. $S^{cc} = P^T \underset{r}{D^{-1}} P$ contains the similarities between all pairs of column objects. With this definition, the self-similarities $\text{sim}(r_i, r_i) \neq 1$. To properly judge the magnitude of the similarity, we sometimes use the similarity coefficients

$$\rho^{rr}_{ij} = \frac{\text{sim}(r_i, r_j)}{\sqrt{\text{sim}(r_i, r_i)}\sqrt{\text{sim}(r_j, r_j)}} = \frac{(\hat{P} \underset{c}{D^{-1}} \hat{P}^T)_{ij}}{(\hat{P} \underset{c}{D^{-1}} \hat{P}^T)^{1/2}_{ii} (\hat{P} \underset{c}{D^{-1}} \hat{P}^T)^{1/2}_{jj}}$$

between two r-type objects, and

$$\rho^{cc}_{kl} = \frac{(\hat{P}^T \underset{r}{D^{-1}} \hat{P})_{kl}}{(\hat{P}^T \underset{r}{D^{-1}} \hat{P})^{1/2}_{kk} (\hat{P}^T \underset{r}{D^{-1}} \hat{P})^{1/2}_{ll}}$$

between two c-type objects.

An important property of these similarity matrices is that, the sum of the ith row of S^{rr} is

$$\begin{aligned}(S^{rr}\mathbf{e}_m)_i &= \sum_j S^{rr}_{ij} = \sum_j \sum_k p_{ik}(1/p_{.k})p_{jk} \\ &= \sum_k p_{ik}(1/p_{.k}) \sum_j p_{jk} = \sum_k p_{ik} = p_{i.}\end{aligned}$$

which is $(D_r)_{ii}$, and similarly for S^{cc}. This property guarantees that the associations between the same-type objects in CA are correctly maintained, as if the data objects are clustered separately based on their own data types.

5.4.2.2 Ordination

Now we return to the ordination of the data objects with the same data type. Suppose two c-type data objects c_i and c_j are both close to some r-tpye data objects r_k, then c_i, c_j should not be too far away. Sum over all possible "middle man" r_k, the relationship between c_i and c_j is determined by their similarity defined in (5.17). Thus, the association between two variables or between two samples produced in correspondence analysis is governed by their similarity. We call this ordination "duality induced" ordination, to differentiate it from the direct ordination for the symmetric graphs.

More precisely, this duality induced ordination follows the similarity between two row objects defined in (5.16). To see this point more clearly, note that (5.14), which determines the indicator vector $\mathbf{f}$, can be written as $(\underset{r}{D^{-1/2}} P \underset{c}{D^{-1}} P^T \underset{r}{D^{-1/2}})\mathbf{x} = \lambda\mathbf{x}$,

the same as

$$S^{rr}\mathbf{f} = \lambda D_r \mathbf{f}. \tag{5.18}$$

Using (5.18), the second largest eigenvector of (5.4) can be obtained by minimizing the following objective

$$J(\mathbf{q}) = \frac{\mathbf{f}^T(D_r - S^{rr})\mathbf{f}}{\mathbf{f}^T D_r \mathbf{f}} = \frac{\sum_{ij}(f_i - f_j)^2 S^{rr}_{ij}}{\mathbf{f}^T D_r \mathbf{f}}.$$

f_i and f_j with a large S^{rr}_{ij} are forced to be close in order to minimize the objective. After sorting $\mathbf{f}$, they will stay near each other.

5.5 Examples

5.5.1 Hair color and eye color

The classic example analyzed by Fisher, the hair color - eye color example, is listed in Table 5.1. The similarity coefficients are listed in Table 5.2 and the CA plot is shown in Figure 5.1. In Figure 5.1, we see one clear cluster in the right upper part, roughly corresponding to the upper left part of the contingency table. Another cluster of three points is in the upper left part of Figure 5.1, roughly corresponding to the lower right part of the contingency table. The third cluster of two points, medium hair and medium eyes, is in the middle of the contingency table. The global structure of this example is in fact similar to the above analysis.

On associations between different-type objects, three high similarity pairs are seen: light eye, fair hair; medium eye, medium hair; and dark eye, dark hair. Each pair is both close in CA plot and also has large joint probability of occurrence in the contingency table. On associations between the same-type objects, "blue eye" is very close to "light eye" in CA plot. Their similarity coefficient (Table 5.2) is very high, 0.99. The ordination is also quite clear. The order of the data objects is determined in CA such that the farther away the objects are in that order, the less similar they are. This is clearly reflected in the similarity coefficient table: the farther away from the diagonal, the smaller the row-row and column-column similarities.

	Hair				
Eye	Fair	Red	Medium	Dark	Black
Light	688	116	584	188	4
Blue	326	38	241	110	3
Medium	343	84	909	412	26
Dark	98	48	403	681	85

Table 5.1.
Hair color and eye color

5.5.2 Document clustering

A simple Bell Labs tech memo [17] provides a more interesting situation. Here, we illustrate the difference between CA and PCA (more precisely, latent semantic indexing

	Fair	Red	Medium	Dark	Black	**Hair**
	1	0.95	0.83	0.52	0.26	Fair
Light	1	1	0.94	0.71	0.48	Red
Blue	0.99	1	1	0.80	0.57	Medium
Medium	0.85	0.84	1	1	0.95	Dark
Dark	0.52	0.55	0.78	1	1	Black
Eye	Light	Blue	Medium	Dark		

Table 5.2.
Similarity coefficients cross table. Lower left half for eye color and upper right half for hair color.

[17]) as popularly done in the information retrieval community. (This example is also in [5]). The tech memo is listed in Table 5.3.

Table 5.4 shows the contingency table with columns and rows ordered by CA. The similarity coefficients are listed in Table 5.5.

Look at the global structure first. Columns c_4, c_1, c_3 appear to form a cluster, and m_1, m_2, m_3, m_4 appear to form another cluster; c_2 must be in the middle of the columns because it has non-zero similarity with both clusters. Being most similar to c_2, c_5 must stay next to c_2.

On finer scale structures, note first that c_5 has to be located left of c_2 because c_5 has a non-zero similarity with cluster $\{c_4, c_3, c_3\}$ and zero similarity with cluster $\{m_1, m_2, m_3, m_4\}$. Also, m_4 must be located left of m_3 because m_4 has non-zero similarity with c_2. Most intriguing is that c_4 is located left of c_1, even though c_4 has a larger similarity (0.56) to c_3 than c_1 does (0.32). However, note that c_4 has a smaller similarity (0.22) to c_2 than c_1 does (0.25) due to the normalization effects so that the word *system* has a smaller weight than *computer*, although it may seem that c_4 should have a larger

Titles:
c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Table 5.3.
Bell Labs tech memo

similarity with c_2 since c_4 contains a "2" instead of the corresponding "1" in c_3. It seems the global consideration in minimizing (5.15) favors the difference in long-range correlations (c_4, c_2) and (c_1, c_2) instead of the short-range correlations (c_4, c_1) and (c_1, c_1). If we modify the data by replacing "2" with "3", then c_4 will be located right of c_1 because both short- and long-range correlations favor this order. In Figure 5.2, we show CA and PCA analysis of the tech memo example. In Figure 5.2(a), words are intermingled with documents. The words *human*, *EPS*, *interface*, and *system* are closely mingled with documents c_4, c_1, c_3. Indeed, they belong to the same cluster using the Mcut algorithm (see below). The word *user* is very close to document c_2 and the words *response* and *time* (they completely overlap) are very close to document c_5. The words *survey*, *graph*, and *trees* are very close to documents m_4, m_3, m_2, m_1.

In Figure 5.2(b), we plot documents only, in order to facilitate the comparison with PCA analysis in Figure 5.2(c) and 5.2(d); m_1, m_2, m_3, m_4 forms a cluster, and documents

	c_4	c_1	c_3	c_5	c_2	m_4	m_3	m_2	m_1
human	1	1							
EPS	1		1						
interface		1	1						
system	2		1		1				
computer		1			1				
user			1	1	1				
response				1	1				
time				1	1				
survey					1	1			
minors						1	1		
graph						1	1	1	
trees							1	1	1

Table 5.4.
Contingency table

c_4, c_3, c_1 form another cluster. Documents c_2, c_5 form the third cluster. These trends are not as clear in the PCA plot. In Figure 5.2(c), using the first and second principal components, one might say there are 2 clusters; one consists of $m_1, \cdots, m_4$, and another consists of $c_1, \cdots, c_5$. It is not apparent that c_2, c_5 belong to the same cluster. In Figure 5.2(d), we use the second and third principal components. The results are qualitatively similar to Figure 5.2(b). The three-cluster structure is more visible, although m_1 is a bit closer to c_1 than to its own cluster.

All these analyses can be performed automatically using the Mcut clustering algorithm described in section 5.2. First, the contingency table is divided into 2 blocks, one consisting of the lower-right block in Table 5.4, and the other consisting of the upper-left block including $c_1, \cdots, c_5$, which is then further split into 2 clusters, leading to the

c_4	1								
c_1	0.28	1							
c_3	0.56	0.32	1						
c_5			0.22	1					
c_2	0.22	0.25	0.28	0.71	1				
m_4					0.26	1			
m_3						0.66	1		
m_2						0.35	0.75	1	
m_1							0.53	0.70	1
	c_4	c_1	c_3	c_5	c_2	m_4	m_3	m_2	m_1

Table 5.5.
Similarity matrix

three-cluster structure in Table 5.4. The essential feature is the clear correspondence between words and documents.

5.6 Concluding remarks

In this chapter, we provide a new interpretation of correspondence analysis from the point of view of data clustering using a bipartite graph to represent the two-way contingency table. The key result of this work is that the principal components in CA are the indicator variables used for clustering the row and column objects in the min-max cut bi-clustering algorithm. This clustering framework further provides an automatic and effective clustering procedure to analyze the two-way contingency table.

We analyze the contingency table from a bipartite graph model and propose a similarity metric based on probabilistic accumulation of associations. This similarity defines the clusters that the clustering procedure is seeking. Furthermore, we give a

precise explanation of the ordination or seriation of CA as emphasized by Hill: In terms of similarity coefficient matrix, as we move away from the diagonal elements, the similarity steadily decreases.

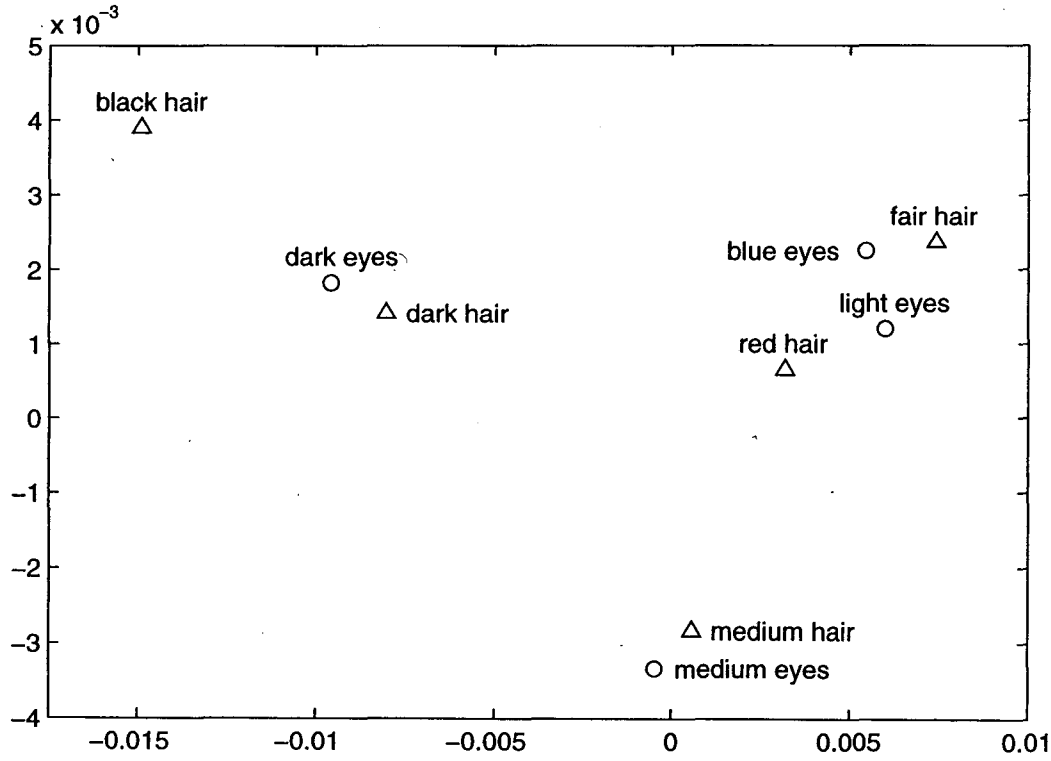

Fig. 5.1. 2D CA plot of the hair color and eye color example.

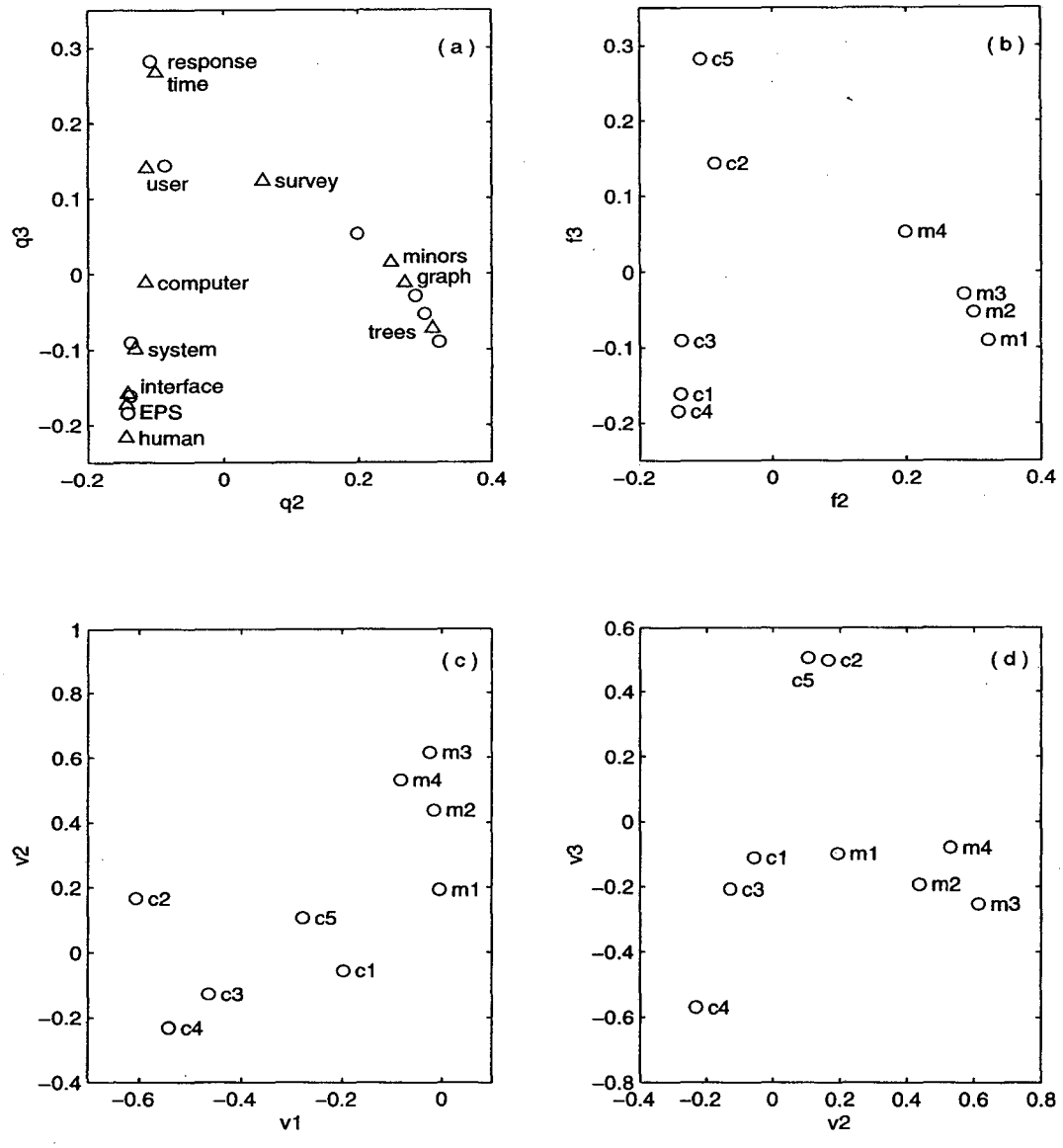

Fig. 5.2. Display of tech memo. (a) Apply CA analysis to both words (circles) and documents (triangles). (b) Apply CA analysis to documents only. (c) Apply PCA to documents, first principal component vs second principal component. (d) Apply PCA to documents, second principal component vs third principal component.

Chapter 6

Conclusions

The task of organizing the large amounts of data becomes increasingly important in the digital world in which Internet access is becoming available almost everywhere and the digital data is generated, in many cases, at an exponential rate. To make things worse, many data generated at such a high rate have no apparent order, making it difficult for the user to enjoy the abundance of information.

Data clustering is a useful approach for analyzing the patterns of the observed data objects. Among different data clustering methods, the spectral graph partitioning methods are very effective tools used to reveal the relations hidden in the seemingly chaotic data objects and restore the order of the data. This naturally leads us to consider applying the spectral methods to organize the Internet world; §2 is an effort to carry out this task.

In §2, the idea of *normalized cut* criterion is borrowed from the image segmentation field to cluster the web documents of the World Wide Web. The normalized cut method utilizes the underlying latent human annotation of the WWW, the hyperlink structure, to group the web documents into meaningful topics. The normalized cut method gives good linear order to search for the approximate optimal cut point, which reduces the combinatoric nature of the problem to a linear one with good mathematical formulation.

The experiment shows that the normalized cut method is capable of addressing the problem raised in the first paragraph.

Currently, the World Wide Web contains billions of documents and is still growing rapidly. This poses a serious problem to the normalized cut, or the spectral methods in general. The spectral method has a notorious reputation of having long running time. This problem can be partially tackled in our algorithm. First, the number of web documents involved is only a very small subset of the entire web. We merely retrieve relevant documents and their neighbors. By neighbors, we mean those documents linking to or linked by them. This is a sound choice. Second, the link graph formed is sparse, which means the computation of the eigenvector can be fast. Third, with the improvement of hardware and software packages, the running time can be reduced significantly in the foreseeable future. Two problems with this method remain. One is how to evaluate the clustering results. The evaluation scheme under the web circumstance seems hard to design, and the judgement of the clustering quality is more subjective; this problem exists for most web document clustering methods. The second is how to modify the clustering results incrementally, that is, omitting the need to re-compute the entire data set again. These two problems are the main focus of future research.

There are many web communities that have very low connection or none at all, which form a disconnected, or nearly disconnected, web graph. This phenomenon is hard to detect by traditional algorithms in graph theory, but the spectral method can be applied successfully [20].

We have seen the success application of the normalized cut method to the web document clustering, but the new clustering criterion, the *min-max cut* criterion, proves

to be a better method, both experimentally and theoretically. When the overlap between two true clusters are relatively large, the min-max cut method often gives better results. This conclusion is supported in §3. The min-max cut tends to produce balanced clusters and is less prone to skewed cut, which is preferable in many applications, such as the load balance in multiprocessor environment.

The Fiedler vector provides a good linear order for searching for an optimal cut point, but it may not be the best one. The Linkage Differential order proposed in §3 seeks to find a better linear search order by reducing the min-max cut objective function. Searching for the new cut point based on the Linkage Differential order consistently outperforms those based on the order provided by the Fiedler vector. More important, this refinement procedure can start with any good initial partition.

Now, the normalized cut and min-max cut methods are applied successfully to cases in which the matrix formed is symmetric. Intuitively, we wonder if these two methods can be extended to the application area of an asymmetric case. The results in §4 provide a positive answer. The bipartite graph model has a wide variety of applications. Examples include the terms and documents analysis, as well as customers and purchased items in market basket analysis. The application of the normalized cut and min-max cut methods to the bipartite graph partition naturally leads to an SVD problem for the underlying asymmetric weight matrix, automatically associating two different types of clusters.

Correspondence analysis (CA) is a well-known method in multivariate statistics used to analyze contingency tables formed by two types of data objects. The bipartite graph partitioning methods introduced in §4 use the similar data objects. In §5, we reveal

that the spectral bi-clustering method using the min-max cut has an intrinsic relation to CA. CA is the direct result of clustering row and column data objects simultaneously using the bi-clustering method based on the min-max cut. CA captures the essential correlation between two types of data objects. Based on this result, we claim that CA is more advantageous than the popular Principle Component Analysis (PCA).

The spectral methods provide a promising result on data clustering; however, there still remains interesting and challenging problems. One is the K-way partition using multiple eigenvectors (or singular vectors in the asymmetric case) to get the multiple clusters at the same time. The preliminary results in this direction are presented in [39, 81]. Another problem is how to deal with missing data in the spectral method. The missing data is common in many commercial data sets. Finally, current spectral methods lack the capability of dealing with cases in which the data objects belong to more than one cluster. These problems deserve further investigation.

Appendix A

The HITS algorithm

Here we briefly introduce Kleinberg's HITS algorithm to find the authorities and hubs of each cluster we obtained. Kleinberg [56] defines the *authorities* as the most relevant documents for the topic. The *Hubs* are defined as the web documents which link to many related authorities. They implicitly represent an "endorsement" of the authorities they point to. The authority and hub information can be retrieved based entirely on the link structure information. Since a good *authority* is pointed to by many good *hubs* and a good *hub* points to many good authorities, such mutually rein forcing relationship can be represented as:

$$x_p = \sum_{q:(q,p)\in E} y_q \tag{A.1}$$

$$y_p = \sum_{q:(p,q)\in E} x_q \tag{A.2}$$

where x_p is the authority weight of web document x and y_p is the hub weight. E is the set of links(edges). Iteratively update the authority and hub weights of every web document, using Eqns.(A.1) and (A.2), and sort the web documents in decreasing order according to their authority and hub weights, respectively, we can obtain the authorities

and hubs of the topic. Many other issues need to be taken into consideration, such as the hyperlinks from the same web site, etc.

Let A be the adjacency matrix of the link graph, $\mathbf{x}$ be the vector of authority weights and $\mathbf{y}$ be the vector of hub weights, then (A.1) and (A.2) can be transformed to

$$\mathbf{x} = A^T\mathbf{y}, \qquad \mathbf{y} = A\mathbf{x}$$

which leads to

$$\mathbf{x} = A^T A\mathbf{x}, \qquad \mathbf{y} = AA^T\mathbf{y}.$$

Clearly, $\mathbf{x}$ and $\mathbf{y}$ are the principal eigenvectors of $A^T A$ and AA^T, respectively.

Appendix B

Mcut produces balanced cut

Theorem: Let $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_n$ be the eigenvalues of $\widehat{W}$ and assume that $\hat{\lambda}_1 + \hat{\lambda}_2 + \cdots + \hat{\lambda}_k > 0$. Then

$$\min_{Y_k^T Y_k = I_k, \mathbf{y}_i^T \widehat{W} \mathbf{y}_i > 0} \sum_{i=1}^{k} \frac{1}{\mathbf{y}_i^T \widehat{W} \mathbf{y}_i} = \frac{k^2}{\sum_{i=1}^{k} \hat{\lambda}_i}. \tag{B.1}$$

In addition, this minimum is achieved by any orthonormal basis $\{\mathbf{y}_1, \cdots, \mathbf{y}_k\}$ of the subspace spanned by the eigenvectors pertaining to the largest k eigenvalues of $\widehat{W}$ which further satisfies

$$\mathbf{y}_i^T \widehat{W} \mathbf{y}_i = \frac{\sum_{i=1}^{k} \hat{\lambda}_i}{k}.$$

proof. Let $\widetilde{Y}$ be the solution to the minimization problem (B.1). Then there exists an orthonormal matrix $\widetilde{Y}^c$ such that the matrix $(\widetilde{Y}\ \widetilde{Y}^c)$ is orthogonal. Define

$$\widetilde{W} = (\widetilde{Y}\ \widetilde{Y}^c)^T \widehat{W} (\widetilde{Y}\ \widetilde{Y}^c) = \begin{pmatrix} \widetilde{W}_{11} & \widetilde{W}_{12} \\ \widetilde{W}_{21} & \widetilde{W}_{22} \end{pmatrix}, \quad \text{and} \quad \mathcal{L}(Y_k) = \sum_{i=1}^{k} \frac{1}{\mathbf{y}_i^T \widetilde{W} \mathbf{y}_i}, \tag{B.2}$$

with $Y_k = (\mathbf{y}_1, \cdots, \mathbf{y}_k)$ satisfying $Y_k^T Y_k = I_k$. It is clear from the way $\widetilde{W}$ is defined that

$$Y^* = \begin{pmatrix} I_k \\ 0 \end{pmatrix} = (\mathbf{e}_1, \cdots, \mathbf{e}_k)$$

consists the optimal solution vectors that solve the minimization problem on the left hand side of (B.1) where the matrix $\widehat{W}$ is replaced by $\widetilde{W}$. For convenience, we assume that we have re-arranged all the ratios so that the sequence $\{\mathbf{e}_i^T \widetilde{W} \mathbf{e}_i\}$ is in decreasing order.

Let $\delta Y^{(2)} \in \mathcal{R}^{(n-k)\times k}$ be any sufficiently small perturbation. It is easy to show that there exists a perturbation

$$\delta Y^{(1)} = O\left(\left\|\delta Y^{(2)}\right\|_2^2\right) \in \mathcal{R}^{k\times k} \tag{B.3}$$

such that the perturbation

$$\delta Y = \begin{pmatrix} \delta Y^{(1)} \\ \delta Y^{(2)} \end{pmatrix} = (\delta \mathbf{y}_1, \cdots, \delta \mathbf{y}_k)$$

exactly satisfies the equation

$$\left(Y^* + \delta Y\right)^T \left(Y^* + \delta Y\right) = I_k. \tag{B.4}$$

In other words, $Y^* + \delta Y$ exactly satisfies the orthogonality constraint and hence is a feasible solution. Some algebra reveals that:

$$\mathcal{L}(Y^* + \delta Y) - \mathcal{L}(Y^*) = -2\sum_{i=1}^{k} \frac{\delta \mathbf{y}_i^T \widetilde{W} \mathbf{e}_i}{\left(\mathbf{e}_i \widetilde{W} \mathbf{e}_i\right)^2} + O(\|\delta Y\|^2)$$

$$= -2 \text{ trace} \left(\delta Y^T \widetilde{W} \left(\text{diag}(\mathbf{e}_1^T \widetilde{W} \mathbf{e}_1, \cdots, \mathbf{e}_k^T \widetilde{W} \mathbf{e}_k) \right)^{-2} \right) \quad \text{(B.5)}$$

Due to (B.3), linear perturbation terms in (B.5) that involve $\delta Y^{(1)}$ are the second order perturbation terms in $\delta Y^{(2)}$. For $\mathcal{L}(Y^*)$ to be a local minimum, all the linear terms involving $\delta Y^{(2)}$ on the right hand side of (B.5) must vanish. This can happen if and only if partition (B.2) satisfies $\widetilde{W}_{12} = 0$ and $\widetilde{W}_{21} = 0$.

It follows that the eigenvalues of $\widetilde{W}_{11}$ must be those of $\widetilde{W}$, and hence of $\widehat{W}$. Let these eigenvalues be $\widetilde{\lambda}_1, \cdots, \widetilde{\lambda}_k$, and let the diagonal entries of $\widetilde{W}_{11}$ be $\widetilde{w}_1, \cdots, \widetilde{w}_k$. It is well known that

$$\mathcal{L}\left(Y^*\right) = \sum_{i=1}^{k} \frac{1}{\mathbf{e}_i^T \widetilde{W} \mathbf{e}_i} = \sum_{i=1}^{k} \frac{1}{\widetilde{w}_i} \geq \frac{k^2}{\sum_{i=1}^{k} \widetilde{w}_i} = \frac{k^2}{\text{trace}\left(\widetilde{W}_{11}\right)} = \frac{k^2}{\sum_{i=1}^{k} \widetilde{\lambda}_i},$$

where the equality holds if and only if

$$\widetilde{w}_j = \frac{\sum_{i=1}^{k} \widetilde{\lambda}_i}{k} \quad \text{for all } j = 1, \cdots, k.$$

Combining these results, we now have

$$\min_{Y_k^T Y_k = I_k, \mathbf{y}_i^T \widehat{W} \mathbf{y}_i > 0} \sum_{i=1}^{k} \frac{1}{\mathbf{y}_i^T \widehat{W} \mathbf{y}_i} \geq \mathcal{L}(Y^*) \geq \frac{k^2}{\sum_{i=1}^{k} \widetilde{\lambda}_i} \geq \frac{k^2}{\sum_{i=1}^{k} \widehat{\lambda}_i}.$$

Furthermore, there exists an orthogonal matrix $\widetilde{Q}$ such that (see [82])

$$\widetilde{Q}^T \widehat{W} \widetilde{Q} = \begin{pmatrix} \widetilde{W}_1 & \mathbf{0} \\ \mathbf{0} & \widetilde{W}_2 \end{pmatrix},$$

where $\widetilde{W}_1 \in \mathcal{R}^{k \times k}$ is such that every one of its main diagonal entries is equal to $\left(\sum_{i=1}^{k} \widehat{\lambda}_i\right)/k$. Taking the first k columns of $\widetilde{Q}$ as the y_i vectors, we obtain

$$\min_{Y_k^T Y_k = I_k, \mathbf{y}_i^T \widehat{W} \mathbf{y}_i > 0} \sum_{i=1}^{k} \frac{1}{\mathbf{y}_i^T \widehat{W} \mathbf{y}_i} \leq \frac{k^2}{\sum_{i=1}^{k} \widehat{\lambda}_i}.$$

Since this upper bound is exactly equal to the earlier lower bound, the theorem must be true.

Appendix C

Proof of three results used in §2 and §5

In this appendix we prove three results:

C.(1). All the eigenvalues of $D^{-1/2}WD^{-1/2}$ has absolute value at most 1. Equivalently, we need to prove that the eigenvalues of the generalized eigenvalue problem (1.7) has absolute value at most 1. In fact let $\mathbf{x} = (x_i)_{i=1}^n$ and let i be such that $|x_i| = \max|x_j|$, then it follows from

$$\lambda d_i x_i = \sum_{j=1}^n w_{ij} x_j$$

that

$$|\lambda| \leq \sum_{j=1}^n w_{ij}/d_i = 1.$$

C.(2). We prove that

$$\sigma_{\max}(D_X^{-1/2} W D_Y^{-1/2}) = \max_{\mathbf{x}\neq 0, \mathbf{y}\neq 0} \frac{2\mathbf{x}^T W \mathbf{y}}{\mathbf{x}^T D_X \mathbf{x} + \mathbf{y}^T D_Y \mathbf{y}}.$$

Let $\hat{\mathbf{x}} = D_X^{1/2}\mathbf{x}$ and $\hat{\mathbf{y}} = D_Y^{1/2}\mathbf{y}$, then

$$\frac{2\mathbf{x}^T W\mathbf{y}}{\mathbf{x}^T D_X\mathbf{x} + \mathbf{y}^T D_Y\mathbf{y}} = \frac{2\hat{\mathbf{x}}^T D_X^{-1/2} W D_Y^{-1/2}\hat{\mathbf{y}}}{\hat{\mathbf{x}}^T\hat{\mathbf{x}} + \hat{\mathbf{y}}^T\hat{\mathbf{y}}}. \tag{C.1}$$

Let $D_X^{-1/2} W D_Y^{-1/2} = U\Sigma V^T$ be its SVD with

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_m], \quad V = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$$

and

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\min\{m,n\}}), \quad \sigma_1 = \sigma_{\max}(D_X^{-1/2} W D_Y^{-1/2}).$$

Then we can expand $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ as

$$\hat{\mathbf{x}} = \sum_i \hat{x}_i u_i, \quad \hat{\mathbf{y}} = \sum_i \hat{y}_i v_i, \tag{C.2}$$

and (C.1) becomes

$$\frac{2\sum_i \sigma_i \hat{x}_i \hat{y}_i}{\sum_i \hat{x}_i^2 + \sum_i \hat{y}_i^2} \le \frac{2\sigma_1 \sqrt{\sum_i \hat{x}_i^2}\sqrt{\sum_i \hat{y}_i^2}}{\sum_i \hat{x}_i^2 + \sum_i \hat{y}_i^2} \le \sigma_1.$$

Taking $\hat{x}_1 = 1$ and $\hat{y}_1 = 1$ achieves the maximum.

C.(3). Now we consider the constraint

$$\mathbf{x}^T D_X \mathbf{e} + \mathbf{y}^T D_Y \mathbf{e} = 0$$

which is equivalent to $\hat{x}_1 + \hat{y}_1 = 0$ using the expansions in (C.2). We can always scale the vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ without changing the maximum so that $\hat{x}_1 \geq 0$ and $\hat{y}_1 \geq 0$. Hence $\hat{x}_1 + \hat{y}_1 = 0$ implies that $\hat{x}_1 = \hat{y}_1 = 0$. It is then easy to see that

$$\sigma_2 = \max \left\{ \frac{2\mathbf{x}^T W \mathbf{y}}{\mathbf{x}^T D_X \mathbf{x} + \mathbf{y}^T D_Y \mathbf{y}} \mid \mathbf{x}^T D_X \mathbf{e} + \mathbf{y}^T D_Y \mathbf{e} = 0 \right\},$$

and the maximum is achieved by the second largest left and right singular vectors of $D_X^{-1/2} W D_Y^{-1/2}$.

References

[1] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. In *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 658–667, 1998.

[2] P. G. Anick. Adapting a full-text information retrieval system to compute the troubleshooting domain. In *Proc. of the 17th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 349–358, 1994.

[3] R. K. Belew and C. J. V. Rijsbergen. *Finding out about: A Cognitive Perspective on Search Engine Technology and the WWW.* Cambridge University Press, New York, NY, 2000.

[4] J. P. Benzecri. *Correspondence Analysis Handbook.* Marcel Dekker, New York, NY, 1992.

[5] M. W. Berry, B. Hendrickson, and P. Raghavan. Sparse matrix reordering schemes for browsing hypertext. *Lectures in Applied Mathematics*, 32:99–123, 1996.

[6] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proc. of the 7th World Wide Web Conference(WWW7)*, pages 379–388, 1998.

[7] D. Boley. Principal direction divisive partitioning. *Journal of Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[8] B. Bollobas. *Random Graphs.* Academic Press, New York, NY, 1985.

[9] S. Chakrabarti, B. E. Dom, D. Gibson, J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Mining the link structure of the world wide web. *IEEE Computer*, 32(8):60–67, 1999.

[10] S. Chakrabarti, B. E. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems*, 30(1-7):65–74, 1998.

[11] P. K. Chan, D. F. Schlag, and J. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, Sept. 1994.

[12] J. Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Problems in Analysis*, pages 195–199. Princeton University Press, 1970.

[13] C. Chekuri, A. Goldberg, D. Karger, M. Levin, and C. Stein. Experimental study of minimum cut algorithms. In *Proc. of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 324–333, 1997.

[14] C. K. Cheng and Y. C. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE. Trans. on Computed Aided Design*, 10:1502–1511, 1991.

[15] F. R. K. Chung. *Spectral Graph Theory.* American Mathematical Society, Providence, RI, 1997.

[16] W. B. Croft, R. Cook, and D. Wilder. Providing government information on the Internet: Experience with 'THOMAS'. In *Proc. of the 2nd Annual Conference on the Theory and Practice of Digital Libraries(DL) '95*, pages 19–24, 1995.

[17] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of American Society for Information Science*, 41:391–407, 1990.

[18] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. CS Tech Report #TR 2001-05, University of Texas Austin, 2001.

[19] C. Ding. A similarity-based probability model for latent semantic indexing. In *Proc. of the 22nd ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 59–65, Aug. 1999.

[20] C. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD-2001)*, pages 275–280, 2001.

[21] S. Doddi, M. V. Marathe, S. S. Ravi, D. S. Taylor, and P. Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. *Nordic Journal of Computing*, 7(3):185, 2000.

[22] W. E. Donath and A. J. Hoffman. Algorithms for partitioning of graphs and computer logic based on connected matrices. *IBM Tech. Disclosure Bulletin*, 15:938–944, 1972.

[23] W. E. Donath and A. J. Hoffman. Lower bounds for partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425, 1973.

[24] R. V. Driessche and D. Roose. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Computing*, 21:29–48, 1995.

[25] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 291–299, 1999.

[26] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, Inc., New York, NY, 1973.

[27] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* John Wiley and Sons, Inc., New York, NY, 2nd edition, 2000.

[28] E. N. Efthimiadis. A user-centered evaluation of ranking algorithms for interactive query expansion. In *Proc. of the 16th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 146–159, 1993.

[29] B. Everitt. *Cluster Analysis.* Halsted Press, New York, NY, 3rd edition, 1993.

[30] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *Proc. of the 19th IEEE Design Automation Conference*, pages 175–181, 1982.

[31] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math Journal*, 23:298–305, 1973.

[32] M. Fiedler. A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. *Czechoslovak Math Journal*, 25:619–633, 1975.

[33] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Proc. of the 6th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (SIGKDD-2000)*, pages 150–160, 2000.

[34] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo methods for finding low-rank approximations. *http://www.cs.yale.edu/homes/kannan/Papers/pubs.html*, 2000.

[35] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proc. of the 9th ACM Conference on Hypertext and Hypermedia (HYPER-98)*, pages 225–234, New York, June 1998. ACM Press.

[36] G. Golub and C. V. Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 2nd edition, 1989.

[37] A. D. Gordon. *Classification.* Chapman and Hall, Boca Raton, FL, 2nd edition, 1999.

[38] M. J. Greenacre. *Correspondence Analysis in Practice.* Academic Press, London, 1993.

[39] M. Gu, H. Zha, C. Ding, X. He, and H. D. Simon. Spectral relaxation models and structure analysis for k-way graph clustering and bi-clustering. Technical Report CSE-01-007, Department of Computer Science and Engineering, the Pennsylvania State University, 2001.

[40] S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal of Matrix Analysis and Applications*, 19(3):701–719, 1998.

[41] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE. Trans. on Computed Aided Design*, 11:1074–1085, 1992.

[42] M. A. Hearst and J. O. Paderson. Re-examining the cluster hypothesis: Scatter/gather on retrieval results. In *Proc. of the 19th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 246–255, 1996.

[43] B. Hendrickson and T. G. Kolda. Partitioning sparse rectangular and structurally nonsymmetric matrices for parallel computation. *SIAM Journal on Scientific Computing*, 21:2048–2072, 2000.

[44] B. Hendrickson and R. Leland. Chaco mesh partitioning software. *http://www.cs.sandia.gov/CRF/chac.html.*

[45] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for massively parallel computations. Technical Report 92-1460, Sandia National Labs, Albuquerque NM, 1992.

[46] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469, 1995.

[47] M. O. Hill. Correspondence analysis: A neglected multivariate method. *Applied Statistics*, 23:340–354, 1974.

[48] D. C. Hodgson and P. K. Jimack. Efficient mesh partitioning for parallel PDE solvers on distributed memory machines. Technical Report No.93-1, School of Computer Studies, University of Leeds, 1993.

[49] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. of the 22nd ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 50–57, 1999.

[50] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : A review. *ACM Computing Surveys*, 31(3):264–323, Sept. 1999.

[51] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, NY, 1986.

[52] G. Karypis and V. Kumar. Metis* a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *http://www.cs.umn.edu/~karypis.*

[53] G. Karypis and V. Kumar. Metis graph partitioning software. *http://www-users.cs.umn.edu/~karypis/metis/.*

[54] D. G. Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific Journal of Mathematics*, 28:565–570, 1969.

[55] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical J.*, 49:291–307, 1970.

[56] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[57] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The web as a graph: measurements, models, and methods. In *Proc. of the 5th Annual International Conference on Combinatorics and Computing*, pages 26–28. Springer-Verlag, 1999.

[58] J. M. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems: A micro-economic view of data mining. *Journal of Data Mining and Knowledge Discovery*, 2(4):311–324, 1998.

[59] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *Proc. of the 25th Very Large Data Base(VLDB) Conference*, pages 639–650, 1999.

[60] R. R. Larson. Bibliometrics of the world wide web: an exploratory analysis of the intellectual structures of cyberspace. In *Proc. of the 19th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 71–78, 1996.

[61] C. A. Leete, B. W. Peyton, and R. F. Sincovec. Toward a parallel recursive spectral bisection mapping tool. In *Proc. of the 6th SIAM Conference on Parallel Processing*, pages 923–928, 1993.

[62] Y. Li. Towards a qualitative search engine. *IEEE Internet Computing*, pages 2–7, July-August 1998.

[63] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. *http://www.cs.cmu.edu/~mccallum/bow*, 1996.

[64] G. L. Miller, S. H. Teng, W. Thurston, and S. A. Vavasis. *Automatic Mesh Partitioning*, volume 56. Springer Verlag, New York, NY, 1993.

[65] B. Mohar. Laplace eigenvalues of graphs - a survey. *Discrete Mathematics*, 109:171–183, 1992.

[66] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the web. *Proc. of ACM SIGCHI Conference on Human Factors in Computing Systems (SIGCHI'96)*, pages 118–125, 1996.

[67] M. F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[68] A. Pothen, H. D. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvectors of graph. *SIAM Journal of Matrix Analysis and Applications*, 11:430–452, July 1990.

[69] G. Ross. Classification techniques for large sets of data. In A. J. Cole, editor, *Numerical Taxonomy*. Academic Press, Inc., New York, NY, 1968.

[70] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.

[71] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–737, June 1997.

[72] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proc. of the 23rd ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR-2000)*, pages 208–215, 2000.

[73] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24:265–269, 1973.

[74] D. A. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. *IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996.

[75] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-plus.* Springer-verlag, New York, NY, 1999.

[76] C. J. von RijsBergen. *Information Retrieval.* Butterworths, London, 2nd edition, 1979.

[77] L. Wang. *Spectral Nested Dissection.* PhD thesis, The Pennsylvania State University, 1994.

[78] Y. C. Wei and C. K. Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *Proc. IEEE International Conference on Computer-Aided Design,* pages 298–301, 1989.

[79] P. Willett. Recent trends in hierarchical document clustering. *Information Processing and Management,* 24:577–597, 1988.

[80] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *Computer Networks,* 31(11–16):1361–1374, 1999.

[81] H. Zha, C. Ding, M. Gu., X. He, and H. D. Simon. Spectral relaxation for k-means clustering. *Accepted for Neural Information Processing Systems, NIPS*2001,* 2001.

[82] H. Zha and Z. Zhang. A note on constructing a symmetric matrix with specified diagonal entries and eigenvalues. *BIT,* 35:446–452, 1995.

Ernest Orlando Lawrence Berkeley National Laboratory
One Cyclotron Road | Berkeley, California 94720

Prepared for the U.S. Department of Energy under Contract No. DE-AC03-76SF00098