FINAL REPORT

DAVID BENSON-PUTNINS, MARGARET BONFARDIN, MEAGAN MAGNONI, DANIEL MARTIN

Advisors: Carl D. Meyer and Charles D. Wessell

1. Introduction

This summer, we investigated the fields of data mining and cluster analysis. Clustering is the act of assigning a set of objects into subsets, or clusters, so that objects in the same cluster are similar. We began our research into this area by learning about several known clustering techniques such as k-means and hierarchical clustering. Using two sets of documents, the REU paragraphs and the Blackstone documents, we created a search engine using latent semantic indexing and clustered the documents using non-negative matrix factorization. We also used two spectral techniques, the Fiedler Method and the MinMaxCut Method, to cluster these documents as well as Fisher's Iris data set and a data set of gene expression levels for leukemia patients.

Finally, we discovered results on the Fiedler Method and briefly summarize the paper we wrote about our research on Fisher's Iris data set.

2. Clustering Algorithms

At the beginning of the program, we learned about several different clustering techniques. We discuss some of these techniques below.

2.1. k-means.

The k-means clustering algorithm partitions n objects into k clusters. First, k initial centroid locations must be chosen, either randomly or by some other method. Second, the distance from each object to each centroid is measured. Each object is clustered with the nearest centroid, and the mean of each cluster becomes the new centroid location. The previous two steps are repeated until the clustering remains the same.

This clustering algorithm runs quickly and is easy to understand. However, one must specify the number of clusters, k, and the algorithm must be re-run if more data is added. The largest disadvantage of k-means is that it does not give a unique answer since the final clustering depends on the choice of initial centroid locations.

2.2. Hierarchical Clustering.

In hierarchical clustering, data is not partitioned into clusters in a single step. This agglomerative method works through a series of steps by fusing n objects into successively larger groups. Hierarchical clustering can be represented by a 2 dimensional diagram known as a dendrogram; this diagram illustrates the fusions made at each successive stage of analysis.

The advantages of hierarchical clustering are that k does not need to be specified from the begginning, outliers are easy to identify, and the answer is unique. On the other hand, the algorithm may take a long time to run. Also, hierarchical clustering is a greedy algorithm, meaning that it makes the locally optimal choice at each step.

2.3. **Principal Direction Divisive Partitioning.** Principal Direction Divisive Partitioning (PDDP) partitions n objects based on the principal partition. One performs a singular value decomposition of the matrix $\mathbf{A}$, where each column represents a data point. The principal partition of the data in matrix $\mathbf{A}$ is determined by the signs of the principal right-hand singular vector v_1. Positive signs in v_1 are placed in one cluster, and the columns corresponding to negative signs are placed into another cluster. A column associated with a zero entry in v_1 is arbitrarily assigned to either cluster. One repeats this procedure on the cluster of maximum variance until the desired number of clusters is obtained. There are several extensions of this method, including gap partitioning, which looks for gaps in the data, and secondary partitions, which considers both the directions of principal and secondary trend.

Once the singular value decomposition has been computed, it is easy to determine how to cluster the objects. However, the method is iterative, so if any questionable partitions are made, the mistake may be magnified through further iterations. In addition, arbitrarily assigning a zero entry in v_1 to a cluster makes this algorithm non-unique.

2.4. **Consensus Clustering.** Several of the clustering algorithms mentioned are non-unique. With these algorithms, it can be difficult to know which clustering is the best. Consensus clustering combines multiple clusterings of a data set to find a single (consensus) clustering that is a better fit than the existing clusterings. In order to perform consensus clustering, we build a consensus matrix where the (i, j) entry reflects the proportion of clusterings in which object i was clustered with object j. This consensus matrix can be interpreted as an undirected, weighted graph which can be partitioned into k clusters through spectral clustering techniques.

Although consensus clustering may find a single clustering that is better than existing clusterings, any weaknesses of the clustering technique used to cluster the consensus matrix becomes a weakness of the consensus clustering.

2.5. **Spectral Clustering.** Spectral clustering methods use the eigen properties of a matrix to form clusters. There are many spectral techniques, but we use the Fiedler Method and the MinMaxCut Method. These methods use the Laplacian matrix $\mathbf{L}$ of a graph, defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix of the graph and $\mathbf{D}$ is the diagonal matrix containing the rows sums of $\mathbf{A}$.

The Fiedler Method and the MinMaxCut method only work for undirected graphs. Some of the eigenvectors of $\mathbf{L}$ may not be continuous with respect to small changes in its entries, so small changes in weights of edges may lead to significantly different clusterings.

3. Latent Semantic Indexing

After surveying different clustering methods, we turned our focus to clustering and searching text documents. Latent semantic indexing is used in information retrieval to search text documents and is able to identify words with similar context instead of only performing direct key-word searches. We used the paragraphs that each REU participant wrote about themselves before the program began.

3.1. **Singular Value Decomposition.** To use latent semantic indexing, we first investigated the mathematical theory behind the method. Our group studied singular value decomposition and the relationship between eigenvalues and singular values. Singular value decomposition is a factorization of a matrix that can be used to reduce the dimensions of the matrix and eliminate "noise" while keeping important information. In general, a matrix $\mathbf{A}$ can be factored as

$$\mathbf{A_{m\times n}} = \mathbf{U_{n\times n}S_{n\times p}V^T_{p\times p}}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices. Next, $\mathbf{A}$ can be truncated using the first k singular values. This truncation reduces the dimensions of the data so that searches are not only for key-word matches.

3.2. **Term Document Matrix.** To perform latent semantic indexing, we first used a text to matrix generator developed by the University of Patras. This tool creates a dictionary of the documents' terms which have semantic meaning by removing words such as "the" and "a." Stemming maps related words to a common stem. For example, "happy" and "happily" both map to the stem "happ." The program constructs a term document matrix which has the terms as rows and the documents as columns. The entries in this matrix are the frequency each term occurs in each document.

$$\begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_m \end{array} \begin{array}{c} \begin{array}{cccc} D_1 & D_2 & \cdots & D_n \end{array} \\ \begin{pmatrix} freq_{11} & freq_{12} & \cdots & freq_{1n} \\ freq_{21} & freq_{22} & \cdots & freq_{2n} \\ \vdots & \vdots & & \vdots \\ freq_{m1} & freq_{m2} & \cdots & freq_{mn} \end{pmatrix} \end{array}$$

3.3. **Queries.** After truncating the singular value decomposition of the term document matrix, we search the documents. A query is submitted as a pseudo-document. To compare the query to each document we used cosine metric:

$$\cos\theta = \frac{\mathbf{q}^T\mathbf{D}_i}{||\mathbf{q}||\ ||\mathbf{D}_i||}$$

This metric measures the similarity of two vectors, $\mathbf{q}$ and $\mathbf{D}_i$, where $\mathbf{q}$ is the query and each $\mathbf{D}_i$ is a document in the collection.

These techniques allowed us to search for paragraphs relating to one-word and multiple-word queries.

3.4. **Clustering with LSI.** After searching through the documents, we explored using latent semantic indexing to cluster documents. We randomly selected a document and clustered this document with others which were at least as similar as a specified cutoff value. We repeated this process until every document was in a cluster. However, this technique created many singleton clusters, and we decided it was ineffective.

4. Text Clustering

After our failed attempt to cluster text documents using latent semantic indexing, we compared two clustering methods, Non-negative Matrix Factorization (NMF) and the Min-MaxCut Method. We also developed a tool for visualizing clusters, the heat map, which is explained in detail in section 6 of our attached paper.

4.1. **Blackstone Documents.** We realized that we would need a data set more suited for text clustering. So we scrapped the REU paragraphs and used a set of documents assembled by Ralph Abbey during his 2007 REU. The 86 documents (nicknamed "the Blackstone documents") consist of the first few paragraphs gleaned from Wikipedia pages or news articles. The documents break down into the following categories:

1.) Blackstone IPO: 1-10
2.) Blackstone buyout: 11-20
3.) UNC people (athletics): 21-30
4.) Chess grandmasters: 31-34
5.) Chess openings: 35-40
6.) NMF: 41-47
7.) Chess strategy: 48-51
8.) Poker: 52-56
9.) Computers and chess: 57-59, 75-78, 84
10.) Global Warming: 60-64
11.) Backgammon: 65-67
12.) Computer programming games: 68
13.) Computers and backgammon: 69-71, 85-86
14.) Computers and poker: 72-74, 79-83

Using the Blackstone documents, we could easily verify the veracity of our clusterings. Also, the contextual overlap of several categories allowed for different clustering options (for instance, there is no definitive k).

4.2. **Non-negative matrix factorization.** After procuring a suitable data set, we examined two clustering algorithms, including non-negative matrix factorization. NMF starts with a non-negative matrix $\mathbf{A}$, in this case the term document matrix (see section 3.2), and finds non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ such that $\mathbf{A} \approx \mathbf{WH}$. To make k clusters, we want

$$\mathbf{A}_{m\times n} \approx \mathbf{W}_{m\times k}\mathbf{H}_{k\times n}$$

We randomly initialize $\mathbf{W}$ and $\mathbf{H}$, then iteratively update the two matrices:

$$\begin{aligned} \mathbf{H} &= \mathbf{H}.*((\mathbf{W}^T\mathbf{A})./(\mathbf{W}^T\mathbf{WH}+\epsilon)) \\ \mathbf{W} &= \mathbf{W}.*((\mathbf{AH}^T)./(\mathbf{WHH}^T+\epsilon)) \end{aligned}$$

We continue until reaching a maximum iteration count or until $||\mathbf{A}-\mathbf{WH}||$ is less than some tolerance. For example, suppose we want to cluster the 86 documents into 8 clusters. By choosing k=8, NMF attempts to describe each document in terms of 8 *topics*, represented by the columns of $\mathbf{W}$. The columns of $\mathbf{H}$ contain the coordinates for each document in the *topic* space.

$$\mathbf{A} \approx \mathbf{W}_{2020\times 8}\mathbf{H}_{8\times 86} =$$

$$\begin{array}{l} \\ \text{Blackstone} \\ \text{Chess} \\ \vdots \\ \text{Computers} \\ \vdots \\ \text{Stock} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Topic1} & \text{Topic2} & \cdots & \text{Topic8} \end{array} \\ \begin{pmatrix} 1.10 & 1.40 & \cdots & 0.05 \\ 0.12 & 0.04 & \cdots & 2.01 \\ \vdots & \vdots & \ddots & \vdots \\ 0.30 & 0.40 & \cdots & 1.60 \\ \vdots & \vdots & \ddots & \vdots \\ 0.25 & 2.05 & \cdots & 0.10 \end{pmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{cccc} \text{“RecordIPO”} & \text{“StockSales”} & \cdots & \text{“Chess”} \end{array} \\ \begin{pmatrix} 1.30 & 1.02 & \cdots & 0.30 \\ 0.60 & 1.45 & \cdots & 0.05 \\ \vdots & \vdots & \ddots & \vdots \\ 0.05 & 0.01 & \cdots & 1.30 \end{pmatrix} \end{array}$$

For example, the document $RecordIPO = (1.30\times topic1)+(0.60\times topic2)+\cdots+(0.05\times topic8)$. Now we need to use $\mathbf{W}$ and $\mathbf{H}$ to cluster the documents. We assign a document to a cluster based on the largest coordinate in its column of $\mathbf{H}$:

$$\begin{array}{c} \begin{array}{cccc} \text{“RecordIPO”} & \text{“StockSales”} & \cdots & \text{“Chess”} \end{array} \\ \begin{pmatrix} \boxed{1.30} & 1.02 & \cdots & 0.30 \\ 0.60 & \boxed{1.45} & \cdots & 0.05 \\ \vdots & \vdots & \ddots & \vdots \\ 0.05 & 0.01 & \cdots & \boxed{1.30} \end{pmatrix} \end{array}$$

Since NMF naturally forms a topic pseudo-document for each cluster, we labeled each cluster with the top five words in its topic vector. Thus, we can identify clusters at a glance without reading through the documents contained in each cluster. Below is a sample picture (with color-coded clusters), in which we clicked on the cluster of documents describing NMF.

4.3. **MinMaxCut.** We also clustered the Blackstone documents using a spectral method, MinMaxCut. We will quickly outline this method; it is described in detail in Section 4 of our paper. The MinMaxCut algorithm requires three steps:

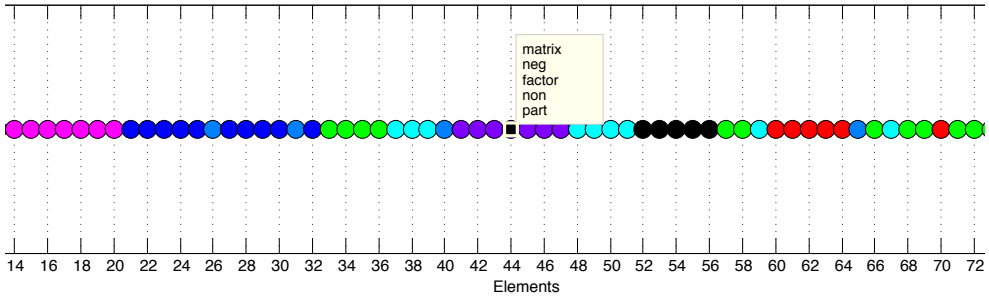

FIGURE 1. Cluster about NMF clicked to display its top 5 words.

(1) Cluster the term document matrix many times with kmeans
(2) Combine results into a consensus matrix to make weighted graph
(3) Partition the graph by minimizing the MinMaxCut objective function

$$min \sum_{i=1}^{k} \frac{w(A_i, \overline{A_i})}{w(A_i, A_i)}$$

where A_j is the j^{th} cluster

After clustering the Blackstone documents, we displayed the results using our heat map. This heat map reorganizes the consensus matrix by cluster and allows us to observe the intra- and inter-connectivity of the clusters. We summed each cluster's constituent document vectors and displayed the top five words of a cluster. In the example below, we clicked on the cluster consisting of documents about NMF.

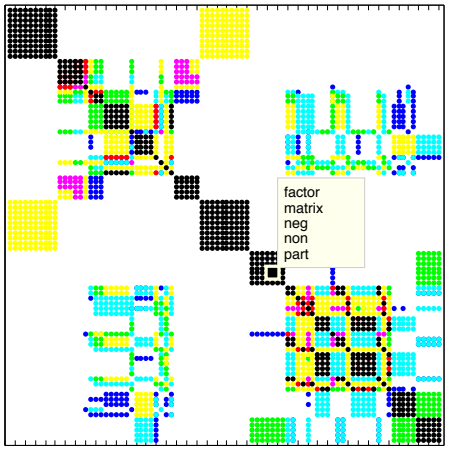

FIGURE 2. Cluster containing NMF documents clicked to display its top 5 words in MinMaxCut heat map.

4.4. **Conclusion.** Clustering documents can prove very useful. After the Enron scandal many of the management's emails were confiscated. The emails were then clustered so that investigators would only need to read relevant documents (and not emails about fantasy football, incidentally the largest cluster created). Our methods and tools cluster documents well and allow for quick textual categorization. NMF creates meta-topics to cluster the data and naturally supplies us with topic words. MinMaxCut creates a weighted graph to cluster the data and allows us to visualize cluster connectivity.

5. Cluster and Data Analysis

The majority of our time this summer was spent researching and writing a paper, *Cluster and Data Analysis*, which is attached.

Abstract

Clustering is the act of partitioning a set of elements into subsets, or clusters, so that elements in the same cluster are, in some sense, similar. Determining an appropriate number of clusters in a particular data set is an important issue in data mining and cluster analysis. Another important issue is visualizing the strength, or connectivity, of clusters. We begin by creating a consensus matrix using multiple runs of the clustering algorithm k-means. This consensus matrix can be interpreted as a graph, which we cluster using a spectral clustering method, the MinMaxCut Method. To determine if increasing the number of clusters from k to k +1 is appropriate, we check whether an existing cluster can be split. We also present our method for visualizing the strength of clusters by using the consensus matrix and the clustering obtained through one of the aforementioned spectral clustering techniques. Using these methods, we then investigate Fisher's Iris data set. Our methods support the existence of four clusters, instead of the generally accepted three clusters in this data.

6. Results on Fiedler

After working extensively with the Fiedler method in our paper, we investigated two possible issues with the Fiedler vector.

6.1. **Fiedler Clustering.** Given a weighted graph, G, number its vertices $1, 2, .., n$. We start by constructing the adjacency matrix $\mathbf{A}$. $\mathbf{A}$ has non-negative entries and is symmetric, and the (i, j) entry is the weight of the edge. To construct the Laplacian of a graph, construct $\mathbf{D}$ the diagonal matrix with $\mathbf{D}_{i,i}$ the row sum of the ith row of $\mathbf{A}$ - equivalently $\mathbf{D}_{i,i}$ is the degree of vertex i. Then the Laplacian is simply $\mathbf{L} = \mathbf{D} - \mathbf{A}$. We observe that since $\mathbf{D}$ and $\mathbf{A}$ are symmetric, $\mathbf{L}$ is also.

First we list some well known properties of the Laplacian whose proofs can be found in the literature:

Theorem 1. *The Laplacian is positive semi-definite*

Theorem 2. *The dimension of $ker(\boldsymbol{L})$ is equal to the number of connected components of G*

In fact, if G has connected components B_1,...,B_r then a basis of the kernel of $\mathbf{L}$ is the indicator vectors of these components, $\mathbf{1}_{B_1}, ..., \mathbf{1}_{B_r}$ where $\mathbf{1}_{B_i}$ has a 1 in entry j if vertex j lies in component B_i and a 0 otherwise.

For the rest of this section, we assume G is connected and contains more than one vertex. When doing Fiedler clustering, the standard method is to compute the Fiedler vector, v_2, and then separate G into the vertices whose corresponding entry in v_2 is positive and those whose entry is negative - the jth entry in v_2 corresponding to vertex j in G. Unfortunately, this does not deal with the situation of when a zero may occur in v_2. Furthermore, the question of whether both positive and negative entries occur in the Fiedler vector is an important one, since it is critical to actually clustering. We now follow up with some results that are not commonly discussed in the literature. We refer to the "'positive cluster" and "negative cluster" as the set of vertices whose corresponding entry in the Fiedler vector are positive and negative respectively.

Theorem 3. *There must be at least one positive and one negative entry in the Fiedler vector*

We prove this by contradiction. Suppose the Fiedler vector contains only zeros and positive entries. Pick vertex j corresponding with the smallest entry in the Fiedler vector; if there is a tie pick the vertex corresponding to the first occurrence. Now consider the jth entry of $\mathbf{L}v_2$ which is equal to λv_2 for some value of $\lambda > 0$ by the simplicity of the zero eigenvalue and positive semi-definiteness of $\mathbf{L}$

This means in particular the jth entry of $\mathbf{L}v_2$ must be non-negative since the jth entry of v_2 is. Looking at the matrix multiplication this means

$$\sum_{k=1}^{n} l_{j,k}(v_2)_k \geq 0$$

where $l_{j,k}$ is the entry of $\mathbf{L}$ in the jth row and kth column, and $(v_2)_k$ is the kth entry of v_2. We note $(v_2)_j \leq (v_2)_k$ for all values of k and $l_{j,j} = -\sum_{k\neq j} l_{j,k}$ so $(v_2)_j l_{j,j} = -\sum_{k\neq j}(v_2)_j l_{j,k} \leq \sum_{k\neq j}(v_2)_k l_{j,k}$ with equality only if all the entries of v_2 are equal. As discussed above that is the zero eigenvector of $\mathbf{L}$ so v_2 does not have all equal entries, meaning

$(v_2)_j l_{j,j} < \sum_{k\neq j}(v_2)_k l_{j,k}$ which implies $\sum_{k=1}^{n} l_{j,k}(v_2)_k < 0$

This is a contradiction with the statement that it must be non-negative, so it must be that the Fiedler vector cannot have only non-negative entries

The non-positive case is handled in a similar manner, or by simply noting that if v_2 has non-positive entries, then $-v_2$ is also a choice of Fiedler vector and this has only non-negative entries

Theorem 4. *There can be zeros in the Fiedler vector*

Obviously if there were no examples of graphs with zeros in the Fiedler vector, the discussion of what properties those corresponding vertices must have would be moot.

The simplest example is the following graph. The Fiedler vector for this graph is $\begin{pmatrix} 2 \\ 0 \\ -1 \\ -1 \end{pmatrix}$.

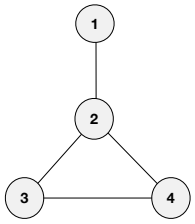

FIGURE 3. Graph with zeros in the Fiedler vector

We observe that Fiedler clustering is most often justified as an attempt at finding a vector x which minimizes $x^t\mathbf{L}x$ subject to $x \perp e$ where e is the vector of all ones, and every entry of x is 1 or -1 (depending on which cluster it is placed in). In this case, there is no ambiguity about how the clustering of this graph should occur: vertex 1 should be in one cluster, and 2, 3 and 4 in the other one. This small example demonstrates that zeros not only can occur in the Fiedler vector, but when they do it is not necessarily because of ambiguity as to which cluster the vertex belongs to. For this reason further study of the zeros of Fiedler vectors is interesting.

In fact the situation can be worse than this. One can modify the above graph by taking vertex 2 and expanding it into a complete graph of, for example, 997 vertices. Vertices 1, 3 and 4 each have an edge weight of $\frac{1}{997}$ to each of the verticecs in the newly created complete graph. Then the Fiedler vector for this graph is

$$\begin{pmatrix} 2 \\ 0 \\ \vdots \\ 0 \\ -1 \\ -1 \end{pmatrix}$$

So graphs with arbitrarily high percentages of the Fiedler vector being zero are possible.

Theorem 5. *Every vertex with a corrseponding zero in the Fiedler vector is adjacent to both a vertex in the positive cluster and a vertex in the negative cluster, or only to vertices that correspond to zeros.*

The proof is by contradiction, and similar to the one for theorem 3.
Suppose entry j in v_2 is zero, and every edge from vertex j connects to a vertex with a corresponding positive entry or zero entry, and at least one entry is positive. Then $\mathbf{L}v_2$ is a scalar multiple of v_2 because the Fiedler vector is an eigenvector of $\mathbf{L}$, so the jth entry of $\mathbf{L}v_2$ must be zero. By matrix multiplication, this means

$$\sum_{k=1}^{n} l_{j,k}(v_2)_k = 0$$

The only non-zero values of $l_{j,k}$ will be when $k = j$ or when $(v_2)_k$ is positive, because $l_{j,k}$ is the negative of the weight of the edge from vertex j to vertex k, and there only exist edges from j to vertices in the positive cluster. Since $(v_2)_j$ is zero, the left hand side of the above equation is a summation of terms that are of the form negative number multiplied by positive number, so is a summation of negative numbers. Hence it cannot be zero as required, which is a contradiction.

The case where vertex j has edges only adjacent to vertices in the negative cluster is handled similarly, or by noting that by using $-v_2$ to cluster, j will have edges only to vertices in the positive cluster. So the vertices with zeros in the Fiedler vector essentially lie in the boundary between the positive and the negative cluster, as one would expect.

6.2. **Non-simple eigenvectors.** If the smallest non-zero eigenvalue of the Laplacian is not simple, Fiedler clustering breaks down, because one now has to choose the eigenvector from the eigenspace. We first note that in general one way that non-simple eigenvalues can occur for the Laplacian is when there is symmetry in the graph; if a non-trivial graph isomorphism from the graph to itself exists. Then if v is an eigenvector with eigenvalue λ, if the values of v are re-arranged according to the permutation of vertices from the isomorphism, since re-arranging the Laplacian according to the permutation does not change it (by definition a graph isomorphism does not change the edge connections) the re-arranged v will be an eigenvector with eigenvalue λ also. An example of a graph where this occurs is:

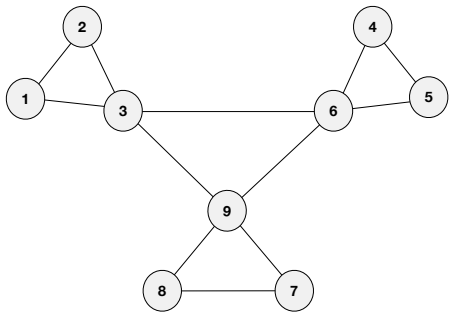

FIGURE 4. Graph with non-simple second eigenvalue

In fact the double eigenvalue occurs for the smallest non-zero eigenvalue.

Fiedler clustering is often justified as a way of minimizing $x^t\mathbf{L}x$ where x is a vector of 1's and -1's (the clustering is all the 1 vertices and all the -1 vertices form the two clusters). In this graph there is ambiguity as to which vertices are to be clustered together using this; clearly the clusters will be one of six vertices and one of three vertices, but which group of three should be separated is ambiguous. To demonstrate that this is not what is causing the non-simple eigenvalue another example is presented:

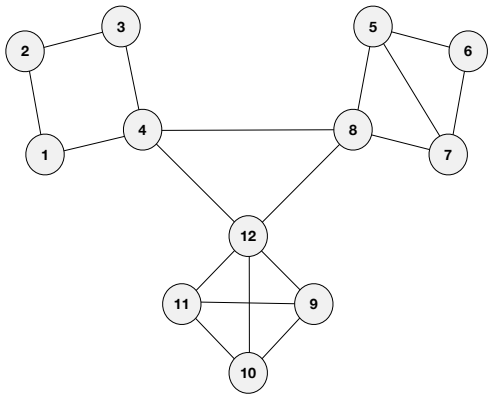

FIGURE 5. Symmetric graph w.r.t. clustering with simple eigenvalue

The objective function to miminize is independent of internal edge weights, so again there is ambiguity as to how the clustering should occur. But the Fiedler vector occurs for a simple eigenvalue; the complete K_4 is assigned zeros, and the cycle and the other graph are given positive and negative values respectively. So the ambiguity of the clustering does not manifest itself as a multiple eigenvalue as might be expected, but rather in the form of zeros in the Fiedler vector.

This suggests the hypothesis that non-simple eigenvalues occur only because of graph symmetries, and not because of clustering ambiguities as it might seem to be. At any rate, we propose here a mechanism for dealing with a non-simple second eigenvalue: change all the edge weights by a small random amount. The objective function that our goal is to minimize is continuous with respect to changes in the Laplacian, so the "correct" clustering is left unchanged. On the other hand, arbitrarily small changes in a matrix can change it from having non-simple eigenvalues to simple eigenvalues. Since the set of matrices with non-simple eigenvalues has measure zero, with probability one this will result in a Laplacian with simple eigenvalues.

Running this ten times on graph of nine vertices above, changing all the edge weights by a number uniformly picked in the interval $[-.5, .5]$, all ten times returned a choice

of correct clustering: six vertices were placed in one cluster, three in the other as would be expected. Compare this to simply picking out an eigenvector that has the smallest eigenvalue (.5505): while matlab's algorithm by luck gives a vector that gives a correct clustering, other possible vectors returned include, for example:

$$\begin{pmatrix} 2.9826 \\ 2.9826 \\ 1.3425 \\ -2.9826 \\ -2.9826 \\ -1.3406 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

And we have to deal with the situation of zeros in the Fiedler vector (which is not a resolved issue). In fact, we can go so far as to state:

Theorem 6. *If the eigenspace for the second smallest eigenvalue is non-simple, then there exists an eigenvector which contains a zero in the first entry*

The proof is simple: If v_2 and v_2' are two linearly independent vectors of the eigenspace, and their first entries are a and b respectively, there are two cases:
If a or b is zero, we're done. Otherwise, let $v = v_2 - \frac{a}{b}v_2'$. v is an element of the eigenspace as well, and the first entry is $a - \frac{ab}{b} = 0$ as required.

We finish with a final observation and point of further inquiry: In any vector space, the measure of the set of vectors that contains a zero is zero. Also, in the example above, all choices of Fiedler vector actually have the same ratio between the first three entries of the vector, and the fourth through sixth entries of the vector, and the final three entries of the vector. This means that vertices 1, 2 and 3 will all have the same sign for their corresponding entries, as will vertices 4, 5 and 6, as will vertices 7, 8 and 9. So any choice of vector in the eigenspace will give a correct clustering (not always the same clustering though, since the symmetry of the graph means there is no unique best clustering) as long as it contains no zeros.

A more complicated example is the following graph:

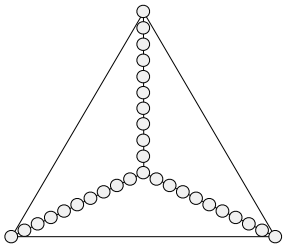

FIGURE 6. Graph with 3 dimensional second eigenspace

This has a three dimensional eigenspace for the second smallest eigenvalue. We compare the two possible techinques: pick a random eigenvector from the space and use it to run Fiedler clustering, and change all the edge weights a small amount and run Fiedler clustering on the new graph.

There are three common clusterings that come up using this technique (though others may be possible). Two of them give the minimal value of $x^t\mathbf{L}x$ of 12, and one gives a value of 16. The three clusterings are shown below, with one of the clusters colored purple

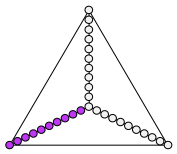

FIGURE 7. One optimal clustering

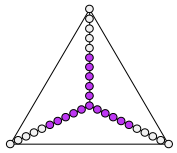

FIGURE 8. Second optimal clustering

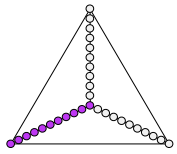

FIGURE 9. Non-optimal clustering

To pick a random eigenvector from the eigenspace, we picked a point uniformly on the unit sphere of the eigenspace. Doing this resulted in getting one of the optimal clusterings seventy seven percent of the time. We compared this to the strategy of changing every edge weight by a random quantity uniformly picked between -.05 and .05. This strategy resulted in an optimal clustering 84 percent of the time. Both strategies were run 10,000 times to get these percentages. The changing edge weight strategy is slightly more successful for this particular graph, and further research into it may be called for.

7. Summary

Over the ten weeks of the REU at North Carolina State University, our group:

- surveyed different clustering algorithms
- created a search engine using latent semantic indexing
- clustered documents with non-negative matrix factorization and MinMaxCut
- submitted a paper, *Cluster and Data Analysis*, to SIAM Undergraduate Research Online (SIURO) about our results for Fisher's Iris data set
  - created a tool for visualizing cluster connectivity
  - created a tool for determining the number of clusters
- researched aspects of the Fiedler method and proved results concerning Fiedler vector