Distributed Spectral Clustering and Outlier Detection, Fall 2010

Sandipan Dey,
CMSC 698
UMBC CSEE

January 23, 2011

Abstract

Spectral clustering in directed graphs can be formulated as an optimization problem as shown in [11, 12, 15], the objective being a weighted cut in the graph. There are several variants of the spectral clustering algorithm, but all of them start by computing the pairwise similarity matrix from the data and perform eigen decomposition of the Laplacian matrix. In this paper, a thorough survey is done on the existing spectral clustering methods and the centralized version is extended to a distributed version. Outliers can be found as bi-product of the clustering algorithm. An extensive set of experiments (including a set of challenging clustering problems) are conducted using Mathematica and the results are shown.

Keywords

Spectral Clustering, Outlier

Introduction

As in [11], this paper focuses on pairwise (similarity-based) clustering methods. In contrast to statistical clustering methods that generates the observed data points, pairwise clustering defines a similarity function. An increasingly popular approach to similarity based clustering is by spectral methods. These method use the eigenvalues and eigenvectors of the similarity matrix and produce impressive segmentation results. Here a distributed extension to this spectral partitioning problem is proposed in the context of outlier detection.

Literature

As explained in [15], the spectral clustering approach is most related to the graph theoretic formulation of grouping or spectral partitioning [1]. The set of points in an arbitrary feature space are represented as a weighted undirected graph $G = (V, E)$, where the nodes of the graph are the points in the feature space, and an edge is formed between every pair of nodes. The weight on each edge, s_{ij}, is a function of the similarity between nodes i and j.

Symbols	Definitions
x_i	The i^{th} Row (data tuple)
S	Similarity Matrix $S_{ij} = d(x_i, x_j)$
D	Diagonal Degree Matrix $D_{ii} = \sum_j s_{ij}$
L	Un-normalized Laplacian $L = D - S$
L_{sym}	Normalized Laplacian $L_{sym} = D^{-\frac{1}{2}}.L.D^{-\frac{1}{2}}$
L_{rw}	Normalized Laplacian $L_{rw} = D^{-1}.L = I - D^{-1}.S$
P	Row-stochastic Transition Matrix $P = D^{-1}.S$

As stated in [7], spectral clustering is the technique of partitioning the rows of a (similarity) matrix according to a top singular eigenvector of their components in the top few singular vectors. There are several variants of the spectral clustering algorithm, some of them are described in [17] as follows,

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

Normalized spectral clustering according to Ng, Jordan, and Weiss (2002)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the normalized Laplacian L_{sym}.
- Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{sym}.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- Form the matrix $T \in \mathbb{R}^{n \times k}$ from U by normalizing the rows to norm 1, that is set $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of T.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

Figure 1: Spectral Clustering Algorithms

Different similarity graphs

Start by computing the similarity (or the affinity) matrix S, where s_{ij} measures how "similar" the two objects x_i and x_j are [14, 13]. As explained in [17], similarity graph can be constructed in many different ways as follows:

1. ϵ-neighborhood graph: All points whose pairwise distances are smaller than ϵ, usually considered as an unweighted graph.

2. k-nearest neighbor graph: Connect vertex v_i with vertex v_j if v_j is among the k-nearest neighbors of v_i (and / or) vice-versa.

3. Fully connected graph: Simply connect all points with positive similarity with each other, and weight all edges by s_{ij}. An example for such a similarity function is the Gaussian similarity function $s(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$.

Different Cuts (Objective functions)

Next step is to obtain an optimal partitioning of the dataset (e.g., into $k = 2$ clusters), which is same as finding minimum cut w.r.t. the similarity matrix. As shown in [15], the Cut is needed to be normalized in order to rule out the isolated points to be chosen as min-cut sets. The following defines different cuts [17],

1. $RatioCut(A, \bar{A}) = \left(\frac{1}{|A|} + \frac{1}{|\bar{A}|}\right) Cut(A, \bar{A})$

2. $NCut(A, \bar{A}) = \left(\frac{1}{vol(A)} + \frac{1}{vol(\bar{A})}\right) Cut(A, \bar{A})$
   $= \left(\frac{1}{assoc(A,V)} + \frac{1}{assoc(\bar{A},V)}\right) Cut(A, \bar{A})$
   where $Cut(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} S_{ij}$ and
   $vol(A) = assoc(A, V)$ is the total connection from nodes in A to all nodes in the graph [15].

Computing the optimal partitioning becomes equivalent to minimizing the Rayleigh quotient [15] [4] [18],

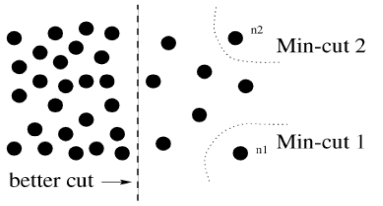
$$min_x NCut(x) = min_y \frac{y^T(D - S)y}{y^T Dy},$$
$$\text{w.r.t. } y(i) \in \{1, -b\} \text{ and } y^T D1 = 0$$

As shown in [15], this is a discrete optimization [5] and an NP hard problem (by reducing from PARTITION). If y is relaxed to take on real values, this is shown to be minimized by solving the generalized eigenvalue system,

$$(D - S).y = \lambda Dy$$

which follows from the following Linear Algebraic theorem for Rayleigh quotient [6]

Let A be a real symmetric matrix. Under the constraint that x is orthogonal to the $j - 1$ smallest eigenvectors $x_1, \ldots x_j$, the quotient $\frac{x^T Ax}{x^T x}$ is minimized by the next smallest eigenvector x_j and its minimum value is the corresponding eigenvalue λ_j. This can be converted to the standard eigenvalue



A case where minimum cut gives a bad partition[9]

problem [15] as:

$$D^{-\frac{1}{2}} S D^{-\frac{1}{2}}.x = \lambda x$$

which can be solved in $O(n^3)$, where n is the number of nodes in the graph.
The relative merits of different eignevector based clustering methods in [15].

Thus, the second smallest eigenvector of the generalized eigensystem [15] is the real valued solution to the normalized cut problem. Using similar argument it can be shown that the eigenvector with the third smallest eigenvalue is the real valued solution that optimally subpartitions the first two parts. In fact, this line of argument can be extended to show

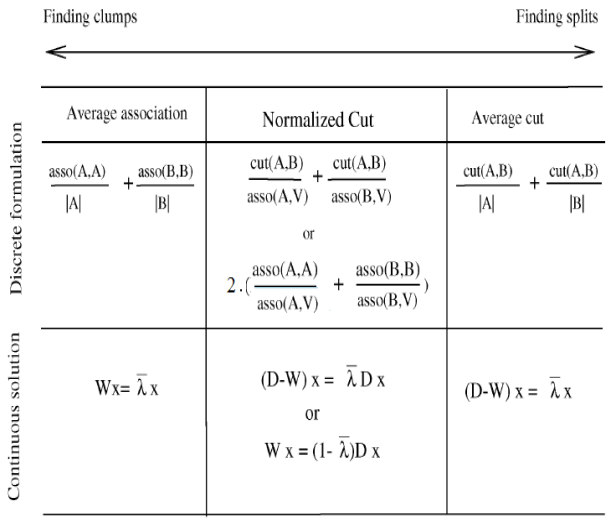


Figure 2: Relative Merits of spectral clustering techniques

that one can subdivide the existing graphs, each time using the eigenvector with the next smallest eigenvalue. However, in practice, because the approximation error from the real valued solution to the discrete valued solution accumulates with every eigenvector taken and all eigenvectors have to satisfy a global mutual orthogonality constraint, solutions based on higher eigenvectors become unreliable. It is best to restart solving the partitioning problem on each subgraph individually [15].

Performance of Spectral Clustering

Using matrix perturbation theory [6], the spectral clustering algorithms can be analyzed [17], as in [13] such a performance guarantee is computed under a set of reasonable assumptions. Also, as argued in [7], one variant of spectral clustering provides an effective worst-case guarantees, while another finds a "good" clustering if it exists. The quality of clustering are measured by two parameters α (conductance, minimum quality of the cluster) and ϵ (measures total edges that are not covered by the cluster) and the objective is to find an (α, ϵ) clustering that maximizes α and minimizes ϵ. The following theoretical performance guarantees are obtained from [7],

Figure 3: Performance Guarantees of Spectral Clustering

Random Walk

As explained in [11, 12] the NCut algorithm focuses on the second smallest eigenvalue and its corresponding eigenvector, λ_L and x_L respectively. The elements of x_L have approximately the same value within each cluster (piecewise constant). In [15] it is shown that when there is a partitioning of A, s.t.,

$$x_i^L = \begin{cases} \alpha & : i \in A \\ \beta & : i \in \bar{A} \end{cases}$$

then $(A, \bar{A})$ is the optimal NCut and the value of the cut itself is λ_L.
As explained, the NCut algorithm lacks the following intuitive explanation:

1. what causes x_L to be piecewise constant?

2. what happens when there are more than two segments?

3. how does the algorithm degrade its performance when x_L is not piecewise constant?

By "normalizing" the similarity matrix S one obtains the stochastic matrix

$$P = D^{-1}S$$

whose row sums are all 1.

If λ, x are solutions of $Px = \lambda x$ and $P = D^{-1}S$, then $(1-\lambda)$, x are solutions of $Lx = \lambda Dx$

Define $P_{AB} = Pr[A \to B|A]$ as the probability of the random walk transitioning from set $A \subset I$ to set $B \subset I$ in one step if the current state is in A and the random walk is started in its stationary distribution.

$$P_{AB} = \frac{\sum_{i\in A, j\in B} \pi_i^{\infty} P_{ij}}{\pi^{\infty}(A)} = \frac{\sum_{i\in A, j\in B} S_{ij}}{\mathrm{vol}(A)}$$

If G is connected and non bi-partite and a random walk $(X_t)_{t\in N}$ starting with X_0 in the stationary distribution Π. For disjoint subsets $A, B \in V$, denote by $P(B|A) = P(X_1 \in B|X_0 \in A)$.

Then as shown in [17], [11, 12, 10] [2],

$$Ncut(A, \bar{A}) = P(A|\bar{A}) + P(\bar{A}|A)$$

Issues

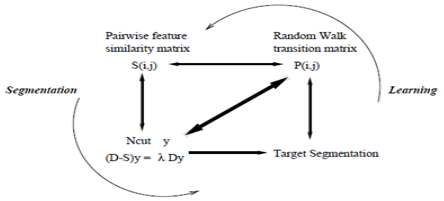
As discussed in [9], there are two great obstacles when applying these methods to large-scale text datasets:

1. these methods require finding eigenvectors of the similarity matrix, a very slow operation

2. the similarity matrix itself, for large text datasets, is dense and therefore prohibitively expensive in both storage space and algorithm runtime, not scalable.

The Grouping Algorithm

1. Given an image or image sequence, set up a weighted graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.
2. Solve $(\mathbf{D} - \mathbf{S})\boldsymbol{x} = \lambda \mathbf{D}\boldsymbol{x}$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
4. Decide if the current partition should be subdivided and recursively repartition the segmented parts if



If the NCut is small for a certain partition $A, \bar{A}$ then it means that the probabilities of evading set A, once the walk is in it and of evading its complement $\bar{A}$ are both small. Intuitively, we have partioned the set I into two parts such that the random walk, once in one of the parts, tends to remain in it.

The NCut is strongly related to a the concept of low conductivity sets in a Markov random walk. A *low conductivity set* A is a subset of I such that $h(A) = \max(P_{A\bar{A}}, P_{\bar{A}A})$ is small. They have been studied in spectral graph theory in connection with the *mixing time* of Markov random walks

Figure 4: Random Walk View of Spectral Clustering

The issues are addressed along different directions:

1. sample the data points and do computation on a much smaller matrix,

2. sparsify the matrix by a k-nearest neighbor technique and do computation on a much sparser matrix, and

3. do computation on lots of machines at the same time parallelly by sparsifying the dense similarity matrix and then by using Nystrm method [16].

4. considering the local convergence by modifying power iteration method (PI) to power iteration clustering (PIC) [9].

5. bipartite Graph and "Path Folding" [9]

The Distributed Spectral Clustering and Outlier Detection Problem

Consider the dataset $D_{m\times n}$ horizontally partitioned to N nodes, s.t., $D = \bigcup_{i=1}^{N} D_i$, with dataset $D_{m_i\times n}$ at node i, $\forall i \in 1 \dots N$, s.t., $\sum_{i=1}^{N} m_i = m$. We have to partition the dataset (globally) into two disjoint sets $D = O \cup (D - O)$, where O is the (global) outlier set. We can view this grouping problem as graph partitioning problem as described in [15], with $O \cap (D - O) = \Phi$.

The challenge is that we can't run outlier detection algorithm on the entire data at once, hence, have to analyze the data locally first and then combine the results to find global outliers. We shall use spectral clustering algorithm (recursively) to cluster the data into clusters and find outlying clusters as bi-product of the spectral clustering algorithm.
Let us propose couple of different ways of distributed extension of the algorithm and present the experimental results. Most of the existing approaches work best for separating image foreground from its background. The goal of this paper is little different, we instead want to find outliers.

Algorithm DISPEC

Assumption

Let us start with the very simple case $k = 2$ clusters (C_1 and C_2) and at every node both the clusters are present. Moreover, the data is homogeneous in the sense that at every node the $\left|(x \in C_1)_{N_i}\right| \geq \left|(x \in C_2)_{N_i}\right|$, $\forall i$.

Algorithm 1 DISPEC: Distributed Iterative Spectral Clustering

1: **for** i=1 to t **do**
2: (Map Phase) Read Input. Locally compute the Normalized Laplacian matrices and solve $D^{\frac{-1}{2}} S D^{\frac{-1}{2}} x = \lambda x$ to find the 2nd smallest eigenvector at each node.
3: Use this (piecewise constant) eigenvector as threshold to label the clusters.
4: (Reduce Phase) Combine the similar labels from all nodes.
5: Store data tuples corresponding to 2 different clusters separately and use them as input for next iteration.
6: **end for**

For step 2, if the eigenvector has only positive and negative eigenvalues and a sharp transition at 0, mark all the positive values in the eignevector, let E_+ hold the set of corresponding tuples while E_- contains the remaining tuples, i.e. corresponding to the negative values in the eigenvector.Now, by assumption, if $|E_+| > |E_-|$, mark all the tuples in E_+ as C_1 and E_2 as C_2, otherwise vice-versa).For step 3, label can be used as key.
The input t for the algorithm depends upon the user choice (how many clusters he wants to find). This is a top-down approach like [8] and it ends up with 2^t clusters. Outliers can be found as a biproduct from the resulting clusters.

Algorithm DSCSTE

Algorithm 2 DSCSTE: Distributed Spectral Clustering with Symmetric Tridiagonal Eigenproblem

1: Use Divide & Conquer Method for the symmetric Tridiagonal Eigenproblem [3]
2: Initially the entire data is at a centralized location, preprocess to find the global affinity matrix.
3: Use Householder transform to recursive tridiagonalize the matrix and send the blocks to each node (divide step: these tridiagonal matrices will have same eigenvalues as the original data matrices).
4: Do eigen-analysis on the local tri-diagonal matrices and combine to get the global eigenvectors (conquer step).
5: Use k-means to find the clusters from the global top k eigenvectors.

Experimental Results

We present extensive results on standard UCI Machine learning data (Iris) as well as several interesting images presenting challenging problems for spectral clustering, both for centralized and distributed version using DISPEC in this section. Also we compare the distributed version with its centralized counterpart.

Conclusion

Computing the similarity and Laplacian matrices along with computing the eigenvectors have been a serious scaling issue for spectral clustering specially if the number of data tuple are huge. This paper provides a way of locally computing the similarity matrices, computing the eigenvectors and eigenvalues under certain assumptions combining them to get the global clustering. The technique can be made much faster using Hadoop like parallel frameworks.

Figure 8: Output of Spectral Clustering (Distributed vs. Centralized)

References

[1] F. Chung. *Spectral Graph Theory*. Number 92. 1997.

[2] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, pages 239–246, 2007.

[3] J. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36(1):177–195, 1981.

[4] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.

[5] P. Drineas, A. M. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *SODA*, pages 291–299, 1999.

[6] G. Golub and C. V. Loan. *Matrix Computations*. 1989.

[7] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

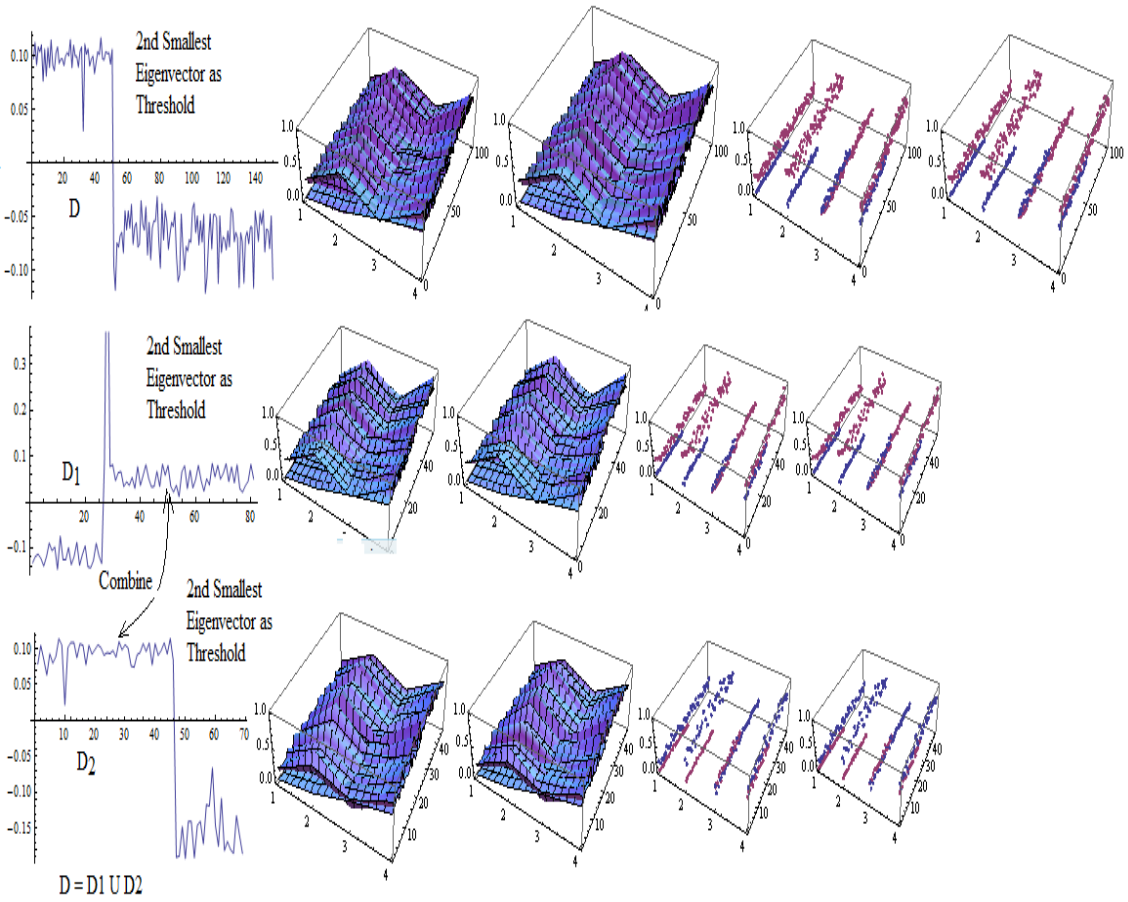


Figure 5: Output of Spectral Clustering on Iris Data (Centralized vs. Distributed)

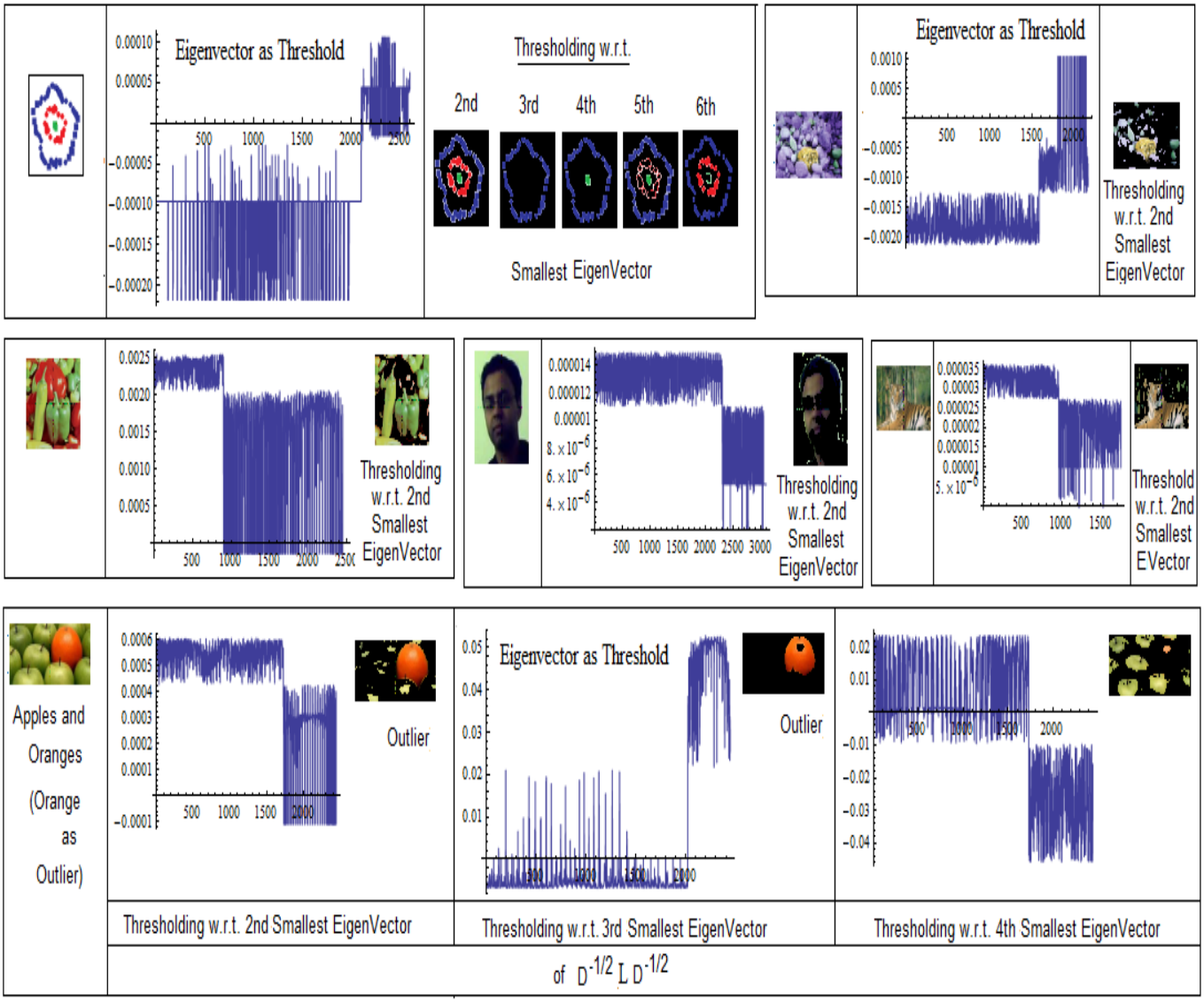


Figure 6: Output of Spectral Clustering (Centralized)

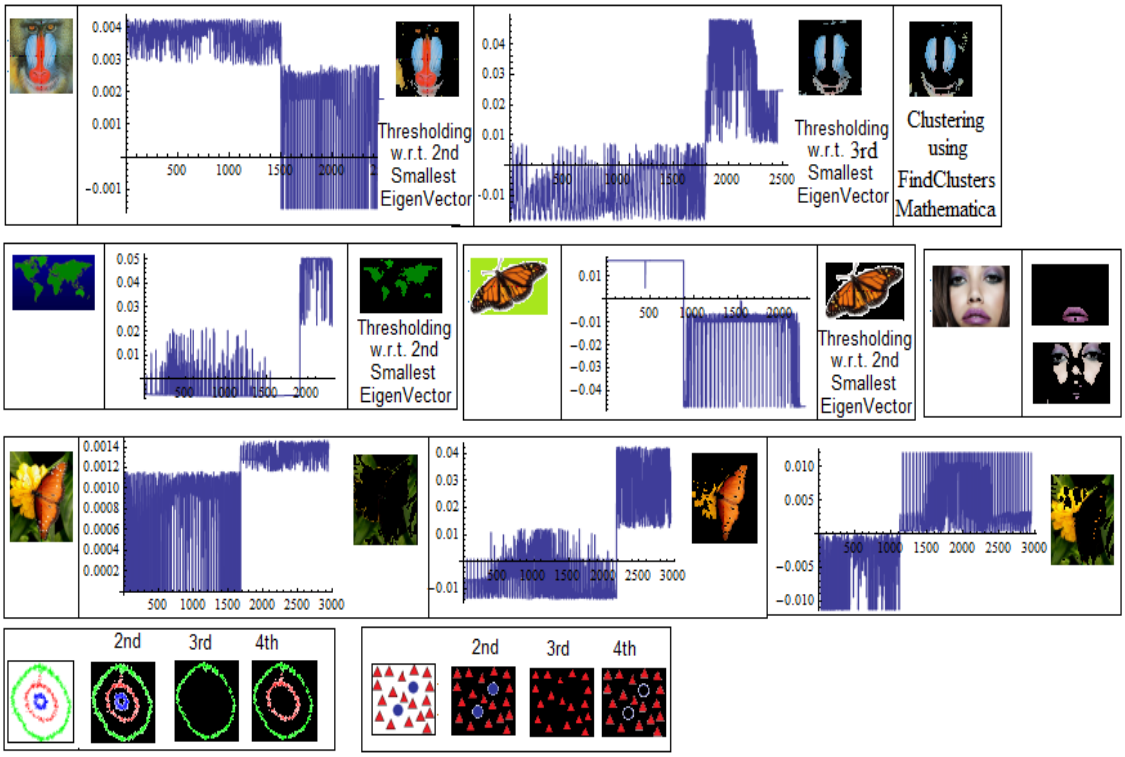
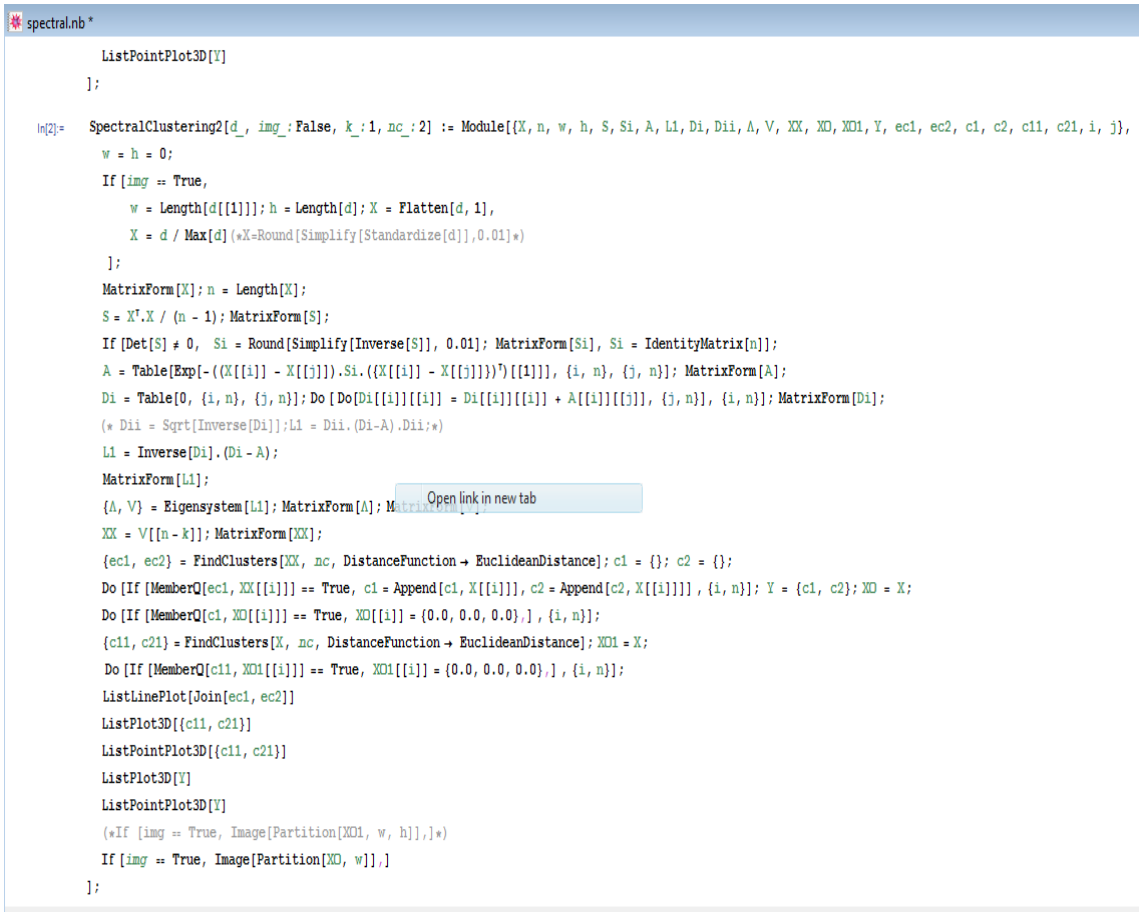


Figure 7: Output of Spectral Clustering (Centralized



spectral.nb *

```
    ListPointPlot3D[Y]
  ];

In[2]:= SpectralClustering2[d_, img_:False, k_:1, nc_:2] := Module[{X, n, w, h, S, Si, A, L1, Di, Dii, Λ, V, XX, XD, XD1, Y, ec1, ec2, c1, c2, c11, c21, i, j},
    w = h = 0;
    If [img == True,
        w = Length[d[[1]]]; h = Length[d]; X = Flatten[d, 1],
        X = d / Max[d] (*X=Round[Simplify[Standardize[d]],0.01]*)
     ];
    MatrixForm[X]; n = Length[X];
    S = X^T.X / (n - 1); MatrixForm[S];
    If [Det[S] ≠ 0,  Si = Round[Simplify[Inverse[S]], 0.01]; MatrixForm[Si], Si = IdentityMatrix[n]];
    A = Table[Exp[-((X[[i]] - X[[j]]).Si.({X[[i]] - X[[j]]})^T)[[1]]], {i, n}, {j, n}]; MatrixForm[A];
    Di = Table[0, {i, n}, {j, n}]; Do [Do[Di[[i]][[i]] = Di[[i]][[i]] + A[[i]][[j]], {j, n}], {i, n}]; MatrixForm[Di];
    (* Dii = Sqrt[Inverse[Di]];L1 = Dii.(Di-A).Dii;*)
    L1 = Inverse[Di].(Di - A);
    MatrixForm[L1];
    {Λ, V} = Eigensystem[L1]; MatrixForm[Λ]; M
    XX = V[[n - k]]; MatrixForm[XX];
    {ec1, ec2} = FindClusters[XX, nc, DistanceFunction → EuclideanDistance]; c1 = {}; c2 = {};
    Do [If [MemberQ[ec1, XX[[i]]] == True, c1 = Append[c1, X[[i]]], c2 = Append[c2, X[[i]]]] , {i, n}]; Y = {c1, c2}; XD = X;
    Do [If [MemberQ[c1, XD[[i]]] == True, XD[[i]] = {0.0, 0.0, 0.0},] , {i, n}];
    {c11, c21} = FindClusters[X, nc, DistanceFunction → EuclideanDistance]; XD1 = X;
    Do [If [MemberQ[c11, XD1[[i]]] == True, XD1[[i]] = {0.0, 0.0, 0.0},] , {i, n}];
    ListLinePlot[Join[ec1, ec2]]
    ListPlot3D[{c11, c21}]
    ListPointPlot3D[{c11, c21}]
    ListPlot3D[Y]
    ListPointPlot3D[Y]
    (*If [img == True, Image[Partition[XD1, w, h]],]*)
    If [img == True, Image[Partition[XD, w]],]
  ];
```

Open link in new tab

Figure 9: Mathematica Code for Simulation

[8] H. Kargupta, V. Puttagunta, M. Klein, and K. Sarkar. On-board vehicle data stream monitoring using minefleet and fast resource constrained monitoring of correlation matrices. *New Generation Comput.*, 25(1):5–32, 2006.

[9] F. Lin and W. Cohen. A very fast method for clustering big text datasets.

[10] M. Meila and W. Pentney. Clustering by weighted cuts in directed graphs. In *SDM*, 2007.

[11] M. Meila and J. Shi. Learning segmentation by random walks. In *NIPS*, pages 873–879, 2000.

[12] M. Meila and J. Shi. A random walks view of spectral segmentation. 2001.

[13] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[14] P. Perona and W. T. Freeman. A factorization approach to grouping. In *ECCV (1)*, pages 655–670, 1998.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[16] Y. Song, W. Chen, H. Bai, C.-J. Lin, and E. Y. Chang. Parallel spectral clustering. In *ECML/PKDD (2)*, pages 374–389, 2008.

[17] U. von Luxburg. A tutorial on spectral clustering. In *Statistics and Computing*, page 17 (4), 2007.

[18] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *ICCV*, pages 975–982, 1999.