Distributed Outlier Detection by Spectral Partitioning

Project Report - CMSC 698

12/20/2010

Sandipan Dey
UMBC CSEE

Problem Definition

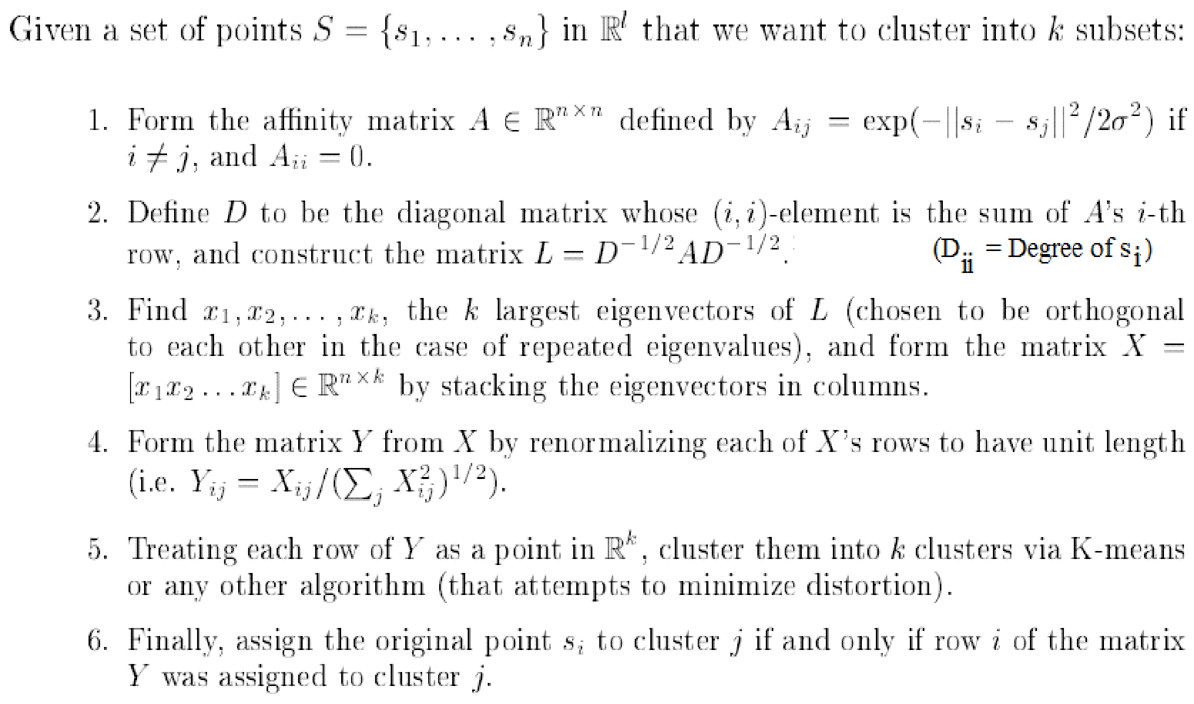
Consider the dataset D_{mxn} horizontally partitioned to N nodes, s.t., $D = \bigcup_{i=1}^{N} D_i$, with dataset $D_{m_i xn}$ at node i, $\forall i = 1, \ldots, N$, s.t., $\sum_{i=1}^{N} m_i = m$. We have to partition the dataset (globally) into two disjoint sets $D = O \cup (D - O)$, where O is the (global) outlier set. We can view this grouping problem as graph partitioning problem as described in [10], 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.

Existing literatures (Centralized)

Spectral clustering

The algorithm is described in [11] as follows:



Given a set of points $S = \{s_1, \ldots, s_n\}$ in $\mathbb{R}^l$ that we want to cluster into k subsets:

1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by $A_{ij} = \exp(-||s_i - s_j||^2/2\sigma^2)$ if $i \neq j$, and $A_{ii} = 0$.
2. Define D to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix $L = D^{-1/2}AD^{-1/2}$. (D_{ii} = Degree of s_i)
3. Find $x_1, x_2, \ldots, x_k$, the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [x_1 x_2 \ldots x_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors in columns.
4. Form the matrix Y from X by renormalizing each of X's rows to have unit length (i.e. $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{1/2}$).
5. Treating each row of Y as a point in $\mathbb{R}^k$, cluster them into k clusters via K-means or any other algorithm (that attempts to minimize distortion).
6. Finally, assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

As explained in [19],

Different similarity graphs **(different types)**

The ε-neighborhood graph: connect all points whose pairwise distances are smaller than ε

k-nearest neighbor graphs: connect vertex v_i with vertex v_j if v_j is among the k-nearest neighbors of v_i.

The fully connected graph: connect all points with positive similarity with each other, and weight all edges by similarity function e.g., Gaussian, $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$

As explained in [1], the following interpretations of the affinity matrix A (which is the same matrix W as defined in [2], [19]) are of interest:

- The matrix encodes the pairwise similarities of vertices of a graph.
- The rows of the matrix are points in a d-dimensional Euclidean space. The columns are the coordinates.
- The rows of the matrix are documents of a corpus. The columns are terms. The (i, j) entry encodes information about the occurrence of the jth term in the ith document.

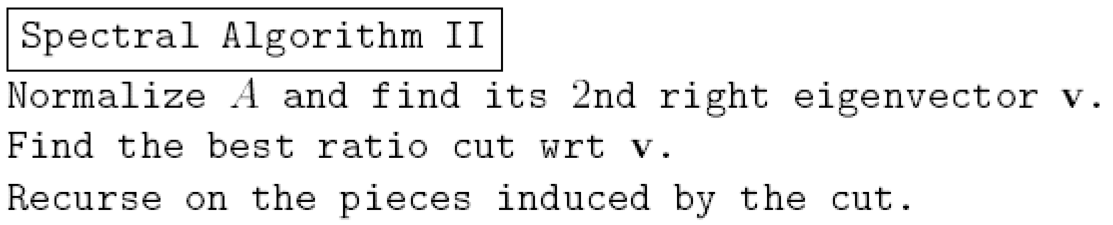
As pointed out in [3], spectral clustering is a pair-wise (similarity based) clustering algorithm. A more concise description of the algorithm and its intuition is explained in [1], as follows:

Given a matrix A, the spectral algorithm for clustering the rows of A is given below.

```
Spectral Algorithm I
Find the top $k$ right singular vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.
Let $C$ be the matrix whose $j$th column is given by $A\mathbf{u}_j$.
Place row $i$ in cluster $j$ if $C_{ij}$ is the largest entry in the $i$th row of $C$.
```

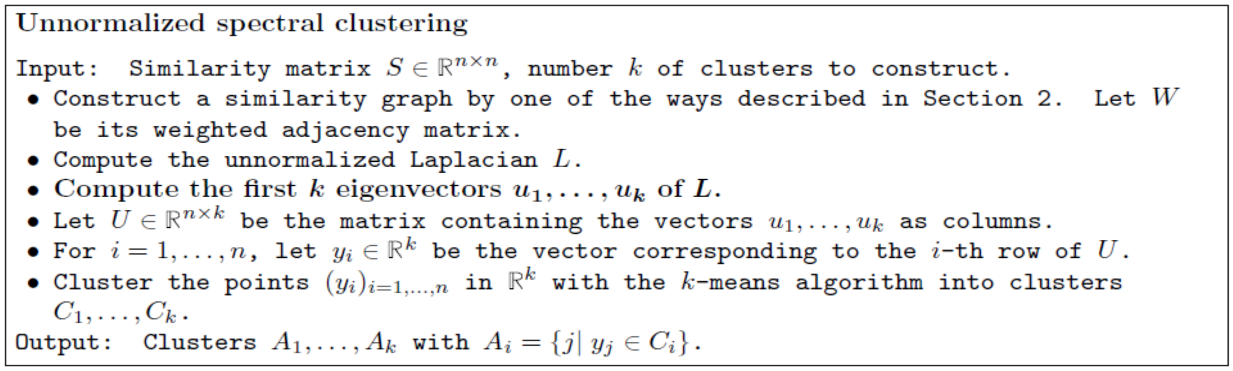
The algorithm has the following interpretation . Suppose the rows of A are points in a high-dimensional space. Then the subspace defined by the top k right singular vectors of A is the rank k subspace that best approximates A. The spectral algorithm projects all the points onto this subspace. Each singular vector then defines a cluster; to obtain a clustering we map each projected point to the (cluster defined by the) singular vector that is closest to it in angle.

Given an (α, ϵ)-partition, then an $(\frac{\alpha}{12\log^2 n}, 25\epsilon\log^2 n)$-partition will be found by the approximate-cut algorithm

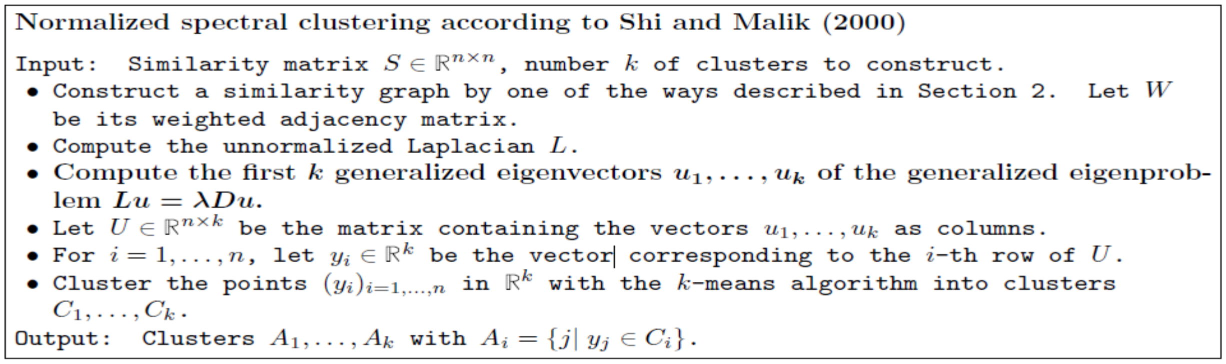


```
Spectral Algorithm II
Normalize $A$ and find its 2nd right eigenvector v.
Find the best ratio cut wrt v.
Recurse on the pieces induced by the cut.
```

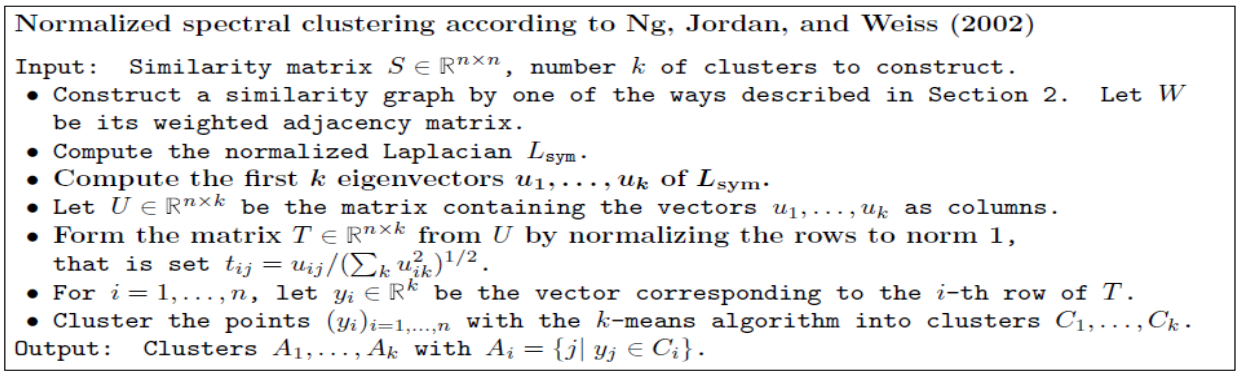
Also, as explained in [2], the Perona & Freeman algorithm [9] describes an O(n) approximation of the affinity matrix, thereby proposing a spectral clustering based on the first eigenvector of the affinity matrix. Also, different spectral algorithms are analyzed in generic a simple grouping setting in [2] that shows that these spectral clustering algorithms have inherent similarities. Some of the variants of spectral clustering algorithms as described in [19] are



```
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 $A_i = \{j | \, y_j \in C_i\}$.
```

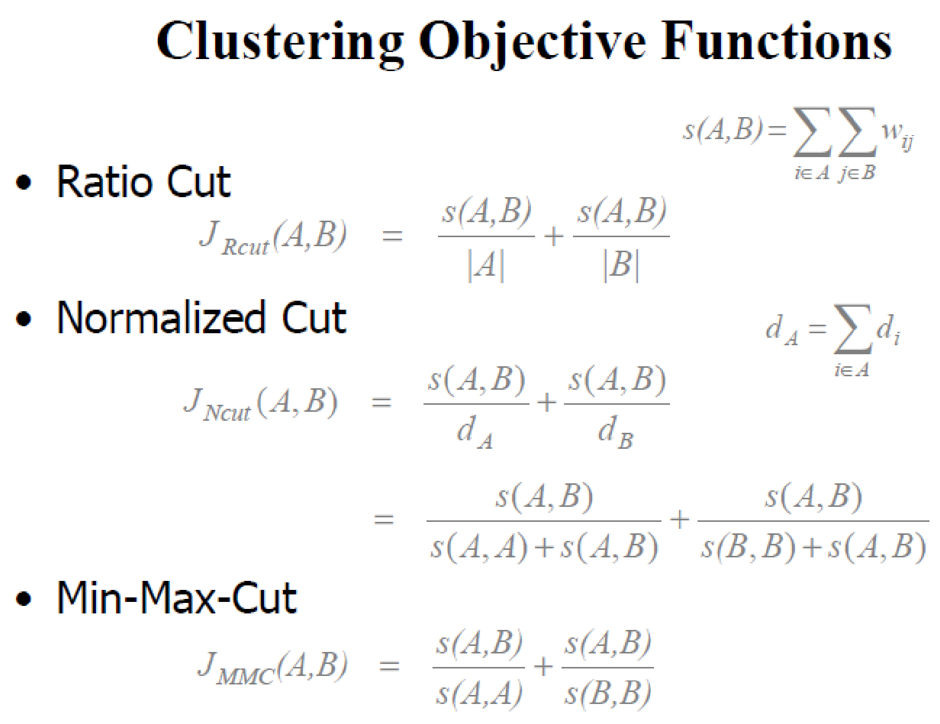


```
Normalized spectral clustering according to Shi and Malik (2000)
Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct.
  • Construct a similarity graph by one of the ways described in Section 2. Let $W$
    be its weighted adjacency matrix.
  • Compute the unnormalized Laplacian $L$.
  • Compute the first $k$ generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenprob-
    lem $Lu = \lambda Du$.
  • 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 $L_{\mathrm{sym}}$.
  • Compute the first $k$ eigenvectors $u_1, \ldots, u_k$ of $L_{\mathrm{sym}}$.
  • 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 $A_i = \{j | \, y_j \in C_i\}$.
```

Types of Cuts [18]



Clustering Objective Functions

$$s(A,B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$$

- Ratio Cut

$$J_{Rcut}(A,B) = \frac{s(A,B)}{|A|} + \frac{s(A,B)}{|B|}$$

- Normalized Cut

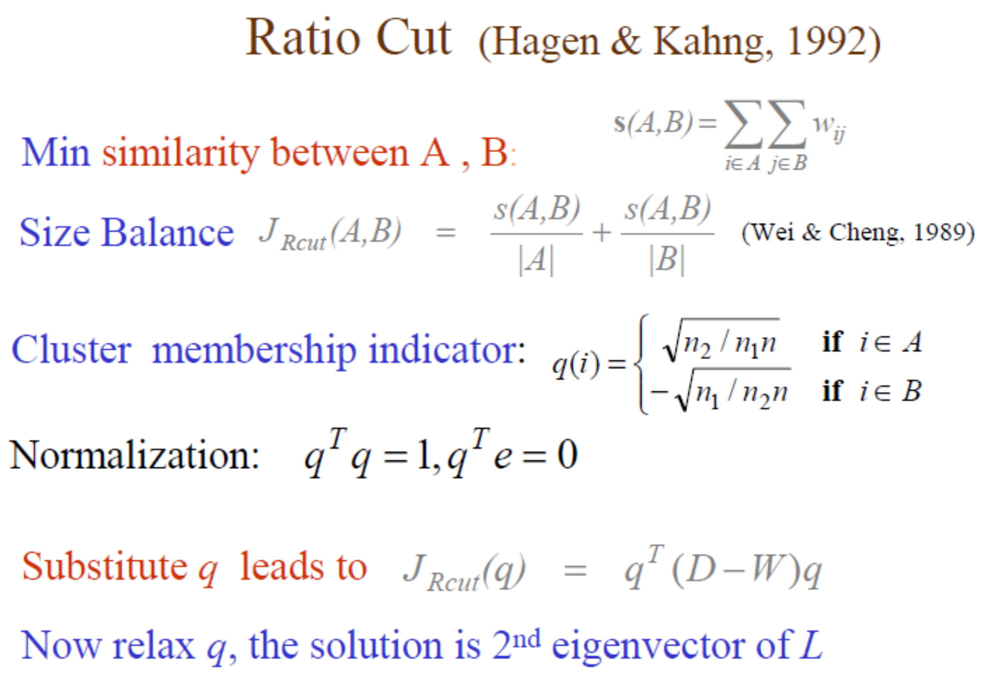
$$d_A = \sum_{i \in A} d_i$$

$$J_{Ncut}(A,B) = \frac{s(A,B)}{d_A} + \frac{s(A,B)}{d_B}$$

$$= \frac{s(A,B)}{s(A,A)+s(A,B)} + \frac{s(A,B)}{s(B,B)+s(A,B)}$$

- Min-Max-Cut

$$J_{MMC}(A,B) = \frac{s(A,B)}{s(A,A)} + \frac{s(A,B)}{s(B,B)}$$



Ratio Cut (Hagen & Kahng, 1992)

Min similarity between A , B: $s(A,B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$

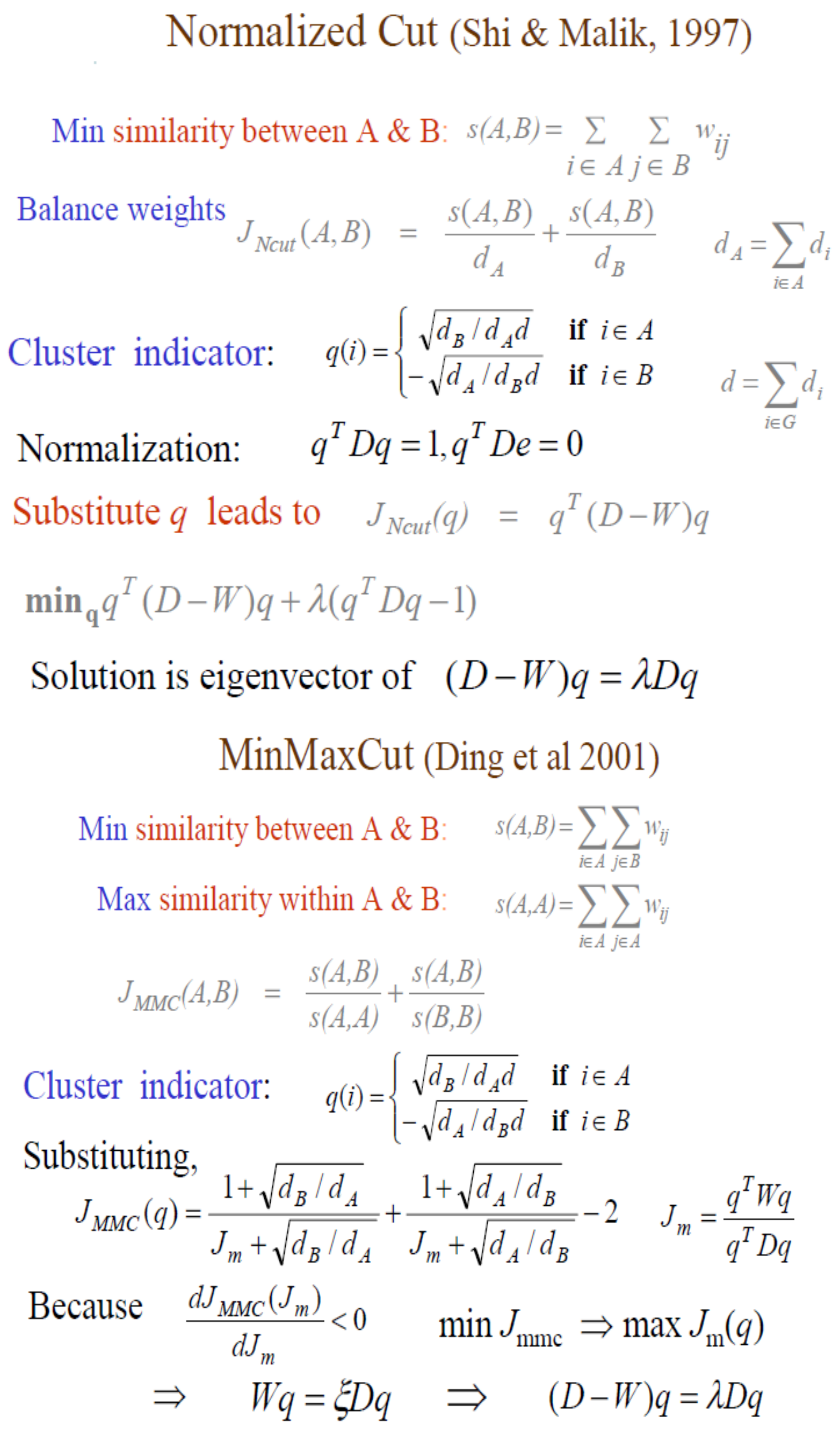
Size Balance $J_{Rcut}(A,B) = \frac{s(A,B)}{|A|} + \frac{s(A,B)}{|B|}$ (Wei & Cheng, 1989)

Cluster membership indicator: $q(i) = \begin{cases} \sqrt{n_2 / n_1 n} & \textbf{if } i \in A \\ -\sqrt{n_1 / n_2 n} & \textbf{if } i \in B \end{cases}$

Normalization: $q^T q = 1, q^T e = 0$

Substitute q leads to $J_{Rcut}(q) = q^T (D - W) q$

Now relax q, the solution is 2nd eigenvector of L



Normalized Cut (Shi & Malik, 1997)

Min similarity between A & B: $s(A,B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$

Balance weights $J_{Ncut}(A,B) = \frac{s(A,B)}{d_A} + \frac{s(A,B)}{d_B}$ $\quad d_A = \sum_{i \in A} d_i$

Cluster indicator: $q(i) = \begin{cases} \sqrt{d_B / d_A d} & \textbf{if } i \in A \\ -\sqrt{d_A / d_B d} & \textbf{if } i \in B \end{cases}$ $\quad d = \sum_{i \in G} d_i$

Normalization: $q^T D q = 1, q^T D e = 0$

Substitute q leads to $J_{Ncut}(q) = q^T (D - W) q$

$$\mathbf{min}_{\mathbf{q}} q^T (D - W) q + \lambda (q^T D q - 1)$$

Solution is eigenvector of $(D - W) q = \lambda D q$

MinMaxCut (Ding et al 2001)

Min similarity between A & B: $s(A,B) = \sum_{i \in A} \sum_{j \in B} w_{ij}$

Max similarity within A & B: $s(A,A) = \sum_{i \in A} \sum_{j \in A} w_{ij}$

$$J_{MMC}(A,B) = \frac{s(A,B)}{s(A,A)} + \frac{s(A,B)}{s(B,B)}$$

Cluster indicator: $q(i) = \begin{cases} \sqrt{d_B / d_A d} & \textbf{if } i \in A \\ -\sqrt{d_A / d_B d} & \textbf{if } i \in B \end{cases}$

Substituting,

$$J_{MMC}(q) = \frac{1 + \sqrt{d_B / d_A}}{J_m + \sqrt{d_B / d_A}} + \frac{1 + \sqrt{d_A / d_B}}{J_m + \sqrt{d_A / d_B}} - 2 \qquad J_m = \frac{q^T W q}{q^T D q}$$

Because $\frac{dJ_{MMC}(J_m)}{dJ_m} < 0$ $\quad \min J_{\mathrm{mmc}} \Rightarrow \max J_{\mathrm{m}}(q)$

$$\Rightarrow \quad Wq = \xi D q \quad \Rightarrow \quad (D - W) q = \lambda D q$$

These spectral algorithms can be described from the **perturbation theory** point of view as well [19].

Normalized N-Cut and the grouping algorithm (Shi & Malik algorithm [2])

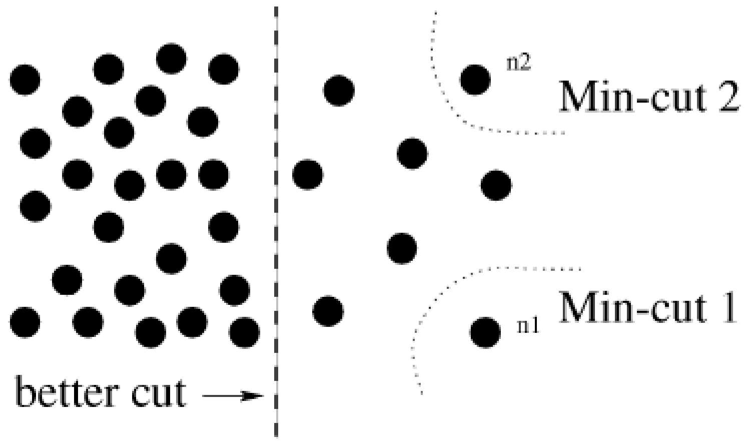
As described in [10],

A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ can be partitioned into two disjoint sets, A, B, $A \cup B = V$, $A \cap B = \emptyset$, the disassociation measure *normalized cut* (*Ncut*):

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)},$$

where $assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$ is the total connection from nodes in A to all nodes in the graph

As shown in the next figure, the Cut is needed to be normalized in order to rule out the isolated points to be chosen as min-cut sets.



A case where minimum cut gives a bad partition.

The same NCut problem is formulated in [3] as:

$$NCut(A, \overline{A}) = \left(\frac{1}{\text{vol } A} + \frac{1}{\text{vol } \bar{A}} \right) \sum_{i \in A, j \in \bar{A}} S_{ij}$$

Computing the optimal partitioning becomes equivalent to minimizing the Rayleigh quotient [10], [17]

$$min_{\boldsymbol{x}} Ncut(\boldsymbol{x}) = min_{\boldsymbol{y}} \frac{y^T(\mathbf{D} - \mathbf{W})\boldsymbol{y}}{y^T \mathbf{D} y},$$

with the condition $\boldsymbol{y}(i) \in \{1, -b\}$ and $y^T \mathbf{D1} = 0$.

As shown in [10], this is an NP hard problem, but can be approximately solved in the following manner as described in [1]:

```
Approximate-Cut Algorithm
Find an approximate sparsest cut in $G$.
Recurse on the pieces induced by the cut.
```

Now, relaxing y to take real values and solving (minimizing) the generalized eigenvalue system:

$$(\mathbf{D}-\mathbf{W})y = \lambda \mathbf{D}y$$

$$\text{i.e., } \mathrm{L}y = \lambda \mathbf{D}y$$

where L = D - W is the Laplacian. As defined in [19],

The unnormalized graph Laplacian: $L = D - W$

The normalized graph Laplacians: $L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$

$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W.$$

This can be converted to the standard eigenvalue problem [10] as:

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\boldsymbol{x} = \lambda \boldsymbol{x}$$

which can be solved in $O(n^3)$, where n is the number of nodes in the graph.

Now, with the following theorem for quadratic forms / Rayleigh quotient,

Let $\mathbf{A}$ be a real symmetric matrix. Under the constraint that $\boldsymbol{x}$ is orthogonal to the j-1 smallest eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{j-1}$, the quotient $\frac{\boldsymbol{x}^T \mathbf{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}$ is minimized by the next smallest eigenvector $\boldsymbol{x}_j$ and its minimum value is the corresponding eigenvalue $\boldsymbol{\lambda}_j$.

The second smallest eigenvector of the generalized eigensystem becomes the real valued solution to the normalized cut problem [10].

The grouping algorithm [10] is described as follows:

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 $(\mathbf{D} - \mathbf{W})\boldsymbol{x} = \lambda \mathbf{D}\boldsymbol{x}$ 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 necessary.

As described in [3],

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. In [10] it is shown that when there is a partitioning of $A, \bar{A}$ of I such that

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

then $A, \overline{A}$ is the optimal NCut and the value of the cut itself is $NCut(A, \bar{A}) = \lambda^L$.

As explained in [3], the eigenvectors possess piecewise constant property w.r.t. the partition. As computed in [1],

Given an (α, ϵ)-partition, spectral algorithm will find an $(\frac{\alpha^2}{36 \log^2 \frac{n}{\epsilon}}, 24\sqrt{\epsilon} \log^2 \frac{n}{\epsilon})$-partition

The results of the normalized cut algorithm [3], [9] are shown in the next figure. As can be noticed, the algorithm is used to find a 2-way-cut (to separate background from foreground in images), but can be generalized to k-way cuts by recursive application of the algorithm (as described in [10]) and can be used for finding outlying clusters as well.

[3] establishes a connection of random walk with the spectral partitioning as follows: The NCut algorithm lacks a satisfactory intuitive explanation. In particular, the NCut algorithm and criterion offer little intuition about (1) what causes x^L to be piecewise constant? (2) what happens when there are more than two segments and (3) how does the algorithm degrade its performance when x^L is not piecewise constant?

Markov Random Walk and normalized cuts

As explained in [3] and also shown in the next figure,

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

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

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.

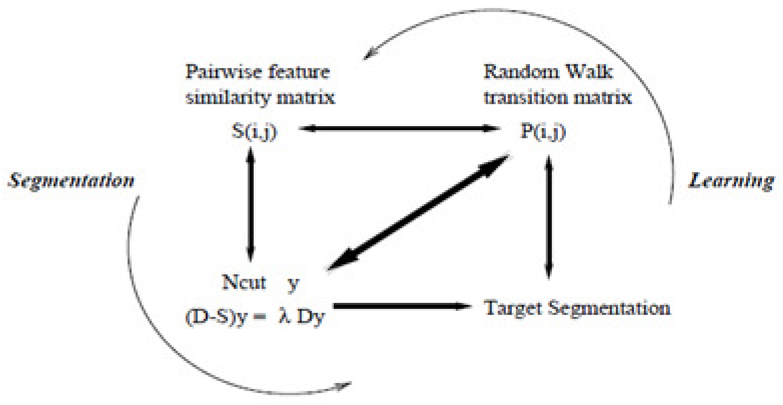
$$P_{AB} = \frac{\sum_{i\in A, j\in B} \pi_i^\infty P_{ij}}{\pi^\infty(A)} = \frac{\sum_{i\in A, j\in B} S_{ij}}{\text{vol}(A)}$$

From this it follows that

$$NCut(A, \bar{A}) = P_{A\bar{A}} + P_{\bar{A}A}$$

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 *mixinq time* of Markov random walks



The relationships between the similarity matrix S_{ij}, a Markov random walk, spectral segmentation such as Ncut, and image segmentation. The equivalance between key properties of the Markov random walk and the Ncut critirion offers a principled way of learning the feature similarity matrix S_{ij} in a probabilistic framework.

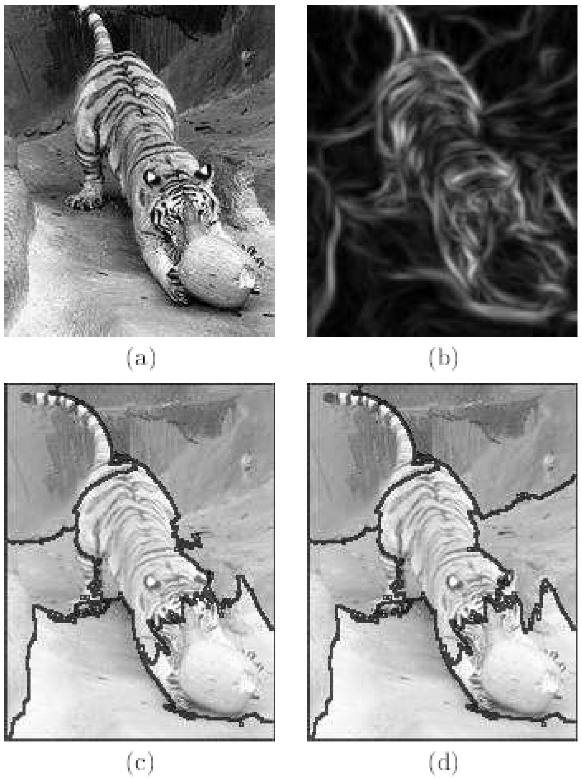
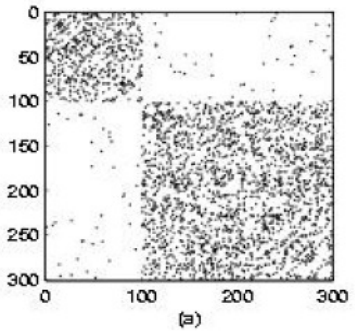


Image segmentation by the MNCut algorithm: (a) the original image; (b) the output of the edge detector; (c,d) segmentation by MNCut using the first 6 respectively 7 eigenvectors and k-means clustering. Two pixels are dissimilar if they are more than 30 apart or if they are separated by an edge; otherwise they are considered similar. Note that even with this simple similarity measure and in spite of the many stripes, most of the tiger is segmented correctly.

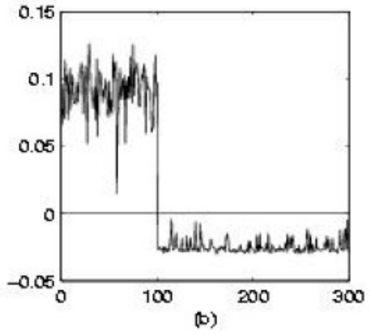
A simple example

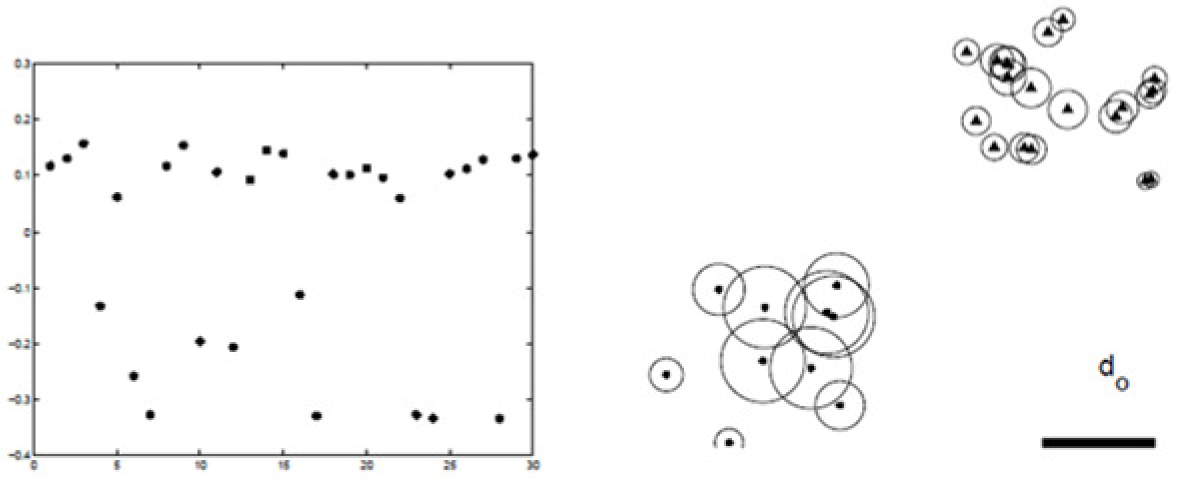
2 dense clusters, with sparse connections between them.

Adjacency matrix

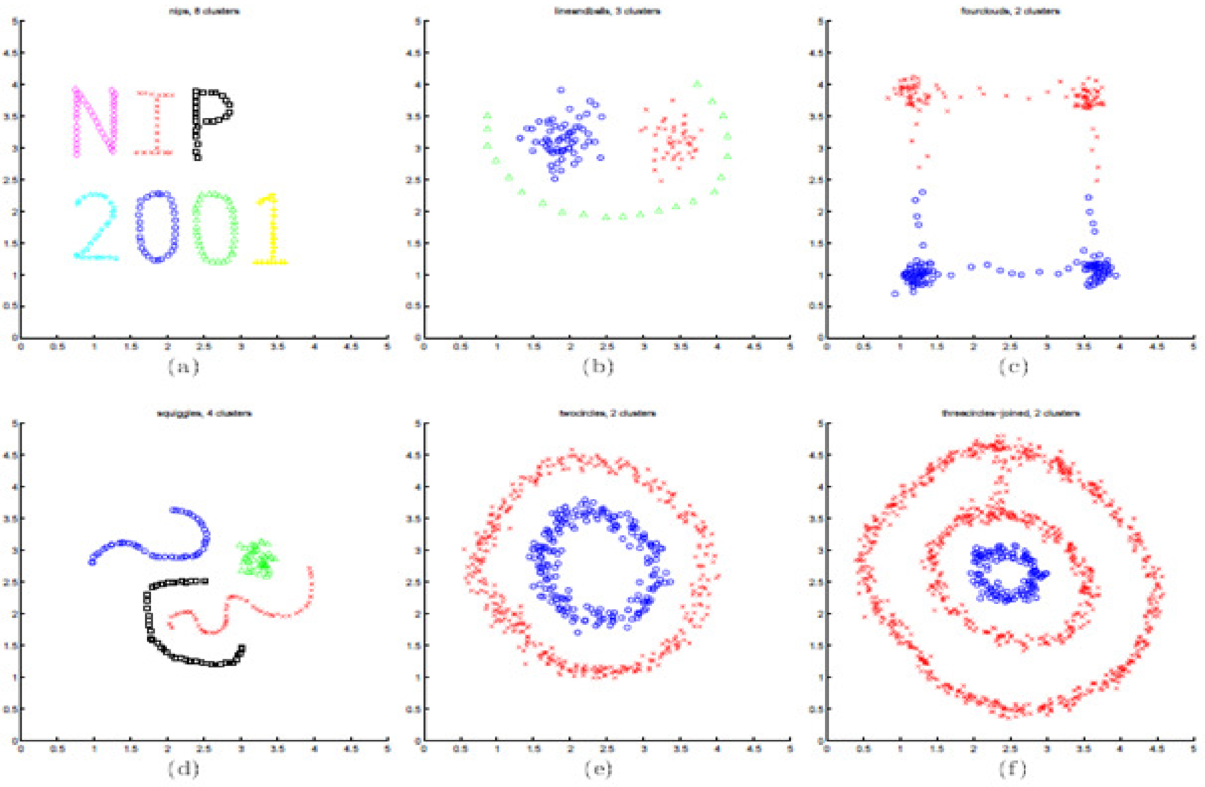


Eigenvector q_2





The Normalized-Cut algorithm on the dataset shown in Fig. On the left the value of the indicator eigenvector is shown (see text). The plot to the right, shows the corresponding grouping – triangles indicate positive values of the eigenvector, dots indicate negative values, the diameter of the circles surrounding each marker indicates the magnitude.



Clustering examples, with clusters indicated by different symbols (and colors, where available). Results from our algorithm, where the only parameter varied across runs was k.

Distributed Approach

1. Use Average Eigenvector heuristic

- Locally compute the Laplacian matrices and do Eigen analysis to find the local eigenvectors.
- By gossiping find the average of top k eigenvectors (should be piecewise constant, since average of piecewise constant vectors).
- Use k-,means to find the clusters from the global top k eigenvectors.
- Find outliers from the clusters.

2. Use Divide & Conquer Method for the symmetric Tridiagonal Eigenproblem [14]:

- Initially the entire data is at a centralized location, preprocess to find the global affinity matrix.
- Use **Householder transform** to recursive **tri-diagonalize** the matrix and send the blocks to each node (divide step: these tridiagonal matrices will have same eigenvalues as the original data matrices).
- Do eigen-analysis on the local tri-diagonal matrices and combine to get the global eigenvectors (conquer step).
- Use k-,means to find the clusters from the global top k eigenvectors.
- Find outliers from the clusters.

References

1. Ravi Kannan, Santosh Vempala, Adrian Vetta, On Clusterings: Good, Bad and Spectral
2. Yair Weiss, Segmentation using Eigenvectors: a unifying view
3. Marina Meila, Jianbo Shi, A random walks view of Spectral segmentation
4. Marina Meila, Jianbo Shi, Learning segmentation by random walks
5. Deepayan Chakrabarti, AutoPart: Parameter-Free Graph Partitioning and Outlier Detection
6. Marina Meila, William Pentney, Clustering by weighted cuts in directed graphs
7. F.R.K.Chung, Spectral Graph Theory, American Mathematical Society.
8. P. Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, V. Vinay, Clustering in large graphs and matrices.
9. P. Perona, W. Freeman, a factorization approach to grouping
10. Jianbo Shi and Jitendra Malik, Normalized Cuts and Image Segmentation
11. Andrew Y. Ng, Michael I. Jordan, Yair Weiss, On spectral clustering: Analysis and an algorithm
12. C. J. Alpert, So-Zen Yao, Spectral Partitioning: the more eigenvectors, the better
13. Yoram Gdalyahu, Daphna Weinshall, Michael Werman, A randomized algorithm for pairwise clustering
14. J.J.M. Cuppen, A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem
15. Hillol Kargupta, Vasundhara Puttagunta, An Efficient Randomized Algorithm for Distributed Principal Component Analysis from Heterogeneous Data
16. Souptik Datta, Hillol Kargupta, Uniform Data Sampling from a Peer-to-Peer Network
17. Inderjit S. Dhillon, Co-clustering documents and words using Bipartite Spectral Graph Partitioning
18. Chris Ding, a tutorial on spectral clustering
19. Ulrike von Luxburg, A Tutorial on Spectral Clustering