SPECTRAL CLUSTERING

A Dissertation submitted to the University of Hyderabad in partial fulfillment of

the degree of

MASTER OF TECHNOLOGY

in
Artificial Intelligence

by

Abdo Mahyoub Taher Nasser

08MCMI18

Department of Computer and Information Sciences

School of MCIS

University of Hyderabad

(P.O.) Central University, Gachibowli

Hyderabad – 500 046

Andhra Pradesh

India

April, 2011

SPECTRAL CLUSTERING

A Dissertation submitted to the University of Hyderabad in partial

fulfillment of the degree of

MASTER OF TECHNOLOGY

in
Artificial Intelligence

by

Abdo Mahyoub Taher Nasser

08MCMI18

Department of Computer and Information Sciences

School of MCIS

University of Hyderabad

(P.O.) Central University, Gachibowli

Hyderabad – 500 046

Andhra Pradesh

India

April, 2011

CERTIFICATE

This is to certify that the dissertation entitled Spectral Clustering submitted by **Abdo Mahyoub Taher Nasser** bearing Reg. No **08MCMI18** in partial fulfillment of the requirements for the award of Master of Technology in Computer Science is a bonafide work carried out by him under my supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Rajendra Prasad Lal
Project Supervisor,
Department of CIS,
University of Hyderabad

P.S.V.S. Sai Prasad
Project Supervisor,
Department of CIS,
University of Hyderabad

Head of Department,
Department of CIS,
University of Hyderabad

Dean,
School of MCIS,
University of Hyderabad

DECLARATION

I **Abdo Mahyoub Taher Nasser** hereby declare that this Dissertation entitled ***Spectral Clustering*** submitted by me under the guidance and supervisions of **Rajendra Prasad Lal** and **P.S.V.S. Sai Prasad** is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date Abdo Mahyoub Taher

08MCMI18

Dedication

This thesis is dedicated to my father, who taught me to be myself. It is also dedicated to my mother, and my wife, who taught me that even the largest task can be accomplished if it is done one step at a time.

Acknowledgment

I would like to express my sincere gratitude to **Rajendra Prasad Lal and P.S.V.S. Sai Prasad**, for valuable suggestions and keen persona interest throughout the progress of my course of research. Their invaluable guidance enabled me to accomplish my project successfully on time. I feel fortunate to have worked with them and owe a great deal for their constant encouragement, inspiring guidance and valuable suggestions throughout the project.

I am also grateful to the head of the department, Prof. C. R. Rao, for providing excellent facilities and such a nice atmosphere for doing this project.

I am extremely thankful to my friends Muhammad Farhan , Muaadh Farhan, Mufeed Ahmed Naji, Muzeer Vasum Abdul, and Ammar Abduallah Qasem for their encouragement to take up this course and for their support throughout the course with their love and affection.

My special thanks to my wonderful parents and my wife who have always supported me in all my decisions.

Abdo Mahyoub Taher Nasser

Abstract

Clustering is a fundamental problem in the areas like data mining, statistical machine learning, Pattern recognition, Information retrieval, etc. Popular clustering techniques, like K-means, don't perform well if the clusters are non-convex. Spectral Clustering is a new technique which is based on graphical representation and eigenvector selection of the data. It has shown remarkable improvement over K-means. It has been applied in various areas like Speech Separation, images segmentation, Wireless. In this project, we study and implement different Spectral Clustering algorithms. We also investigate its application in validating the sensors in distributed sensor networks.

Keywords: Spectral Clustering, Similarity graph, Graph Lapalcian.

Contents

List of figures

List of Algorithms

Chapter 1

Introduction

Clustering is one of the most widely used techniques exploratory data analysis with applications ranging from statistics, computer science, and biology to social sciences or psychology. In every scientific field, while dealing with empirical data, people try to identify groups of data points having similar behaviour. This similar behaviour is usually modeled by some mathematical functions of their characteristics. Based on this mathematical model, clusters are formed in a systematic manner. In the following sections we discuss few clustering techniques.

1.1 What is clustering?

Clustering is the process of grouping the data into classes or groups, so that objects within a group have high similarity in comparison to one another but are very dissimilar to objects in other groups. Clustering can be considered the most important unsupervised learning problem. Like every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A loose definition of clustering could be the process of organizing objects into groups whose members are similar. A cluster is, therefore, a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters. We can show this with a simple example(Fig1-1).

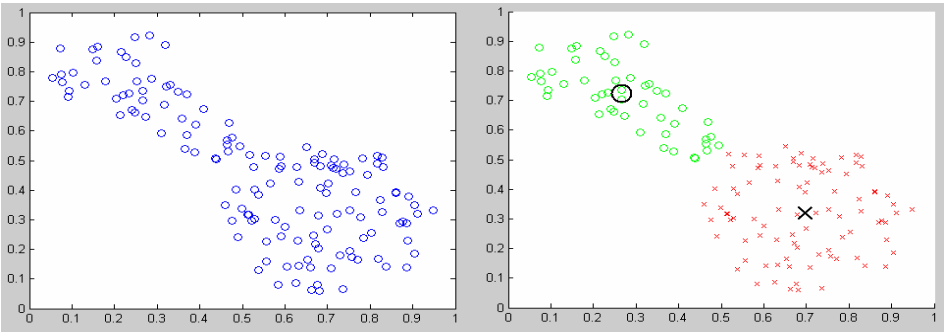

Figure 1.1 Original data in left side and data of the clustering in the right side.

1.2 Applications of Clustering

Clustering has wide applications in the following areas :

- Pattern recognition
- Spatial data analysis
- Creating thematic maps in GIS
- Image processing
- Economic science (especial market research)
- Biological sciences (Protein classification)
- Document classification

1.3 Different types of clustering

There are several clustering techniques, which are organized into the following categories: partitioning methods, hierarchical methods, density-based methods, grid-based methods, model-based methods, methods for high-dimensional data, and constant-based clustering.

- **Partitioning method**

  In this method, an initial set of k partitions are created, where parameter k is the number of partitions to be constructed. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. Typical partitioning methods include $k-means,\ k-medoids$, CLARANS[7], and their improvements.

- **Hierarchical method**

  This method creates a hierarchical decomposition of the given set of data objects. The method can be classified as being either agglomerative (bottom-up) or divisive (top-down), based on how the hierarchical decomposition is formed [7].

  Example:

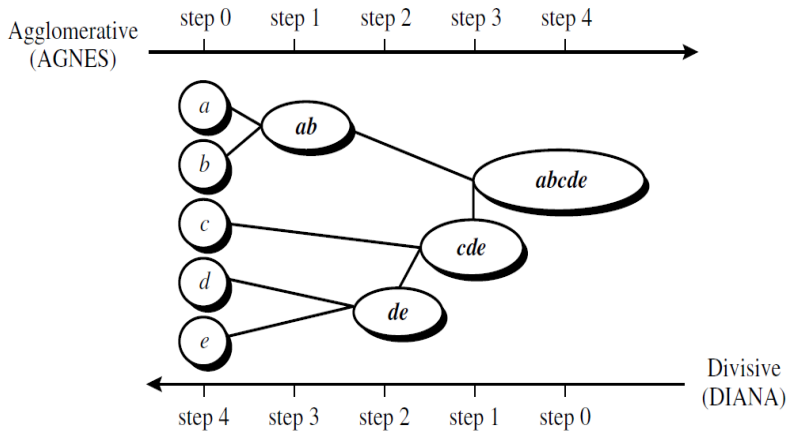

Figure 1.2Agglomerative and divisive hierarchical clustering on data objects {a,b,c,d,e}.

- **Density-based method**

  This method clusters objects based on the notion of density. It either grows clusters according to the density of neighborhood objects or according to some density function [7].

  Example:

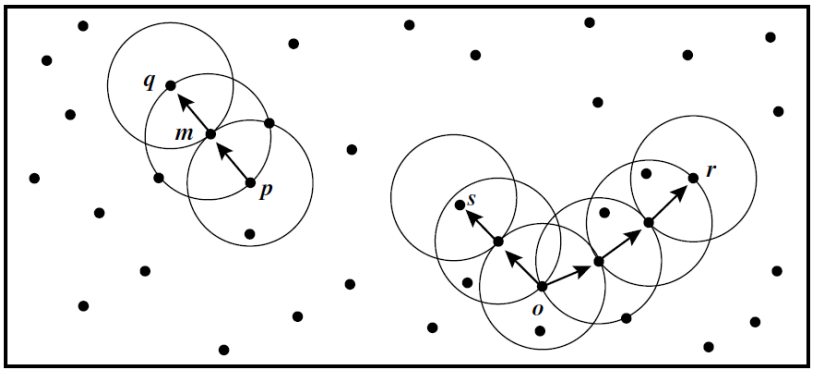

Figure 1.3 Density reachability and density connectivity in density-based clustering.

- **Grid-based method**

  This method first quantizes the object space into a finite number of cells that form a grid structure, and then performs clustering on the grid structure.

  Example:

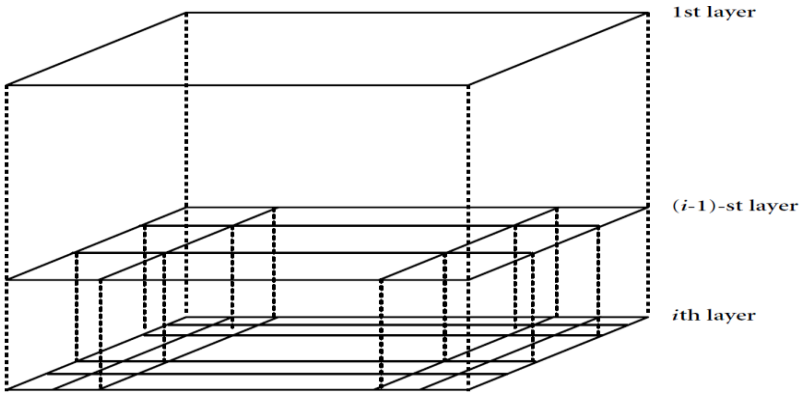

Figure 1.4 A hierarchical structure for STING clustering.

In the next section, we briefly describe few Partitioning methods for clustering**.**

1.4 K-means clustering algorithms and its variants

Partitioning clustering technique that attempts to divide the data set into a user-specified number of clusters, which are represented by centroids. The most well-known and commonly used partitioning methods are k-means, k-medoids, fuzzy c-means, *k*-medians [8]**,** and their variants.

1.4.1 The k-means Method

K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because the different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done.At this point we need to re-calculate k new centroids of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their

location step by step until no more changes are done. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

$$j = \sum_{j=1}^{k} \sum_{i=1}^{n} \left| x_i^{(j)} - c_j \right|^2$$

Where $|x_i^{(j)} - c_j|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster center c_j is an indicator of the distance of the n data points from their respective cluster centers.

K-means is considered a fast method because it is not based on computing the distances between all pairs of data points. However, the algorithm is still slow in practice for large datasets. The number of distance computations is nke where n the number of data points is, k is the number of clusters to be found, and e is the number of iterations required.

The algorithm is composed of the following steps:

Algorithm 1.1 k-means algorithm

1: Select the k the number of clusters.
2: Assign each point to the closest centroid.
3: When all points have been assigned to the centroid, recalculate the positions of the k centroids.
4: Repeat Steps 2 and 3 until the centroids no changed.

To accelerate k-means algorithm, we need the algorithm to satisfy three properties. First, it should be able to start with any initial centers, so that all existing initialization methods can continue to be used. Second, given the same initial centers, it should always produce exactly the same final centers as the standard algorithm. Third, it should be able to use any black-box distance metric, so it should not rely for example

on optimizations specific to Euclidean distance. One of the simple techniques that use triangle inequality is given in [6].

1.4.1.1 Example of k-means:

Initialization: Here we decide the number of *K* clusters , and initialize *K* centers (randomly). As we can see in figure 1.5.

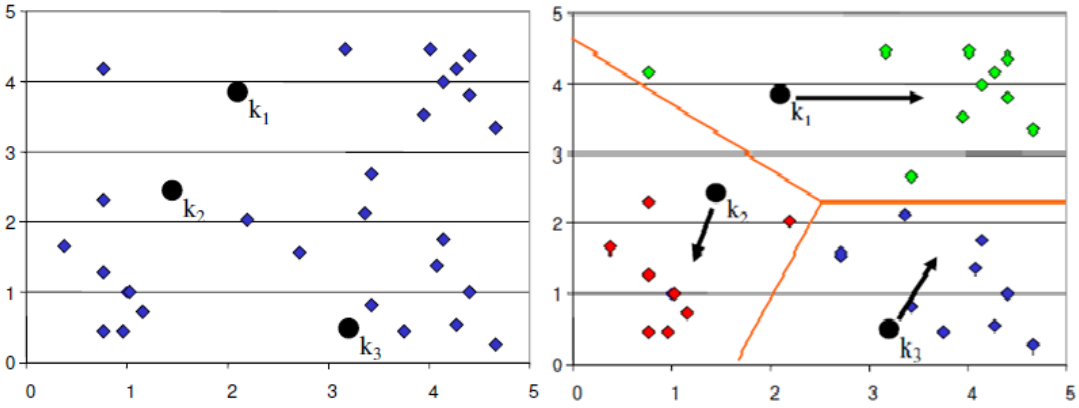

Figure 1.5 in the left side the initialization of the centers and the number of k clusters, also in the right side is the first iteration the means move to centers of clusters.

In the first iteration based on assigned all objects to the nearest center. Then the center moves to the mean of the clusters. In the Iteration 2 after moving centers, we need to reassign the objects to the new centers.

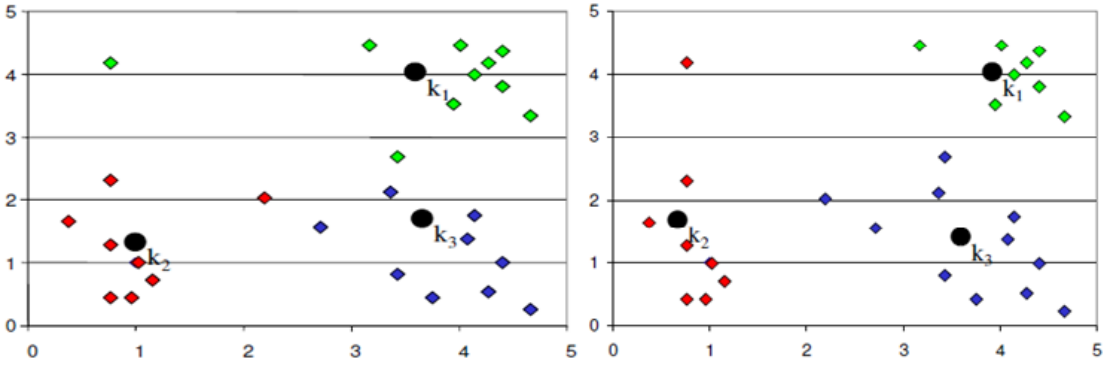

Figure 1.6 In left side is the 2nd iteration reassign objects to centers and right side reassign objects to new means.

Finally we reassign the objects to the new centers and move centers to the means, this process happen until no objects and centers changed their location. And then clustering performed is done.

1.4.1.2 Advantages and disadvantages

K-means is a classical algorithm with many advantages:

- With a large number of variables, K-means is computationally fast.
- K-means may produce tight clusters if they are globular.

But it also presents some disadvantages:

- We have to fix the number of clusters, so to predict what k should be used is difficult.
- The main problem is that it does not work well with non-globular clusters.
- Unable to handle noisy data and outliers.
- Need to specify k, the number of clusters in advance.
- The way to initialize the means was not specified. One popular way to start is to randomly choose k of the samples.
- The results depend on the initial values for means.
- It can happen that the set of sample closest to means is empty so that mean cannot be updated.

1.4.2 The k-medoids method

The ***k*-medoids algorithm** [8] is a clustering algorithm related to the *k*-means algorithm and the medoid shift algorithm. Both the *k*-means and *k*-medoids algorithms are partitioning (breaking the dataset up into groups) and both attempt to minimize squared error, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the *k*-means algorithm, *k*-medoids chooses data points as centers (medoids or exemplars).

k-medoid is a classical partitioning technique of clustering that clusters the data set of *n* objects into *k* clusters known *a priori*. It is more robust to noise and outliers as compared to *k*-means because it minimizes a sum of pair-wise dissimilarities instead of a sum of squared Euclidean distances. A medoid can be defined as the object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal.

1.4.3 The fuzzy c-means methods

Fuzzy c-means (FCM) [8] is a method of clustering which allows one piece of data to belong to two or more clusters. This method is frequently used in pattern recognition. It is based on minimization of the following objective function:

$$j_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\left|x_i - c_j\right|^2, 1 \le m \le \infty$$

where *m* is any real number greater than 1 u_{ij} is the degree of membership of x_i in the cluster *j*, x_j is the i^{th} of d-dimensional measured data c_i is the d-dimension center of the cluster, and $||*||$ is any norm expressing the similarity between any measured data and the center.

1.4.4 The *k*-medians method

K-medians clustering [8] is a variation of *k-means clustering* where instead of calculating the mean for each cluster to determine its centroid, one instead calculates the median. This has the effect of minimizing error over all clusters with respect to the 1-norm distance metric, as opposed to the square of the 2-norm distance metric (which *k*-means does).This relates directly to the *k*-median problem which is the problem of finding *k* centers such as the clusters formed by them are the most compact. Formally, given a set of data points *x*, the *k* centers c_i are to be chosen so as to minimize the sum of the distances from each *x* to the nearest c_i.

1.5 What is Spectral clustering?

Spectral clustering is also a partitioning clustering technique. But it doesn't use squared error minimization. Spectral Clustering technique clusters or represents the

dataset points using information obtained from the eigenvalues and eigenvectors of their similarity matrix. The methods are called spectral, because they make use of the spectrum of the similarity matrix of the data to cluster the points. In general, spectral clustering proceeds by analyzing the properties of eigenvectors of certain matrices which are derived from similarity matrices on the data points. The similarity matrix is a numerical representation of the graph constructed from the data points. Data points are combined as nodes and similarity function gives the edge-weight between a pairs of nodes. We are given an example below to give a fictional view of the spectral clustering. In the figure 1.7 we are given with a data set containing six data points. First we have to construct the similarity matrix to construct the graph and connect all the vertices by an edges where an edges representing the weights between vertices. Then the objective of clustering is to separate points in different groups according to their similarities. From the data given in form of similarity graph, we want to find the partition of the graph such as that the edges between different groups have very low weight we can see that from our graph vertices one and vertices five have very low weight (0.1) whereas the vertices one and vertices two have high weight (0.8). So, we construct a partition graph in our figure1.7 where edges have low weight for example nodes 1, 5, 4, and 3 they have low weight. Therefore we construct graph partitioning on the graph as we can see from the figure1.7. The spectral clustering technique gives an approximate of this partitioning in to clusters.

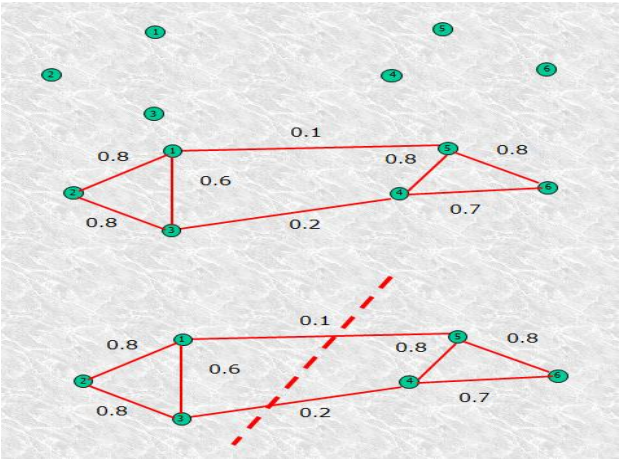

Figure 1.7 Construction the graph

1.6 Why spectral clustering?

- It is clear from figure1.9 the k-means algorithm doesn't perfume when clusters are non-globule. Whereas spectral clustering performs well in this case. The following are the advantages of spectral clustering.
- It is simple to implement in any linear algebra software such as Matlab.
- It is successful in many application such as an image segmentation etc.
- It does not suffer from the same drawback of $k-means$.
- Results obtained by spectral clustering often outperform the traditional k-means.
- Spectral clustering is based on the concept of similarity between points instead of distance.
- Spectral clustering is a more advanced algorithm compared to k-means.

1.6.1 Examples of spectral clustering:

1. Spiral rings

This 2-dimensional dataset consists of 100 data points and contains two spiral
This is a case where just running the k-means algorithm would fail in producing the rings. Intuitively, both, the outer and inner ring, should be grouped into separate clusters.

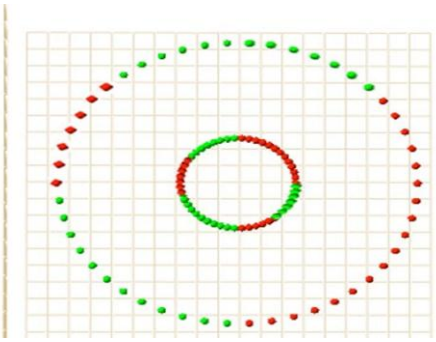

Figure 1.7 Original Dataset

Clustering visualization the following figure shows the results of using the k-means algorithm, and Spectral Clustering (with k = 2):

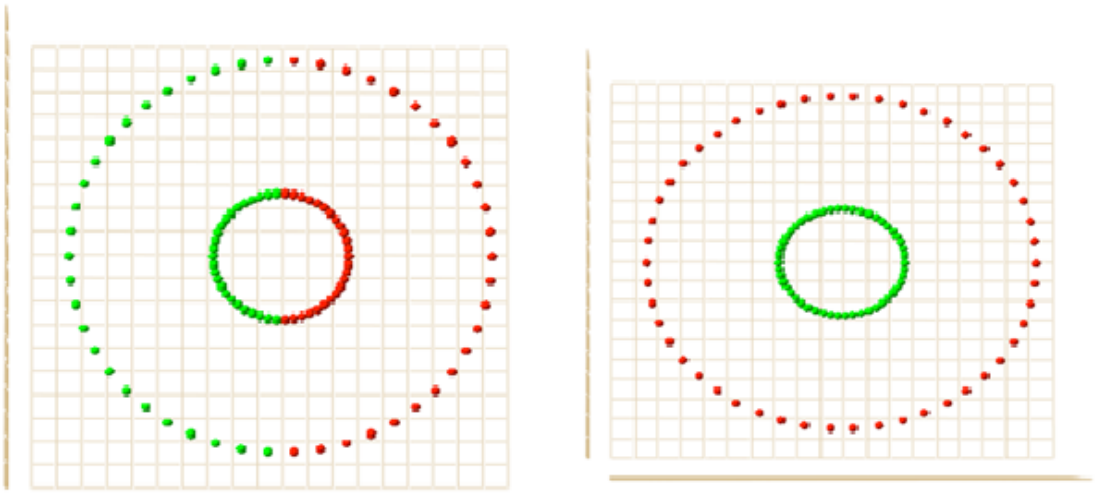

Figure 1.8 Spiral rings, clustering of the k-means in left side and clustering of the spectral clustering algorithm in the right side

The results of the spectral clustering algorithm (with k = 2 and σ = 9) are as expected:
Reasoning The k-means algorithm is not able to recognize this quite complex cluster shape because it would position the two centroids in the center of the left and right.
The spectral clustering algorithm overcomes this problem by constructing the Laplacian matrix from similarity graph that has been created for the original data points. Afterwards, the two eigenvectors for the first two smallest eigenvalues are computed. These two eigenvectors characterize connected components of the similarity graph. They can be assembled in a $n \times k$ matrix and the rows of this resulting matrix can be treated as new data points to be clustered with the k-means.

1.7 Thesis Organization

The project report is organized in to five chapters beginning with the introduction. In chapter 2, we discuss briefly about the Spectral Clustering Algorithms. In chapter3, we give the details of Variants of Spectral Clustering. In chapter 4 we will present Application in Distributed Sensor Networks and the results obtained by our proposed algorithms. Concluding remarks and scope for future work is given in chapter 5 followed by reference. The mathematical results regarding graph Lapalcian are given in the appendix.

Chapter 2

The Spectral clustering Algorithms

In this chapter, we will discuss in depth the Spectral clustering algorithm, and will present a brief discussion about Un-normalized Spectral clustering, normalized spectral clustering as described in [1]. We recollect some basic definition from graph theory.

2.1 Graph

Graph is a collection of some finite vertices and some finite edges which happen to connect these vertices, $G = (V, E)$ where G is a graph, V are vertices and E edges.

2.1.1 Directed graph

It is a pair $G = (V, E)$ of a set V, whose elements are called vertices or nodes, a set E of edges called arcs, directed edges, or arrows.

2.1.2 Undirected graph

An undirected graph is a graph in which the nodes are connected by undirected arcs. An undirected arc is an edge that has no arrow. Both ends of an undirected arc are equivalent, there is no head or tail. Therefore, we represent an edge in an undirected graph as a set rather than an ordered pair $G = (V, E)$ with the following properties:

1. The first component, V, is a finite, non-empty set. The elements of V are called the vertices of G.
2. The second component, E, is a finite set of sets. Each element of E is a set that is comprised of exactly two (distinct) vertices. The elements of E are called the edges of G.

2.2 Adjacency matrix of graph

It is a matrix which expresses the relationship between two nodes in the graph. For a graph containing n nodes, let us assume that we have an $n \times n$ adjacency matrix, in which the entry (i, j) corresponds to the weight of the edge between the nodes $i \; and \; j$. This essentially corresponds to the similarity between nodes $i \; and \; j$. This entry is denoted by w_{ij} and the corresponding matrix is denoted by W. This matrix is assumed to be symmetric, since we are working with undirected graphs. Therefore, we assume that $w_{ij} = w_{ji}$ for any pair (i, j). All diagonal entries of the matrix W are assumed to be 0.

2.3 Degree matrix (D)

It is a diagonal matrix which contains information about the degree of each vertex. In other words it is simply a diagonal matrix, in which all entries are zero except for the diagonal values. The diagonal entry $d(i, i)$ is equal to the sum of the weights of the incident edges.

$$\mathrm{d(i,i)} = \sum_{\mathrm{j=1}}^{\mathrm{n}} \mathrm{w_{ij}}$$

2.4 Similarity graphs

Given a set of data points $x_i, \ldots, x_j$ and some notion of similarity $S_{ij} \geq 0$ between all pairs of data points x_i, and, x_j , the intuitive goal of clustering is to divide the data points into several groups such as the points in the same group which are similar and points in different groups are dissimilar to each other. If we do not have more information than similarities between data points, a nice way of representing the data is in form of the $Similarity \; graph \; G = (V, E)$. Each vertex v_i in this graph represents a data point x_i . Two vertices are connected if the similarity s_{ij} between the corresponding data points x_i, and, x_j is positive or larger than a certain threshold, and the edge is weighted by s_{ij}. The problem of clustering can now be reformulated

using the similarity graph: we want to find a partition of the graph such that the edges between different groups have very low weights (which means that points in different clusters are dissimilar from each other) and the edges within a group have high weights (which means that points within the same cluster are similar to each other).

2.5 Similarity Matrix

Similarity matrix expresses the similarity between two data points. It is the adjacency matrix of the similarity graph. The elements of a similarity matrix measure pair-wise similarities of objects, the greater similarity of two objects, the greater the value of the measure.

2.6 Different ways of construction similarity matrix

There are several popular constructions to transform a given set $x_i, \dots, x_j.$ of data points with pair-wise similarities s_{ij} or pair-wise distances d_{ij} into a graph. The most common similarity graphs are the followings:

1. The ε - neighborhood graph

Here we connect all points whose pair-wise distances are smaller than ε, threshold value. As the distances between all connected points are roughly of the same scale (at most ε), weighting the edges would not incorporate more information about the data to the graph. Hence the ε - neighborhood graph is usually considered as an un-weighted graph.

Example:

This data set is a 500 points of dimension 3. The constructed graph is epsilon neighborhood graph , the dataset which contains three cluster of Gaussians with epsilon parameter is $\boldsymbol{\varepsilon}$ = 0.50. It is clear in the figure 2.1, that the numbers of the clusters are 3. We see some points connected to other point in different scales, because the distances between data points are different from the space.

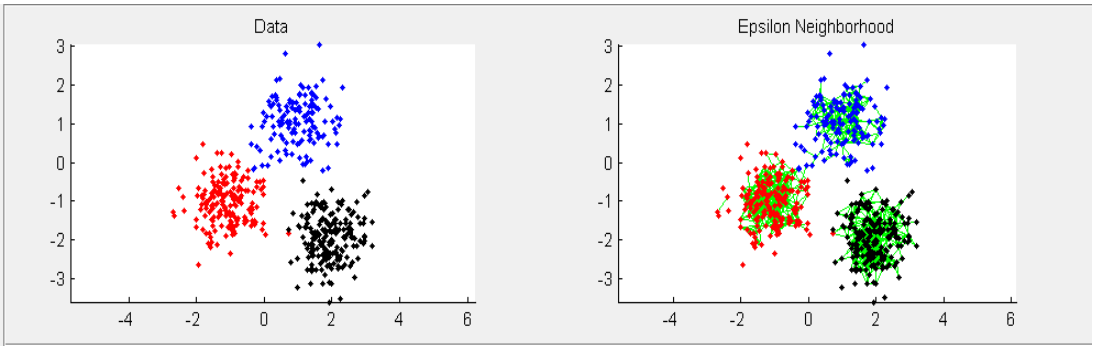

Figure 2.1 ε -Neighborhood graph

2. *k* -nearest neighbor graphs:

In this graph, a vertex v_i is connected with vertex v_j, when v_j is among the k-nearest neighbors of v_i. This concept would lead to a directed graph. However, there exists two possibilities to convert this directed graph into an undirected graph. The simplest method is to ignore the directions of the edges. This kind of graph is denoted as k-nearest neighbor graph. The other Possibility is to connect two vertices v_i and v_j if v_i is among the k-nearest neighbors of v_j and vice-versa. This kind of graph is denoted as mutual k-nearest neighbor graph.

Example:

This data set is a 500 points of dimension 2, the type of the graph is symmetric k-nearest neighborhood graph , the dataset which contains five cluster of Gaussians, the number k of neighbors is $\boldsymbol{k = 11}$. We see in the figure 2.2, the k-nearest neighbor graph can connect points on different scales as points in low density can connect with points in high density. This is as property of k-nearest neighbor graph.

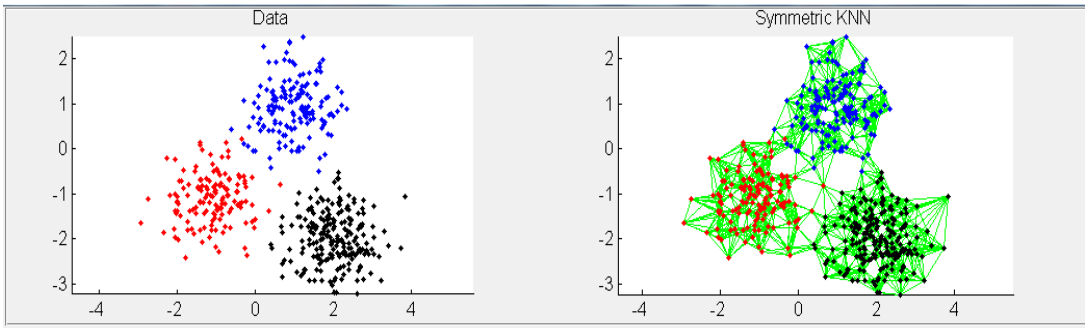

Figure 9 Symmetric k-nearest Neighborhood graph

3. Fully connected graph:

Here we simply connect all points with positive similarity with each other, and we weight all edges by s_{ij}. As the graph should represent the local neighborhood relationships, this construction is only useful if the similarity function itself models local neighborhoods. An example for such a similarity function is the Gaussian similarity function $\boldsymbol{s(x_i,\ x_j) = exp(-|x_i - x_j|^2/(2\sigma^2)}$ where the parameter σ controls the width of the neighborhoods. This parameter plays a similar role as the parameter ε in case of the ε-neighborhood graph.

Example:

This data set is a 500 points of dimension 2, the type of the graph is Fully connected graph , the dataset which contains 3 cluster of Gaussians, the sigma $\sigma = 19$. The figure 2.3 shows the dataset and the fully connected graph.

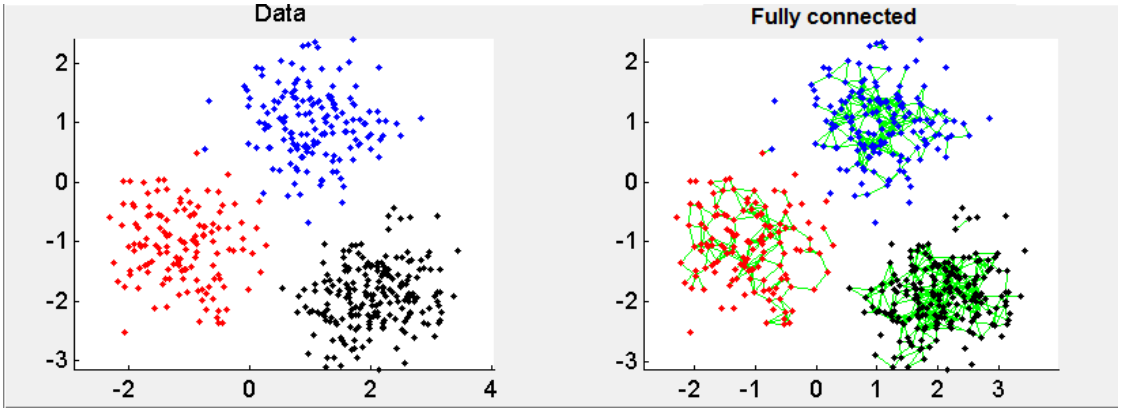

Figure 2.3 Fully connected graph

2.4 Graph Laplacian

The spectral algorithms presented in our project foot on eigenvectors of Laplacians, which are a combination of the weight and the degree matrix. The main tools for spectral clustering are graph Laplacian matrices. It is the difference of diagonal matrix of vertices degrees and adjacency matrix. Thus we first have to introduce the types of matrices which are most commonly used in spectral clustering, called graph Laplacians. In this section we introduce the un-normalized and normalized graph Laplacian on finite weighted graphs. The types of Laplacian are normalized and un-normalized.

2.7.1 The un-normalized graph Laplacian

The un-normalized graph Laplacian matrix is defined as

$$L = D - W \qquad (1)$$

Where D is the degree matrix defined as the diagonal matrix with the degrees and W is the similarity matrix. It has the following properties:

- L is always symmetric and positive semi-definite (non-negative eigenvalues)
- The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector 1.
- The multiplicity of 0 as an eigenvalue of L is the number of connected components of graph

- The smallest eigenvalue of L is called the spectral gap.

2.7.2 The normalized graph Laplacian

There exists two matrix which are called normalized graph Laplacians, Both matrices are closely related to each other and are defined as

$$L_{Sym} = D^{-1/2}D^{-1/2} = I - D^{-1/2}WD^{-1/2} \quad (2)$$
$$L_{rw} = D^{-1}L = I - D^{-1}W \quad (3)$$

We denote the first matrix is obtained through a symmetric matrix, while the second one is the result of performing a row sum (random walk) normalization on the un-normalized Laplacian. Again, these two matrices have similar spectral properties like the un-normalized graph Laplacian, except that for L_{sym} the eigenspace is spanned by $S^{1/2}$ multiplied with the indicator vectors. Some properties are

- λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvalue of L_{rw} with eigenvector $w = D^{1/2}u$
- λ is an eigenvalue of L_{rw} with constant one vector 1 as eigenvector.0 is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}1$
- λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigeproblem $Lu = \lambda Du$

2.8 Spectral Clustering Algorithms

There are several spectral clustering algorithms Proposed by several authors [1] . The efficiency of the spectral clustering algorithms mainly based on the fact that it does not make any assumptions on the form of the clusters. This property comes from the mapping of the original space to a eigenspace. Algorithms differ basically in the number of eigenvectors they use for portioning.

Algorithms

Let us describe some different algorithms to implement Spectral Clustering. The main difference between them is the Laplacian Graph they use to implement the embedding. All these algorithms present the same structure. They receive as input an arbitrary subset $\{x_1, x_2, \dots, x_n\}$ with $x_i \in R^n$. All the algorithms use the weights $w_{ij} = (S_{ij})_{ij}, \dots, n,$ according to some symmetric non-negative similarity function. Now we would like to state the most common spectral clustering algorithms. We assume that our data consists of n "$points$" $\{x_1, x_2, \dots, x_n\}$ which can be arbitrary objects. We measure their pair-wise similarities $S_{ij} = S(x_i, x_j)$ by some similarity function which is symmetric and non-negative, and we denote the corresponding similarity matrix by $S = (s_{ij})_{ij}, \dots, n$.

2.8.1 Un-normalized spectral clustering

The first algorithm we present is based on the Un-normalized Laplacian Graph. We have defined the Un-normalized Laplacian Graph L as

$$\boldsymbol{L = D - W}$$

Algorithm 2.1 Un-normalized Spectral Clustering

Input: Similarity matrix $S \in \mathbb{R}^{nxn}$, number k of Cluster to be constructed.

Output: k clusters $A_1, A_2, \dots, A_k$.

1: Construct a similarity graph using the ε-neighborhood graphs, or K-nearest neighbor graph, or fully connected graph.

2: Compute Wand D matrices.

3: Compute the un-normalized Laplacian $L = D - W$.

4: Compute the first k eigenvectors $u_1, \dots, u_k$ of L.

5: *Let* $U \in \mathbb{R}^{n \times k}$ *be the* matrix containing the vectors $u_1, \dots, u_k$ as columns.

6: For $i = 1, \dots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.

7: Cluster the points (y_i) $i=1,\ldots, n$ *in* $\mathbb{R}^k$ *with k-means algorithm into clusters* $C_1,\ldots, C_k$.

8: Construct $A_{i=}\{j|y_i \in C_i\}$.

2.8.2 Normalized Spectral Clustering

In this case, we present two different versions of the Spectral Clustering algorithm, according to the two different normalized Laplacian Graphs considered. The first one, the symmetric Laplacian, is given by the expression

$$L_{sym} = D^{-1/2} L D^{-1/2} = 1 - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

The second one is the called Random Walk Laplacian, as it is closely related with random walks.

$$L_{rw} = D^{-1} L = 1 - D^{-1} W$$

Algorithm 2.2 Normalized Spectral Clustering according to[1]

Input: Similarity matrix $S \in \mathbb{R}^{nxn}$, number k of Clusters to be constructed.

Output: K clusters $A_1, A_2,\ldots, A_k$.

1: Construct a similarity graph using the ε-neighborhood graphs, or k-nearest neighbor graph, or fully connected graph.

2: Compute $W and\ D$ matrices.

3: Compute the un-normalized Laplacian $L = D - W$.

4: Compute the first k Generalized eigenvectors $u_1, \ldots, u_k$ of of the generalized eigeproblem $Lu = Du$.

5: Let $U \in \mathbb{R}^{n\times k}$ *be the* matrix containing the vectors $u_1, \ldots, u_k$ as columns.

6: For i $= 1, \ldots,$ n, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.

7: Cluster the points (y_i) $i=1,\ldots, n$ in $\mathbb{R}^k$ with k-means algorithm into clusters $C_1, \ldots, C_k$.

8: Construct $A_i = \{j | y_i \in C_i\}$.

The second normalized algorithm2.3 is based on the Normalized Symmetric Laplacian Graph and it includes a normalization step. Note that this algorithm uses the generalized eigenvectors of L, which according to Proposition 3 correspond to the eigenvectors of the matrix L_{rw} [1].

Algorithm 2.3 Normalized spectral clustering [1]

Input: Similarity matrix $S \in \mathbb{R}^{nxn}$, number k of Cluster to be constructed.

Output: k clusters $A_1, A_2, \ldots, A_k$.

1: Construct a similarity graph using the ε-neighborhood graphs, or k-nearest neighbor graph or fully connected graph.

2: Compute $W and D$ matrices.

3: Compute the normalized Laplacian L_{sym}.

4: Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{sym}.

5: Let $U \in \mathbb{R}^{n*k}$ *be the* matrix containing the vectors $u_1, \ldots, u_k$ as columns.

6: From the matrix $T \in \mathbb{R}^{nxk}$ from U by normalizing the row to norm 1, that is set $t_{ij} = u_{ij} / \sum_k u_{ik}^2) 1/2$.

7: For i = 1, … , n, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.

8: Cluster the points (y_i) $i=1,\ldots, n$ in $\mathbb{R}^k$ with k-means algorithm into clusters $C_1, \ldots, C_k$.

9: Construct $A_i = \{j | y_i \in C_i\}$

2.5.1.3 Example:

We illustrate the above algorithms by the following examples taken from [5].

Case 1: Two moon

In this example we are going to show the implementation of spectral clustering with data set is a 500 data points of dimension 2 the type of the data set is two moon and the graph type is epsilon neighborhood graph and the epsilon parameter ε = 0.51, with the number of k=5 clusters. The figure 2.4 show the steps

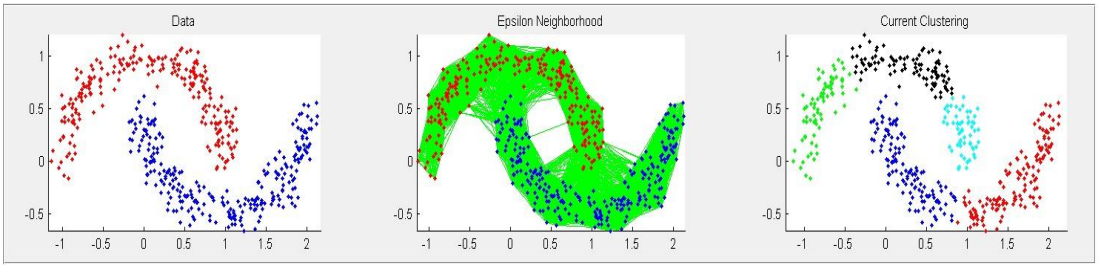

Figure 2.4 Spectral clustering using epsilon-Neighborhood graph

Case 2: Two Gaussian

In this example we are going to show the implementation of spectral clustering with data set which is a 500 data point of dimension 2. The type of the data set is two Gaussian balanced and the graph type is k-nearest neighborhood graph and the number k of neighbors k = 27. The numbers of clusters are 4. The figure 2.5 shows the steps.

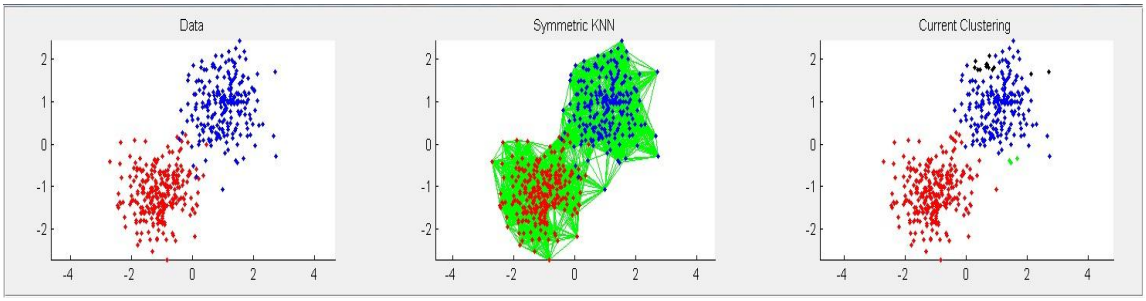

Figure 2.5 Spectral clustering using k-nearest Neighborhood graph

Case 3: Two Gaussian with difference variance

Each point is disturbed by a Gaussian noise of different variance: The data set is a 500 data point sample of dimension 2. The graph type is mutual $k-nearest\ neighborhood$ graph with number k of neighbors k=45, and kernel width sigma = 7.3. The number of clusters have been constructed are four. So we see the operation in the figure 2.6.

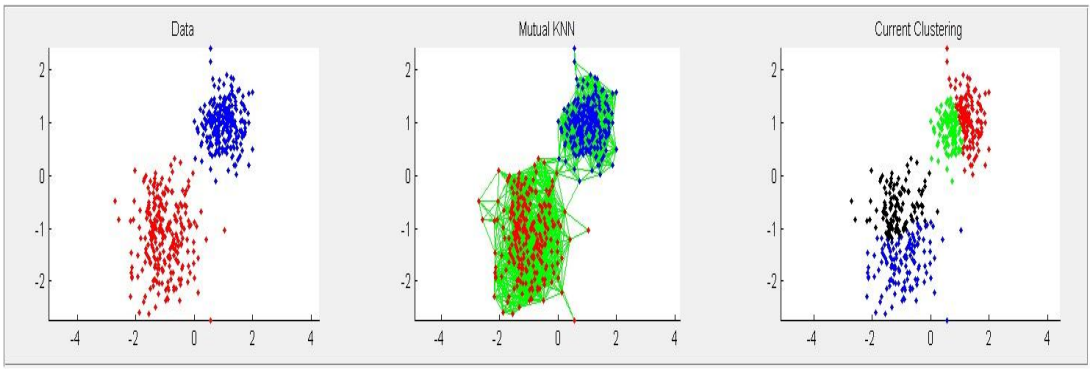

Figure 2.6 Spectral clustering using Mutual k-nearest Neighborhood graph

2. 9 Implementations

For the spectral clustering implementation the un-normalized clustering, and the normalized spectral clustering according to Ng, Jordan and Weiss [1] have been used. First we start with the normalized spectral clustering , where the algorithm takes the similarity matrix $S \in \mathbb{R}^{nxn}$ as input, where the number of data points in the set are

200 data points of dimensional 2, and the number k of clusters to construct as input is 5.

The similarity graph which we have considered it here the fully connected graph. In this graph, all points are connected with positive similarities. We put the sigma in the fully connected graph $\sigma = 30$. Based on the similarity graph which has been constructed, we compute the normalized graph Laplacians (random walk)

$$L_{rw} = D^{-1}L = I - D^{-1}W$$

Where D is the degree matrix defined as the diagonal matrix with degrees and W the weight matrix . Once the normalized graph Laplacians has been computed, then the spectral clustering algorithm computes it is first eigenvectors of L_{rw} which is corresponding to the smallest eigenvalues. And then we build a new matrix called $U \in \mathbb{R}^{n*k}$ w with the computed eigenvectors of random walk and we store it as columns. Then we construct new matrix called T by normalizing the row sum to norm 1.

Finally we interpret the rows of T as new data points $y_i \in \mathbb{R}^k$. Then k-means takes the data points as input to detect the number of clusters. The result of input and output are shown in figure 2.7, and figure 2.8.

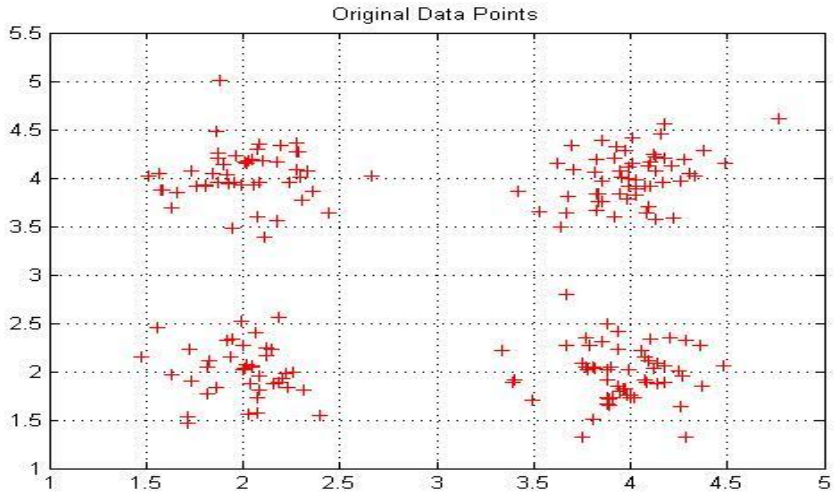

Figure 2.7 Spherical clusters, Original data set using Un-normalized spectral clustering

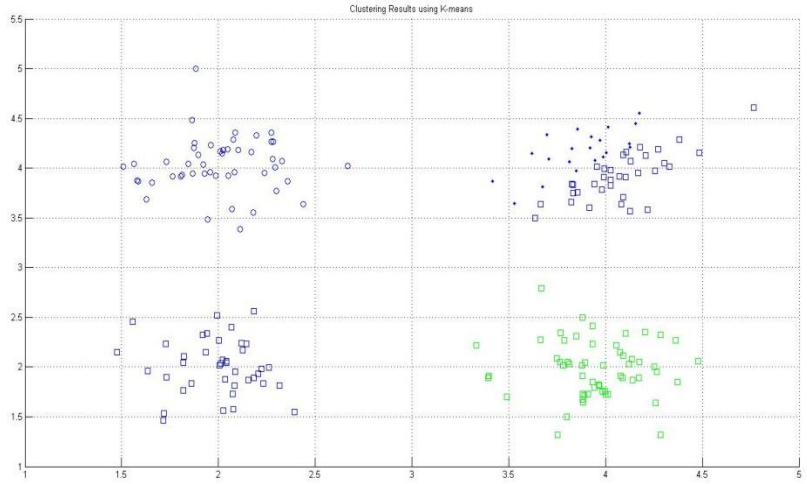

Figure 2.8 Clustering result using Un-normalized spectral clustering

Second case, we work with the un-normalized spectral clustering algorithm where the algorithm takes the similarity matrix $S \in \mathbb{R}^{nxn}$ as input where the number of data points in the set are 200 of dimensional 2, and the number k of clusters to construct as input are 4.

The similarity graph which we have considered it here, is ε-neighborhood graph where in this graph we make the epsilon $\varepsilon = 0.50$.

Once the similarity graph which has been constructed, we compute the un-normalized graph Laplacians by

$$\boldsymbol{L = D - W}$$

Where D is the degree matrix defined as the diagonal matrix with degrees and the weight matrix. based on the un-normalized graph Laplacian has been computed, then the spectral clustering algorithm compute it is first eigenvectors of L which it is corresponding to the smallest eigenvalues. And then we build a new matrix called $U \in \mathbb{R}^{n*k}$ with the computed eigenvectors of L we store it as columns.

Finally we interpret the rows of U as new data points $y_i \in \mathbb{R}^k$. Then k-means takes the data points as input to detect the number of clusters. The following figures show the resulting clusters by running the algorithm on the dataset with k = 4 and $\boldsymbol{\varepsilon = 0.50}$.

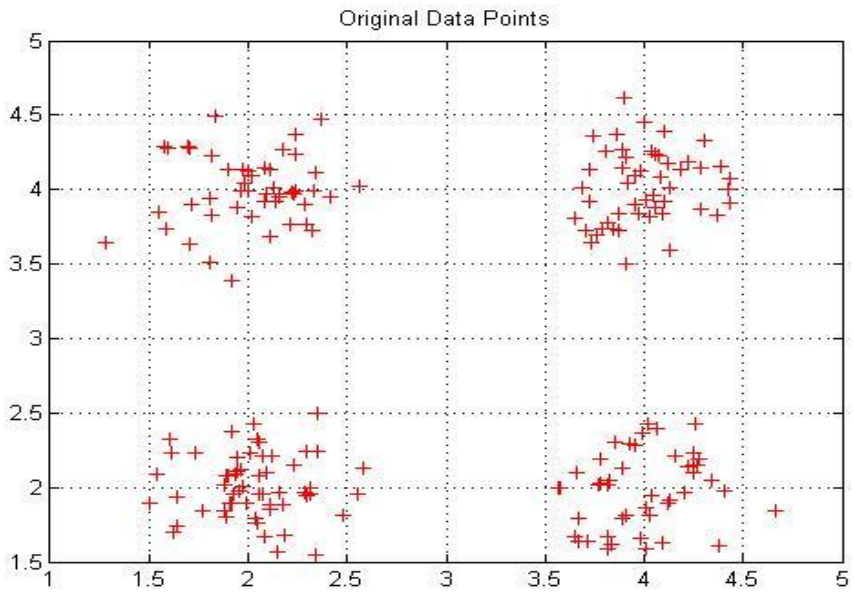

Figure 2.9 Spherical clusters, original data using normalized spectral clustering

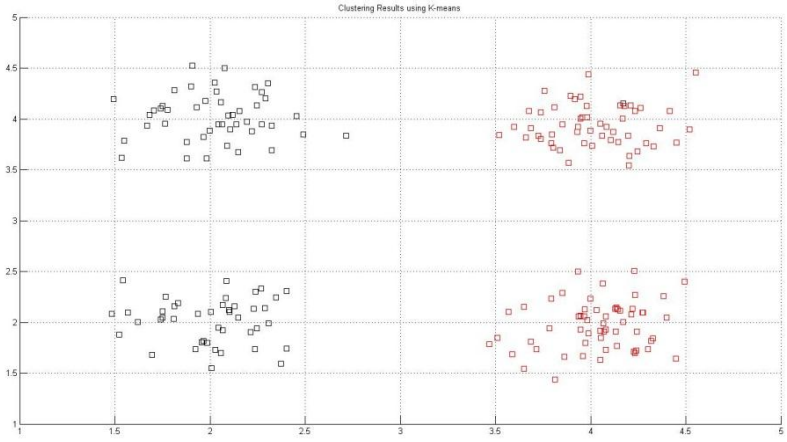

Figure 2.10 Spherical clusters, clustering result using normalized spectral clustering

Chapter 3

Variants of Spectral Clustering

3.1 Introduction

In our work, we study two variants of spectral clustering, namely value-based, and sign-based spectral clustering. Sign-based spectral clustering performs data grouping based on signs of components in the eigenvectors of the input of the matrix which represent some inter-node relationship. Value-based spectral clustering works similarly subject to magnitude of the components of the eigenvectors. We give detail description of each type along with implementation and application in this chapter.

3.2 Value-Based clustering

Before we explain these variants, we give an example which will be referred throughout this chapter.

3.2.1 Example: line detection

Here we give an illustration concerning detection of lines based on the Hough transform. As depicted in Figure3.1, we are given 14 points, $Pi's$, lying on five lines, Line $j's$. Where each point will "vote" on every line as follows: vote 10 if the point is on the line and the line intersects another line, 5 if the point is on the line and the line does not intersect any other line, 2 if the point is on an intersecting line, and 1 otherwise. Figure 3.2 shows the 5×14 point-to-line matrix A capturing the votes. Note that votes are discrete in the sense that they assume only a few values.This is analogous to the discrete measurements imposed by the quantization step function in the next chapter sensor-target example.

Value-based Clustering

A subset of nodes is said to form a value-based cluster if, for each eigenvector of the matrix A^TA, its components corresponding to these nodes assume the same value.

Example:

To illustrate value-based clustering, we consider a matrix A, of the line detection example

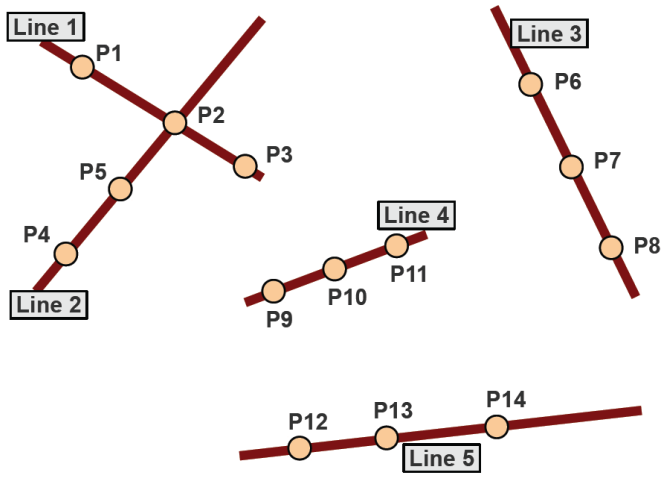

Figure 3.1 Line detection via Hough transform in a scenario

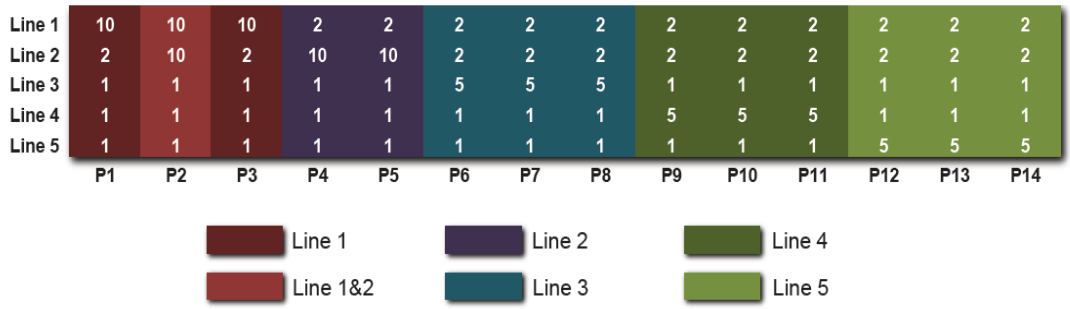

Figure 3.2 Discrete 5×14 point-to-line matrix A resulting from line detection

We compute the eigenvectors of the A^TA and cluster according to the values of the components.

E.v. 1	0.34	0.55	0.34	0.34	0.34	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
E.v. 2	0.50	0.00	0.50	(0.50)	(0.50)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E.v. 3	(0.16)	(0.38)	(0.16)	(0.16)	(0.16)	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
E.v. 4	0.00	0.00	0.00	0.00	0.00	0.35	0.35	0.35	(0.45)	(0.45)	(0.45)	0.09	0.09	0.09
E.v. 5	0.00	0.00	0.00	0.00	0.00	(0.31)	(0.31)	(0.31)	(0.15)	(0.15)	(0.15)	0.46	0.46	0.46
	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14

Line 1 Line 2 Line 4

Line 1&2 Line 3 Line 5

Figure 3.310 The five eigenvectors of A^TA

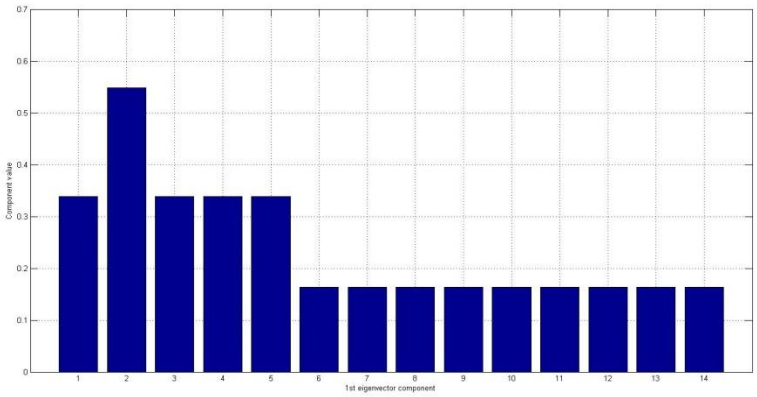

Figure 3.4 1st eigenvector of

Figure 3-4 shows there are three clusters, one is of size nine consisting of points P_6 to P_{14}, another is of size four consisting of points P_1,P_3,P_4, and P_5, and the third cluster contains only one point P_2.

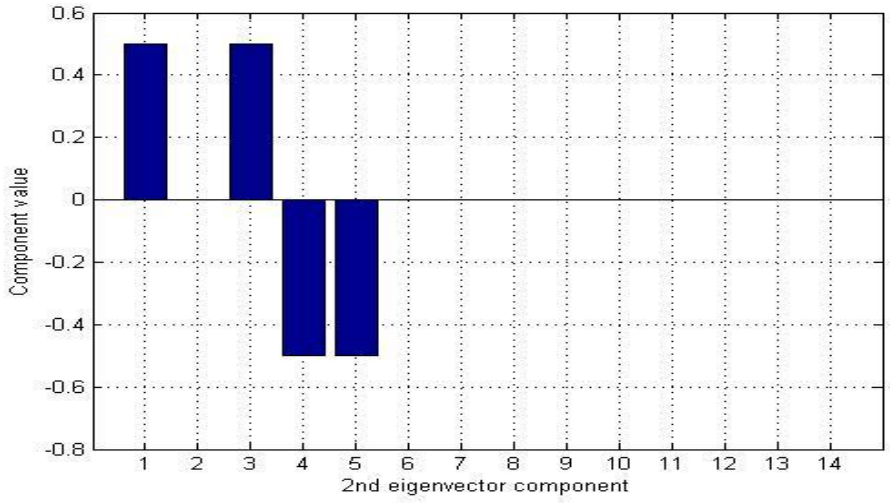

Figure 3.5 2nd eigenvector of A^TA

Figure 3-5 displays the second eigenvector, which shows P_1 and P_3 forms a cluster and P_4 , P_5 forms another cluster. This gives a refinement of clusters obtained from the first eigenvalue. This conforms to what we see in the figure 3-1. We can get all the clusters by refining further using the other eigenvectors of A^TA.

Next we discuss sign-based clustering, which doesn't use all the eigenvectors for clustering.

3.3 Sign-based Clustering

A subset of nodes is said to form a sign-based cluster if, for each eigenvector of the $\boldsymbol{A^TA}$ matrix, the components corresponding to these nodes assume the same signum. That is, for any given eigenvector, its components corresponding to these nodes are either all positive or negative or zero.

For example, if one of the vectors has a positive value for a component position, then the other must have a negative value for that component position. We note that sign-identical or sign-complementary vectors cannot be orthogonal since their inner product is positive or negative, respectively, rather than zero.

A theorem from [3] says that by examining the component signs of any K eigenvectors of A^TA, where $1 < K \leq \text{rank}(A^TA)$, it is guaranteed that we can identify at least K sign-based clusters. That means we need not look at many eigenvectors if we are only interested in finding few sign-based clusters. We recall from linear algebra that that eigenvectors corresponding to different eigenvalues are orthogonal.

Note that since we assume $\boldsymbol{A^TA}$ is nonnegative and irreducible, by the Perron-Frobenius theorem the principal eigenvector corresponding to the largest eigenvalue must have all of its components being of the same sign that is, they are either all positive or all negative. Any other eigenvector must have components with different signs, due to its orthogonality to the principal eigenvector. Thus, every non-principal eigenvector is associated with a positive and also a negative group of nodes which correspond to positive and negative components of the eigenvector. In contrast, the principal eigenvector is associated with either a positive or negative group, not both.

We first consider the case where these non-principal eigenvectors have no zero components.

We will examine one by one the $\boldsymbol{K}$ given eigenvectors. The first eigenvector we examine is the principal eigenvector if it is present in the given set of $\boldsymbol{K}$ eigenvectors, otherwise the first eigenvector is any non-principal eigenvector in the set.

Consider any other eigenvector in the given set, which we call the second eigenvector here. Being orthogonal to each other, the first and second eigenvectors cannot be sign identical nor sign-complementary. Thus the positive and negative groups of the second eigenvector must break at least one group of the first eigenvector into two groups of nodes, which are distinguishable by the sign sequences of the first and second eigenvectors.

We consider yet another eigenvector in the given set, called the third eigenvector. Being orthogonal to each other, the second and third eigenvectors cannot be sign-identical nor sign-complementary. This means that the positive and negative groups of the third eigenvector must break at least one group associated with the second eigenvector. This results in at least three groups of nodes which are distinguishable by the sign sequences of the first, second and third eigenvectors. This procedure can

continue until all the eigenvectors of $\boldsymbol{A^T A}$ have been considered. With **K** eigenvectors considered, we can identify at least $\boldsymbol{K}$ sign-based clusters.

We now consider the case where these K given eigenvectors may have zero components. Suppose that the second eigenvector has some zero components. Then for the other component positions, at least one of the first and second eigenvectors must have both positive and negative components, due to its orthogonality to the first eigenvector. This means that the first and second eigenvectors can already distinguish at least three sign-based clusters rather than two. Consider any other eigenvector in the given set and call it the third eigenvector. Being orthogonal to each other, the second and third eigenvectors cannot be sign-identical nor sign complementary on those component positions for which the second eigenvector has non-zero values. This means that either the third eigenvector has the same sign for these component positions or its group breaks at least one group of the second eigenvector. For the latter case the third eigenvector must assume the opposite sign for either (1) part or (2) all remaining of component positions. For case 1 the third eigenvector has identified an additional sign-based cluster and we are done. In addition the third eigenvector has at least one fewer zero in its components than the second eigenvector. For case 2, the third eigenvector has no zero components, so the next eigenvector must identify a new sign-based cluster. Although in this case the third eigenvector itself doesn't identify an additional sign-based cluster, this is okay since as noted earlier the first and second eigenvectors have already identified at least three sign-based clusters. This argument applies to any newly added eigenvectors in the procedure which ahs zero components, beyond just the second eigenvector.

We demonstrate that sign-based spectral clustering can discover clusters of points in terms of their relationships to the given lines. We first form $\boldsymbol{A^T A}$, which is a $14 - 14$ point-to-point matrix. We can check that rank $(\boldsymbol{A^T A}) = 5$. We then examine the five eigenvectors of $\boldsymbol{A^T A}$ shown in Figure 3.3. We note that the components of eigenvectors exhibit sign sequences (+, +, - , 0, 0), (+, 0, -, 0, 0), (+, -, -, 0, 0), (+, 0, +, +, -), (+, 0, +, -, -) and (+, 0, +, +, +) for points $\{P_1, P_3\}, \{P_2\}$, $\{P_4, P_5\}$, $\{P_6, P_7, P_8\}$, $\{P_9, P_{10}, P_{11}\}$ and $\{P_{12}, P_{13}, P_{14}\}$, respectively. One can check that these

clusters of points identified by the sign sequences indeed correspond to the 5 given lines, and the intersecting point of Line 1 and Line 2. Further, Figure 3.3 shows that as we increase the number of eigenvectors from 2 to 5 the sign sequences will reveal a finer clustering structure.

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14
E.v. 1	0.34	0.55	0.34	0.34	0.34	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
E.v. 2	0.50	0.00	0.50	(0.50)	(0.50)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
E.v. 3	(0.16)	(0.38)	(0.16)	(0.16)	(0.16)	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29	0.29
E.v. 4	0.00	0.00	0.00	0.00	0.00	0.35	0.35	0.35	(0.45)	(0.45)	(0.45)	0.09	0.09	0.09
E.v. 5	0.00	0.00	0.00	0.00	0.00	(0.31)	(0.31)	(0.31)	(0.15)	(0.15)	(0.15)	0.46	0.46	0.46

Line 1 Line 2 Line 4

Line 1&2 Line 3 Line 5

Figure 3.6 The five eigenvectors which has been constructed from figure 3.2

3.4 Relationships between Sign-based and Value-based Clusters

Here we describe some relationships between sign-based clusters and value-based clusters. Note that eigenvector components assuming different signs must assume different values, but the converse may not hold. We have the following from [3].

Lemma 1: A sign-based cluster is either a value-based cluster or a collection of them where nodes share the same sign sequence in the eigenvectors of $\boldsymbol{A^T A}$

Corollary 1: Any K eigenvectors of $\boldsymbol{A^T A}$, for $1 < K \leq rank(\boldsymbol{A^T A})$, can identify at least K sign-based clusters, which in turn can identify at least K collections of value-based clusters.

By Corollary 1, we note that by reading off the component signs of the K eigenvectors, we can find at least K sign based clusters and hence K collections of value-based clusters. However, the given K eigenvectors may specify more value-based clusters than sign-based clusters. We show this in the example in chapter 4.

This means that although it is relatively easy to compute sign-based clusters, they may not reveal all the value-based clusters. We show below in Corollaries 2 and 3 some circumstances where this problem will not occur.

Corollary 2: Suppose that there are no more than K equivalence sets of nodes, for $1 < K \leq rank(\boldsymbol{A^T A})$. Then there are at most K value-based clusters, and sign-based clustering based on any K eigenvectors of $\boldsymbol{A^T A}$, can identify all these value-based clusters.

By Theorem 2, K eigenvectors can identify at least K sign-based clusters. It is often the case that K eigenvectors can identify more than K sign-based clusters. For instance, in the line detection example next section the five eigenvectors of Figure 3.3 identify six sign-based clusters. In this case, we note the following:

Corollary 3: Suppose that K eigenvectors of $\boldsymbol{A^T A}$ can identify S sign-based clusters, and there are E equivalence sets of nodes. Then sign-based clustering based on these K eigenvectors can identify $\min(S, E)$ value-based clusters or their collections.

The line detection example of illustrates Corollary 3. In this example, $K = 5$ eigenvectors in Figure 3.6 can identify $S = 6$ sign-based clusters and all the $E = 6$ value based clusters in Figure 3.6.

Suppose that the number of equivalence sets is not larger than $rank(\boldsymbol{A^T A})$ or the number of sign-based clusters identifiable by the eigenvectors of $\boldsymbol{A^T A}$. Then, by Corollaries 2 and 3, we can compute all value-based clusters via sign-based clustering by just reading off the signs of components in the sign-based clusters.

For the case when the number of equivalence sets is larger than $rank(\boldsymbol{A^T A})$ or the number of identifiable sign-based clusters, sign-based clusters can serve as initial approximations in search for value-based clusters via methods such as k-means.

Chapter 4

Spectral clustering in Distributed Sensor Networks

4.1 Validation of Sensors in a Distributed Setting

In a distributed sensor network deployed in the field, we may find that sensor performance degrades over time due to a number of reasons.

1- Sensors may malfunction or become damaged.
2- Sensors may get blown by the wind to a place where radio reception is poor ,and may low battery power.

This means that we will need to check if sensors still function as expected every so often. In many circumstances it would be impractical to bring calibration instruments to the field and conduct sensor validation procedures on the spot. Fortunately, in a distributed sensor network, there are potentially many sensors in the environment, so they could check each other's validity. In our work, we discuss sign-based spectral clustering approach to validating sensors via their peers in the field. We show that we can identify bad sensors with a high degree of accuracy. After removing these bad sensors from applications, we have a smaller as well as more accurate sensor set for use by applications.

4.2 Overview of the approach

In this work, we consider the problem of clustering sensor nodes in a sensor-target scenario, where there are $\boldsymbol{N}$ sensor nodes taking measurements of $\boldsymbol{M}$ targets. We will classify the sensors in terms of their measurements of the targets. That is, given a $\boldsymbol{M}\times\boldsymbol{N}$ input matrix $\mathbf{A}$ with entries being the sensor to target measurements, we are interested in clustering the $\boldsymbol{N}$ sensor nodes based on $\mathbf{A}$.

We assume the following throughout our work. We are given a set of N sensor nodes and M targets. Let A be the M×N input matrix which captures sensor measurements

of targets. And measurements are non-negative real numbers. Thus A is non-negative, and so is $A^T A$. First, we define some basic concepts and terms. We will use multiple *reference targets*, against which measurements of sensors will be compared. We assume that properly working sensors of similar properties, such as radio and antenna characteristics, and node environments, will report similar measurements against the same targets. We call these working sensors *"good sensors,"* and other sensors which do not report proper measurements *"bad sensors."* Formally, we call sensors of similar properties *"nearby sensors,"* and the associated properties *"sensor indices."* We call the measurements of a sensor on a target the *"weight"* of this sensor target pair. To illustrate this terminology, we consider a simple example, where sensors are indexed by their antenna orientations. In this case, nearby sensors are those which have similar antenna polarizations. Suppose that the received signal strength (RSS) values reported by a sensor on a target correlate with the matching degree of their antenna polarizations. Then nearby sensors, which are in good working condition, are expected to report similar RSS measurements on the same targets. In this technique, sensors are considered to be indexed by their locations rather than antenna angles, so in this case nearby sensors are those in the same general area.

4.2.1 Graph model

We describe a graph model for clustering sensors which will reveal bad sensors.

Let B =(S, N, W) be the weighted bipartite sensor-target graph, with the sensor nodes S on one side, target nodes T on the other side, and the sensor-target edge weights W.

We can represent the graph B by its matrix A, that is, whose entry a_{ij} is the weight between sensor s_j and target t_i.

We expect that nearby sensors will have similar weights at the same targets. Suppose that we have a group of good sensors which have large weights with respect to some targets. Then we expect that this group of sensors will form a cluster in the weighted graph $B_s = (S, W_s)$ whose edge weights W_s are given by the adjacency matrix $A^T A$. On the other hand, by definition, a bad sensor cannot have weights similar to

those of nearby sensors for the same targets and therefore, would not be in the same cluster of nearby sensors.

From the above, it follows that, to identify bad sensors, we need to solve the problem of determining whether a sensor belongs to a cluster of nearby sensors. We assume that there are plentiful sensors so that clusters of nearby sensors will be large. A sensor is deemed to be bad if the sensor satisfies one of the two conditions:

1- the sensor is in a small unique cluster, or

2- the sensor is in a small out-of-place component of a large cluster.

Therefore the essence of the clustering problem is to identify these two conditions in the $\boldsymbol{B_s}$ graph. Next we show that we can achieve this objective by examining the leading eigenvectors of $\boldsymbol{A^T A}$.

4.3 A Model Problem

In the model problem sensors are indexed by their antenna orientations. In this model problem, sensors and reference targets are randomly placed in a region. Each uses an antenna of a certain orientation.

We will solve the sensor clustering problem based on the following two premises:

1. Sensor-target bipartite arrangement, which clusters the best-matched sensors to individual targets. In this model problem, its assumed that the matching of a sensor and a target is based on the degree to which their antenna orientations match. We will identify bad sensors by detecting sensors which do not fall in the cluster of nearby sensors.
2. Use of non-principal eigenvectors, in addition to the principal one, to identify clustering structures in the sensor-target bipartite arrangement.

4.4 Existing Algorithms

The problem we have faced within Distributed Sensor Networks is the identification of the bad sensors in the network. It is hard to identify the bad sensors in the real time as it is difficult to bring the equipment and check which sensors have been changed

according to any of the current device status like reception strength signal(RSS) which may change by the factor like the wind or any other factors. The existing algorithm is based on spectral clustering algorithm. As it is a powerful method to find the principal and non-principal eigenvectors from a sensor-target weight matrix which has non-negative entries, the algorithm uses positive initial vectors. And the results in the principal eigenvector may have aliased values when the underlying graph is a symmetric bipartite graph. By using this techniques and the join approach of principal and non-principal eigenvectors, we have a general method of automatically finding sensor clusters which match individual targets. We use a step function to score the degree of matching in polarization angles, as shown in Figure 4.2. Based on this scoring function, we can assign a matching weight for each sensor-target pair that reflects the antenna polarization difference between the sensor and target. We state the existing algorithm

Algorithm4.1 Spectral clustering in Distributed Sensor Networks

- Construct **A,** the sensor-target weight matrix, that is, given a $m \times n$ input matrix, by finding the difference between their location values of sensor and target or by their antenna orientations. This uses a step scoring function.
- Construct $\boldsymbol{B} = \boldsymbol{A^T A}$ which is a $n \times n$ sensor-to-sensor matrix.
- Compute the eigenvalues$v_1, \dots, v_n$, and eigenvectors $u_1, \dots, u_n$ of $\boldsymbol{B}$ matrix.
- Find the principal eigenvector and non-principal eigenvectors of B matrix.
- Compute the value-based clustering of the leading and non-principal eigenvectors by taking grouping according to the similarity value among of them.
- Compute the sign-based spectral clustering by taking sign sequences such as (-) as zero, and (+) as one. And the number of eigenvectors to use it according to check $rank(\boldsymbol{A^T A}) = n$ Algorithm.

The main idea of this algorithm is to group our data in sets with similar properties and detect the bad sensor by one of the two way sign-value or sign-based clustering.

4.6 Implementation

The algorithm takes the input matrix $m \times n$ by finding the difference between their location values of sensor and target. Based on the $\boldsymbol{A}$ sensor-target weight matrix, the algorithm builds $\boldsymbol{B} = \boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$ a $n \times n$ sensor-to-sensor matrix. Once the sensor-to-sensor matrix has been constructed, we compute the eigenvalues and eigenvectors of B sensor-to-sensor matrix. Now we find the principal eigenvector and non-principal eigenvector where the leading eigenvector carry positive or negative not both, the leading eigenvector can be the first one or second we have to check it.

After finding the principal and non-principal eigenvectors we compute or grouping value-based clustering for the leading eigenvectors and second eigenvector according to the similarity between them. Finally we compute the sign-based spectral clustering algorithm by taken the sign sequences such as (-) as zero, and (+) as one. And the number of eigenvectors to use it in the sign-based according to check rank algorithm $rank(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}) = n$. Where n denotes the number of the eigenvectors to of $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$.

4.6 Illustration

4.7.1 First Example: A sensor-target based on antenna orientation

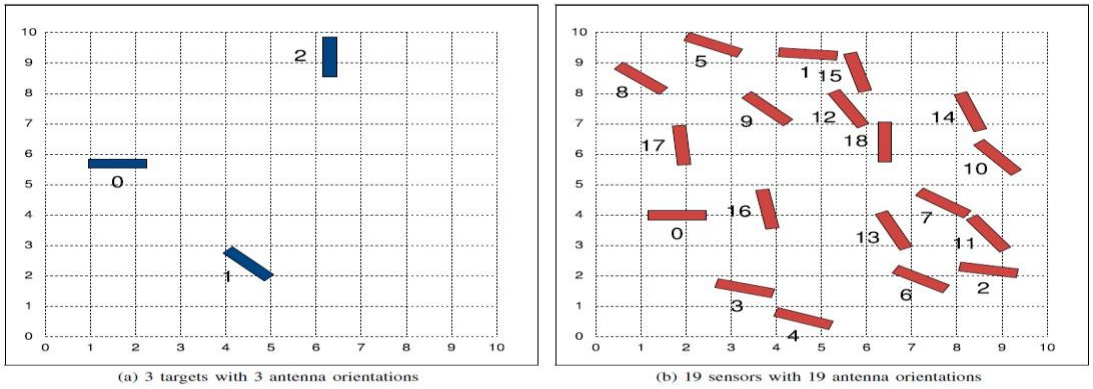

Figure 4.1 Sensors and target in the same region.

First of all we will start with sign-based clustering. We use a step function to score the degree of matching in polarization angles, as shown in Figure 4.2. Based on this scoring function, we can assign a matching weight for each sensor-target pair that reflects the antenna polarization difference between the sensor and target. From the antenna orientations of sensors and targets of Figure 4.1, we derive the sensor-to-target weight matrix **A** in Figure 4.3.

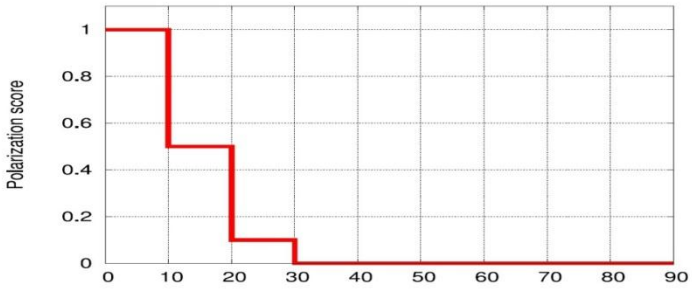

Figure 4.2Step function for scoring polarization matching.

Absolute difference between polarization angles (degrees)

We construct sensor-target weight matrix assume that matching of a target j obtained by sensor i is based on the degree to which their antenna orientations much. We obtain it by using step function as shown in figure 4.1.

$$A = \begin{bmatrix} 1 & 1 & 0.5 & 0.5 & 0.1 & 0.1 & & & & & & & & & & & & \\ & & & & 0.1 & 0.1 & 0.5 & 0.5 & 1 & 1 & 1 & 0.5 & 0.5 & 0.1 & 0.1 & & & \\ & & & & & & & & & & & & & 0.1 & 0.1 & 0.5 & 0.5 & 1 & 1 \end{bmatrix}$$

Figure 4.3 The sensors-target weight matrix

We consider a simple sensor-target example. There are 19 sensors and 3 reference targets randomly placed in a region. With the given sensor-target weight matrix$A$, we will examine some number of leading eigenvectors of $A^T A$ to identify clusters of

sensors best matched to individual targets. The number of leading eigenvectors needed is related to the number of leading eigenvalues of $A^T A$ which are significantly larger than the rest. For example, for the weight matrix $\boldsymbol{A}$ in Figure 4.3, the leading three eigenvalues are 4.04, 2.52 and 2.52, and the rest are zero. Then, we will examine the first and the second eigenvectors of $A^T A$. Now we take the principal and the second eigenvector of $A^T A$ as depicted in figure3.4, and figure 3.5. First, let's take a look at the principal eigenvectors which can be any eigenvector and its sign must be all positive or negative not both. We see the principal eigenvector are all positive . The value of this components, which represent to sensor nodes reflects the connection as well as edge weights of the graph. However there can be aliases problem in this components , we observe in the first eigenvector sensors 0, 1, 17, and 18 all have values 0.00.65 and then the eigenvectors implies that {0,1,17, 18} is one cluster{, 2, 3, 15, 16} all have values 0.00.33 and then we cluster them in one cluster,{4, ,5, 13, 14}is a cluster, {7, 8, 11, 12} is a cluster , and {8, 9, 10}. In the principal eigenvector of figure 4.4, we observe some mistake identified in clusters {2, 3, 15, 16}, and {0, 1, 17, 18}.

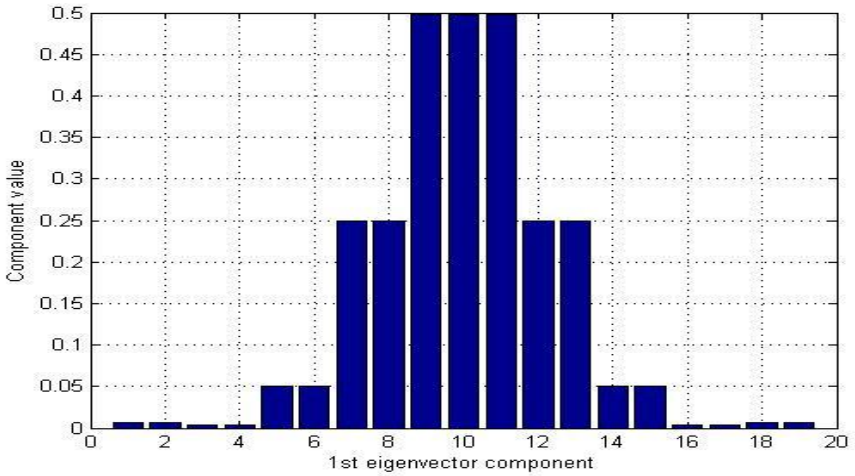

Figure 4.4 The principal eigenvector of $\boldsymbol{A^T A}$

Now, we have a look at the second eigenvector of figure 4.5. Which it has seven distinct values, seven clusters: {0, 1}, {2, 3}, {4, 5}, {6, 7, 8, 9, 10, 11, 12}, {13, 14}, {15, 16}, and {17, 18}.

The second eigenvector has now fixed all the mistaken identified by the first eigenvector and except the mistakenly identifying {6, 7, 8, 9, 10, 11, 12} as one cluster due to aliasing problem.

We see from Figure 4.6 that although all these seven nodes connect to target 1, sensors 8, 9 and 10 connect to target 1 with blue connections, while sensors 6, 7, 11 and 12 with green connections. Thus {8, 9, 10} and {6, 7, 11, 12} should form two clusters. One can check that this partition of {6, 7, 8, 9, 10, 11, 12} into these two clusters is actually implied by the principal eigenvector of Figure 4.4, as the two groups attain two distinct values 0.25 and .5. Thus, for this example, by using both the first and second eigenvectors together, we have obtained the correct eight clusters: {0, 1}, {2, 3}, {4, 5}, {8, 9, 10}, {6, 7, 11, 12}, {13, 14}, {15, 16}, and {17, 18}.

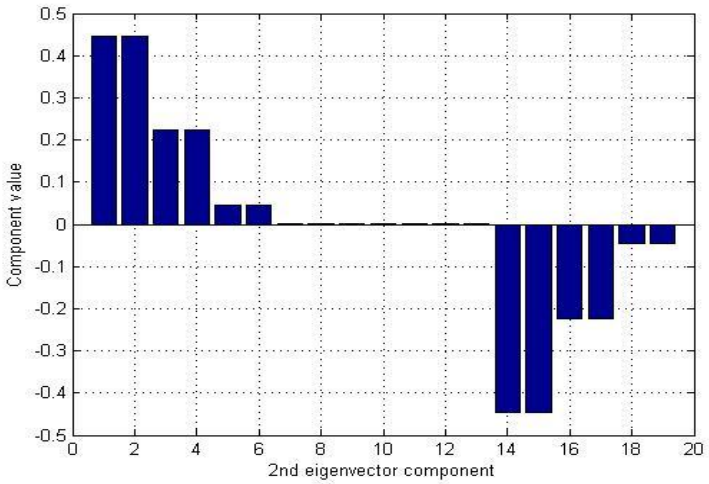

Figure 4.5 The second eigenvector of $A^T A$

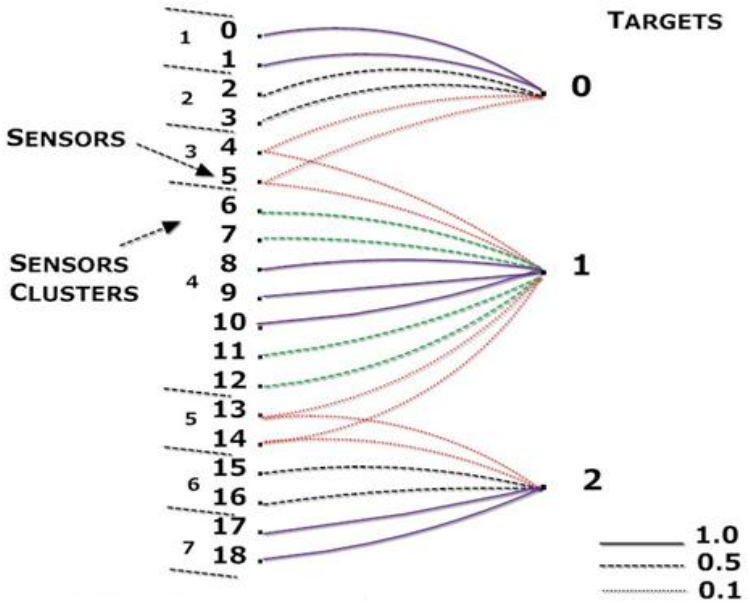

Figure 4.6 Bipartite sensor-target score graph

The seven sensor clusters shown are those found by the second eigenvector of Figure4. 5. Note that with the help of the principal eigenvector of Figure 4, cluster 4 will be correctly partitioned into the two clusters {8, 9, 10} and {6, 7, 11, 12}.

In the figure 4.6 we can see how two eigenvectors are related to polarization matching between sensors and targets. In bipartite sensor-to-target graph of figure 4.6, the color edges represent scores figure 4.2. The color coded scores are blue=1, green = 0.5, and red = 0.1.

In the figure4.6 we have seen sensors 4, and 5 must form a cluster separated from the sensors 13, and 14, since they have a connection to target 0 while latter have a connection to target 2.

Note that in the principal and second eigenvectors can complement each other in other word, they can fix the alias problem of each other.

Second we work with Sign-based spectral clustering where the weight matrix which has been generated and eigenvectors are below.

Weight matrix:

	Sensors																		
Targets	1	1	0.5	0.5	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0.1	0.1	0.5	0.5	1	1	1	0.5	0.5	0.1	0.1	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.1	0.5	0.5	1	1

Eigenvectors:

e_1	0	0	0	0	0	0	0.2	0.2	0.5	0.5	0.5	0.2	0.2	0	0	0	0	0	0
e_2	-0.4	-0.4	-0.2	-0.2	-0	-0	-0	-0	-0	-0	-0	-0	-0	0	0	0.2	0.2	0.4	0.4
e_3	-0.4	-0.4	-0.2	-0.2	-0	-0	0	0	0	0	0	0	0	-0	-0	-0.2	-0.2	-0.4	-0.4

Clustering assignments:

0	0	0	0	0	0	2	2	2	2	2	2	2	1	1	1	1	1	1

Figure 4.7 : Weight matrix A, 3 leading eigenvectors of ATA, and the resulting clustering assignment. Sensors with eigenvector Component signs {+,−, −} are assigned to cluster 0, {+,−,+} to cluster 2, and {+,+, −} to cluster 1.

Suppose that sensor 9 in Figure 4.8 malfunctions, and obtains erroneous measurements on target 0 which are similar to those of sensors 0 and 1. The corresponding bipartite graph is shown in Figure 4.8. The weight matrix **A** reflects this error, as shown in Figure4. 8. That is, with this bad sensor 9, the column in the weight matrix **A** for sensor 9 is now (1, 0, 0) as in Figure 4.8, as opposed to the original column (0, 1, 0) as in Figure 4.6. For this changed **A**, Figure 4.8 shows the three leading eigenvectors of $\mathbf{A}^T\mathbf{A}$, and the resulting sign-based clustering assignments of sensors using these leading eigenvectors. We see that now sensor 9 is clustered in a cluster labeled as 0 while its nearby sensors belong to a large cluster labeled as 2. Thus, sensor 9 is in a small out-of-place component of a large cluster, i.e., it satisfies condition C2 in Section II. This means that our spectral clustering method has correctly identified sensor 9 as a bad sensor.

Weight matrix:

	Sensors																		
Targets	1	1	0.5	0.5	0.1	0.1	0	0	0	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0.1	0.1	0.5	0.5	1	0	1	0.5	0.5	0.1	0.1	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.1	0.5	0.5	1	1

Eigenvectors:

e_1:	0.5	0.5	0.3	0.3	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0
e_2:	-0	-0	-0	-0	0	0	0.3	0.3	0.6	-0	0.6	0.3	0.3	0	0	0	0	0	0
e_3:	-0	-0	-0	-0	0	0	0	0	0	-0	0	0	0	-0	-0	-0.3	-0.3	-0.6	-0.6

Cluster assignments:

0	0	0	0	2	2	2	2	2	0	2	2	2	1	1	1	1	1	1

Figure 4.8 A weight matrix obtained by introducing an error into A at sensor 9 on target 0, the resulting 3 leading eigenvectors, and the new clustering assignments. The cluster label of the erroneous sensor is highlighted with a box.

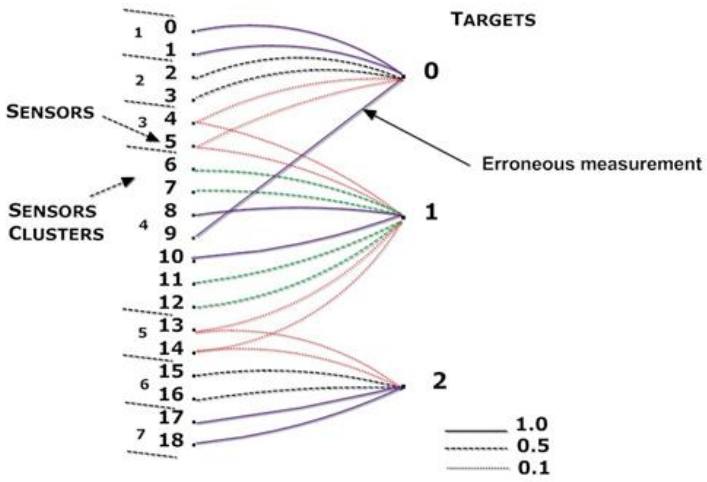

Figure 4.9 Bipartite sensor-target score graph

4.7.2 Second Example: A sensor-target based on the locations

There are 28 sensors and 4 targets place on a line, numbered 0-27 and 0-3, respectively. The line has 136 uniformly spaced grid points, label as $0, 1, \ldots, 135$. Sensor i and target j are located at grid points $5i$ and 45j , respectively.

We model the measurement of target j obtained by sensor i as a function of the difference between their location values. We use step function shown in figure 4.2. For example, sensor 4 is at location 20 and target 2 is at location 90; therefore, their location differs by 70. This means that according to the measurement function in figure 4.2, sensor 4's is measurement value at target 2 is 0.

$$A = \begin{bmatrix} 1 & 1 & .5 & .5 & 0.1 & 0.1 & & & & & & & & & & & & & & & & & & & & & & \\ & & & & 0.1 & 0.1 & .5 & .5 & 1 & 1 & 1 & .5 & 0.1 & 0.1 & & & & & & & & & & & & & & \\ & & & & & & & & & & & & 0.1 & 0.1 & .5 & .5 & 1 & 1 & 1 & .5 & .5 & 0.1 & 0.1 & & & & \\ & & & & & & & & & & & & & & & & & & & & & & & 0.1 & 0.1 & .5 & .5 & 1 & 1 \end{bmatrix}$$

Figure 4.10 The sensor-target weight matrix corresponding to figure 4.10

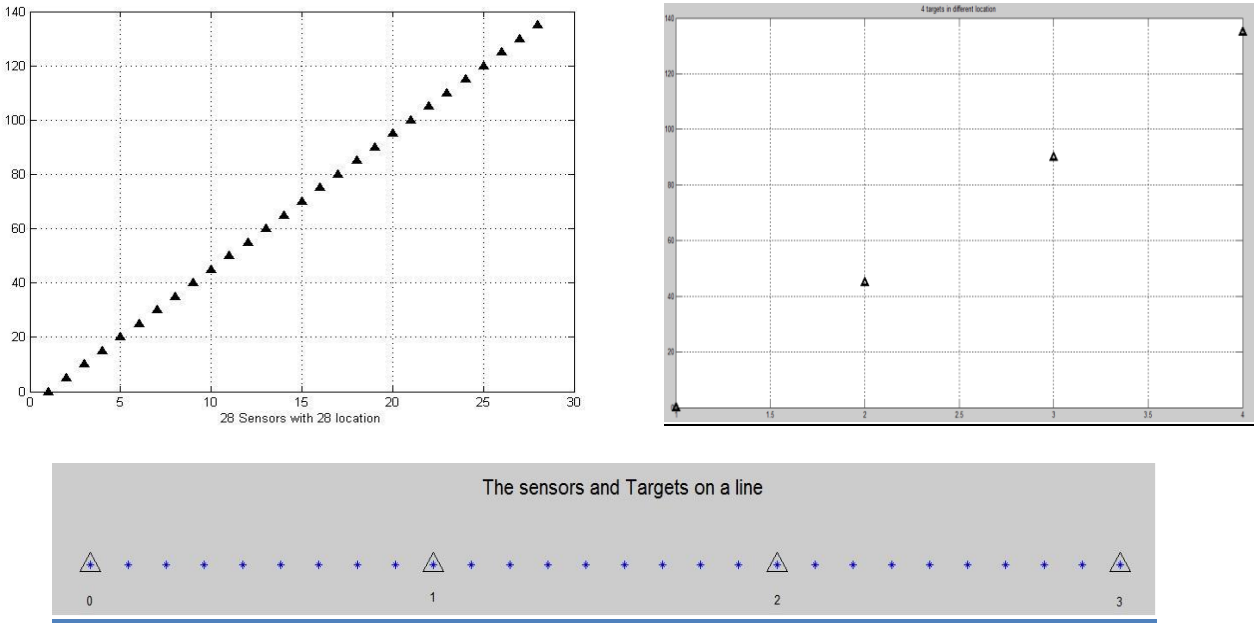

Figure 4.11 20 Sensors on left side, and 4 targets right side in different locations ,the sensors and targets on a line below them

First of all, we construct **A** sensor-target weight matrix 4×28 , by finding the difference between their location values of sensor and target measurements, on one line. Based on **A** sensor-target weight matrix, the algorithm builds a 28×28 sensor-sensor matrix then we will examine some number of leading eigenvectors of $\boldsymbol{A^T A}$ **to** identify clusters of sensors.

The number of leading eigenvectors needed is related to the number of leading eigenvalues of $\boldsymbol{A^T A}$ which are significantly larger than the rest. For example for the weight matrix $\boldsymbol{A}$ in figure4.10, the leading four eigenvalues are 4.06, 4.02, 2.52, 2.52 and the rest are zero.

Therefore, we will examine the first and the second eigenvectors of $\boldsymbol{A^T A}$.

Once $\boldsymbol{A^T A}$ a $m \times n$ sensor-sensor matrix has been constructed, the algorithm computes its eigenvalues and eigenvectors of matrix$\boldsymbol{B}$**.** After the computation of eigenvalues and eigenvectors, the algorithm find the principal and second eigenvectors of $\boldsymbol{A^T A}$, and we check that $rank(\boldsymbol{A^T A}) = 4$. Thus $\boldsymbol{A^T A}$ has four eigenvectors. Based on the principal and non-principal eigenvectors compute the value-based clustering by taken grouping according to the similarity values, at first we take a look at the principal eigenvector of figure 4.11:

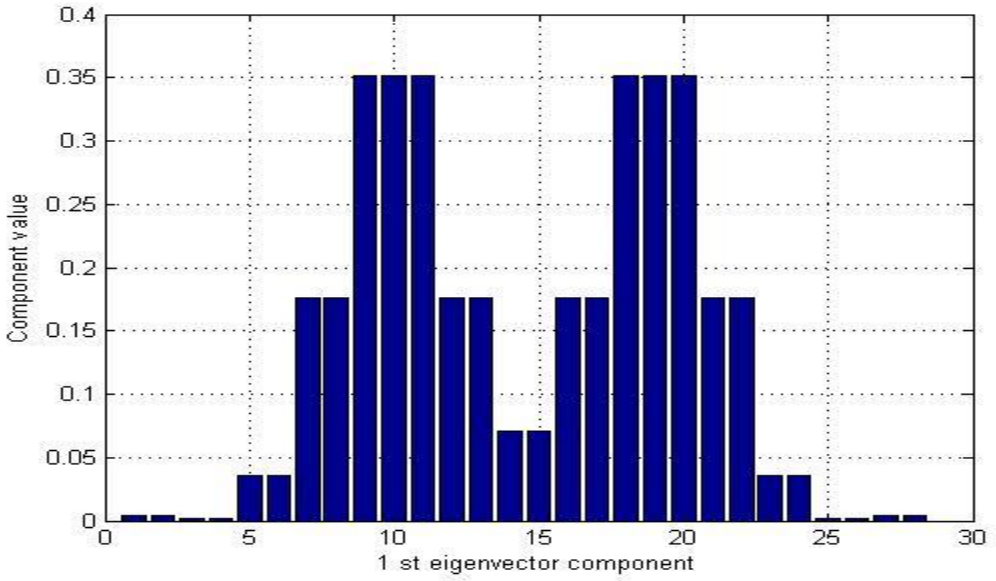

Figure 4.12 The principal eigenvector of $\mathbf{A^T A}$, with values of its components which is corresponding to sensor nodes

In the first principal eigenvector, we note that in the Perron-Ferobenius theorem the principal eigenvector corresponding to the largest eigenvalue must have all of its components being of the same sign that is, they are either all positive or all negative. Any other eigenvector must have components with different signs, due to its orthogonality to the principal eigenvector. Thus, every non-principal eigenvector is associated with a positive and also a negative group of nodes which correspond to positive and negative components of the eigenvector. In contrast, all the components of the principal eigenvector are all positive or negative group, not both.

In the figure 4.11. The values of these components, which correspond to sensor nodes, reflect connectivity as well as edge weights of the graph. For example sensors 1, 2, 27, and 28 all have values 0.0045, and the eigenvector implies that {1,2, 27,28} is a cluster, {3, 4, 25, 26} is a cluster, {5, 6, 23, 24} is a cluster, {7, 8, 12, 13 ,16, 21, 22} is a cluster, {9, 10, 11, 18, 19, 20} is a cluster, and {14, 15}, in the figure 4.11. We identify six clusters only.

The first principal eigenvector of figure 4.11 mistakenly identifies clusters{1, 2, 27, 28}, {3, 4, 25, 26}, {5, 6, 23, 24}, {7, 8, 12, 13, 16, 17, 21, 22}, and {9,10, 11,18, 19, 20}.

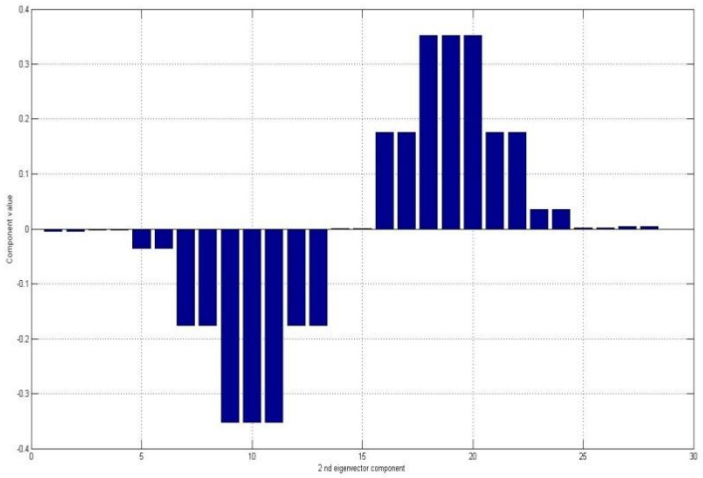

Figure 4.13 The second eigenvector of $\mathbf{A^T A}$, with values of its components which is corresponding to sensor nodes

Now, we take a look at the second eigenvector of figure 4.12. It has eleven distinct values. Which imply eleven clusters:{1, 2},{3, 4}{5, 6},{7, 8, 12, 13},{9, 10 ,11},{14,15},{16, 17, 21, 22},{18, 19, 20},{23, 24},{25, 26}, and {27, 28} .

One can check that the second eigenvector has fixed all the mistaken clusters identified by the first eigenvector. Finally, compute the sign-based spectral clustering by taken the sign sequences such as (-) as zero, and (+) as one.

Weight matrix: Sensors

1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	0.5000	0.5000	0.1000	0.1000	0	0	0	0	0	0	0	0
0	0	0	0	0.1000	0.1000	0.5000	0.5000	1	1	1	0.5000	0.5000	0.1000
0	0	0	0	0	0	0	0	0	0	0	0	0	0.1000
0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1000	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1000	0.5000	0.5000	1	1	1	0.5000	0.5000	0.1000	0.1000	0	0	0	0
0	0	0	0	0	0	0	0	0.1000	0.1000	0.5000	0.5000	1	1

Eigenvectors:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.0046	0.0046	0.0023	0.0023	0.0355	0.0355	0.1754	0.1754	0.3509	0.3509	0.3509	0.1754	0.1754	0.0702
-0.0047	-0.0047	-0.0024	-0.0024	-0.0357	-0.0357	-0.1763	-0.1763	-0.3526	-0.3526	-0.3526	-0.1763	-0.1763	1.4155e-15
0.4454	0.4454	0.2227	0.2227	0.0440	0.0440	-0.0029	-0.0029	-0.0058	-0.0058	-0.0058	-0.0029	-0.0029	-0.0012
0.4454	0.4454	0.2227	0.2227	0.0439	0.0439	-0.0030	-0.0030	-0.0059	-0.0059	-0.0059	-0.0030	-0.0030	5.2483e-14
0	0	0	0	0	0	1	1	1	1	1	1	1	2

15	16	17	18	19	20	21	22	23	24	25	26	27	28
0.0702	0.1754	0.1754	0.3509	0.3509	0.3509	0.1754	0.1754	0.0355	0.0355	0.0023	0.0023	0.0046	0.0046
1.4155e-15	0.1763	0.1763	0.3526	0.3526	0.3526	0.1763	0.1763	0.0357	0.0357	0.0024	0.0024	0.0047	0.0047
-0.0012	-0.0029	-0.0029	-0.0058	-0.0058	-0.0058	-0.0029	-0.0029	0.0440	0.0440	0.2227	0.2227	0.4454	0.4454
5.2483e-14	0.0030	0.0030	0.0059	0.0059	0.0059	0.0030	0.0030	-0.0439	-0.0439	-0.2227	-0.2227	-0.4454	-0.4454
2	2	2	2	2	2	2	2	4	4	4	4	4	4

Figure 4.14 Weight matrix A, 4 leading eigenvectors of $\mathbf{A^T A}$, and the resulting clustering assignment. Sensors with eigenvector component signs {+,-,+,+} are assign to cluster 0, {+,-,-,-} to cluster 1, {+,+,-,+} to cluster 2, and {+,+,+,-} to cluster 4.

The components of the eigenvectors in figure 6. Show the sign sequences {+, -, +, +}, {+, -, -, -}, {+, +, -, +}, and {+, +, +, -} for sensors 0 through 56-12, 13-21, and 22-28 respectively. Therefor the four clusters of sensors identified by the four sign sequences indeed corresponding to the depicted in figure 4.14.

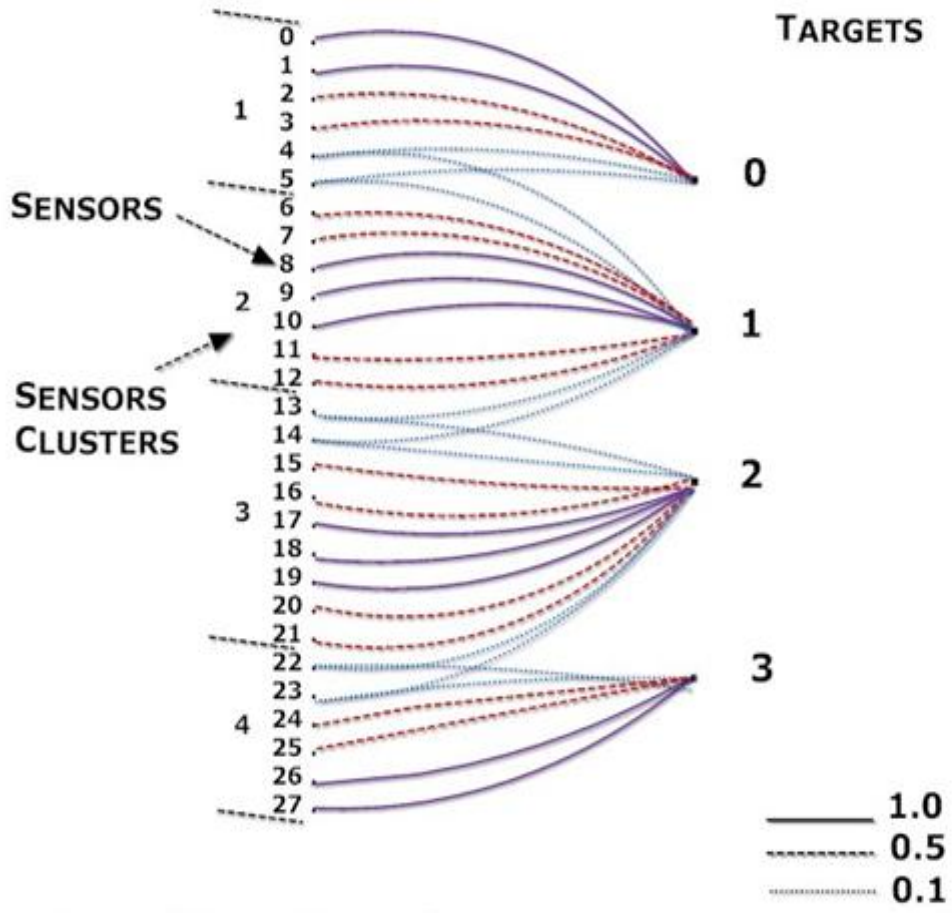

Figure 4.15 Three sensor clusters sensors 0 through 5, 6–12, 13-21, and 22 through 27-corresponding to the four targets.

Suppose that sensor 7 in figure 4.16 malfunctions and gets erroneous measurement on target 2 which are similar to those of sensors 15, 16, 20, and 21. The weight matrix *A* reflects the error , as shown in figure 8. With this bad sensor 7 , the column in the weight matrix *A* for sensor 7 is now (0, 0, 0, .5, 0) as in figure 8, which is opposed to the original column (0, 0.5, 0, 0).

Weight matrix: Sensors

1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	0.5000	0.5000	0.1000	0.1000	0	0	0	0	0	0	0	0
0	0	0	0	0.1000	0.1000	0	0.5000	1	1	1	0.5000	0.5000	0.1000
0	0	0	0	0	0	0.5000	0	0	0	0	0	0	0.1000
0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1000	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1000	0.5000	0.5000	1	1	1	0.5000	0.5000	0.1000	0.1000	0	0	0	0
0	0	0	0	0	0	0	0	0.1000	0.1000	0.5000	0.5000	1	1

Eigenvectors:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
2.1753e-04	2.1753e-04	1.0876e-04	1.0876e-04	0.0019	0.0019	0.2412	0.0096	0.0193	0.0193	0.0193	0.0096	0.0096	0.0502
-0.0081	-0.0081	-0.0040	-0.0040	-0.0521	-0.0521	0.0102	-0.2566	-0.5132	-0.5132	-0.5132	-0.2566	-0.2566	-0.0493
0.0251	0.0251	0.0126	0.0126	0.0025	0.0025	-0.0036	-1.4193e-04	-2.8387e-04	-2.8387e-04	-2.8387e-04	-1.4193e-04	-1.4193e-04	-7.3919e-04
0.6294	0.6294	0.3147	0.3147	0.0619	0.0619	1.9800e-04	-0.0050	-0.0099	-0.0099	-0.0099	-0.0050	-0.0050	-9.5196e-04
0	0	0	0	0	0	2	1	1	1	1	1	1	1

15	16	17	18	19	20	21	22	23	24	25	26	27	28
0.0502	0.2412	0.2412	0.4823	0.4823	0.4823	0.2412	0.2412	0.0488	0.0488	0.0027	0.0027	0.0054	0.0054
-0.0493	0.0102	0.0102	0.0205	0.0205	0.0205	0.0102	0.0102	0.0021	0.0021	1.6145e-04	1.6145e-04	3.2290e-04	3.2290e-04
-7.3919e-04	-0.0036	-0.0036	-0.0071	-0.0071	-0.0071	-0.0036	-0.0036	0.0622	0.0622	0.3147	0.3147	0.6294	0.6294
-9.5196e-04	1.9800e-04	1.9800e-04	3.9599e-04	3.9599e-04	3.9599e-04	1.9800e-04	1.9800e-04	-0.0025	-0.0025	-0.0126	-0.0126	-0.0251	-0.0251
1	2	2	2	2	2	2	2	4	4	4	4	4	4

Figure 4.16 Weight matrix A, 4 leading eigenvectors of $\mathbf{A^T A}$, and the resulting clustering assignment. Sensors with eigenvector component signs {+,-,+,+} are assign to cluster 0, {+,-,-,-} to cluster 1, {+,+,-,+} to cluster 2, and {+,+,+,-} to cluster 4.

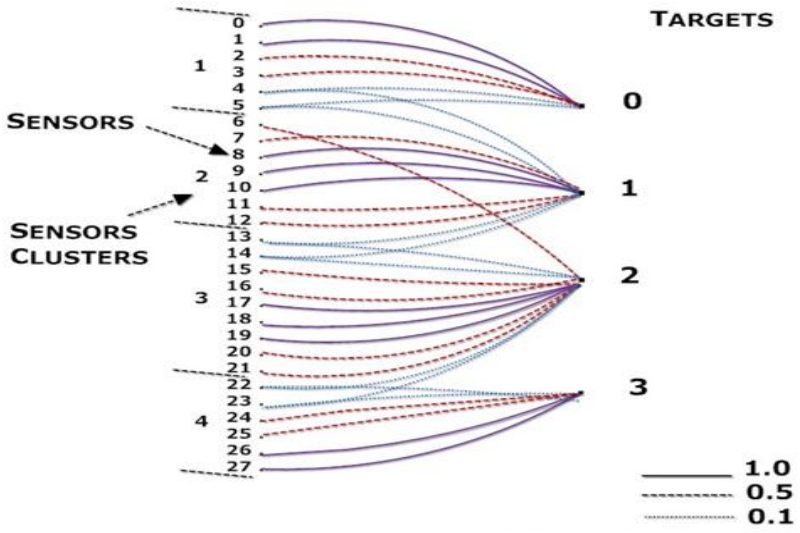

Figure 11 Three sensor clusters sensors 0 through 5, 6–12, 13-21, and 22 through 27 corresponding to the four targets and sensor 7 belong to target 2.

4.8 Proposed Scheme based on Euclidean distance

Algorithm 4.2 Spectral clustering in Distributed Sensor Networks

- Construct **A,** the sensor-target weight matrix, that is, given a $m \times n$ input matrix, by finding the difference between their location values of sensor and target using Euclidean distance formula.
- Construct $\boldsymbol{B} = \boldsymbol{A^T A}$ which is a $n \times n$ sensor-to-sensor matrix.
- Compute the eigenvalues$v_1, \dots, v_n$, and eigenvectors $u_1, \dots, u_n$ of $\boldsymbol{B}$ matrix.
- Find the principal eigenvector and non-principals eigenvectors of B sensor-to-sensor matrix.
- Compute the value-based clustering of the leading and non-principal eigenvectors by taken grouping according to the similarity value among of them.
- Compute the sign-based spectral clustering by taken sign sequences such as (-) as zero, and (+) as one. And the number of eigenvectors to use it according to check $rank(\boldsymbol{A^T A}) = n$ Algorithm.

The authors [3] developed novel mathematical models and computer algorithms which are expected to be useful for future wireless computing, sensing and communications systems. Based on two cases, the first case concerns a new spectral clustering method which is able to perform grouping by examining just the signs in leading eigenvectors of the input data. The method greatly simplifies spectral clustering, while improving the speed and robustness of the clustering process. The second case concerns a spectral-based method for validating sensor nodes via clustering of sensors based on their measurement data. In many circumstances, it would be impractical to bring calibration instruments to the field for conducting sensor validation procedures on the spot. With our peer validation methods, sensors can check each other's validity in the field.

We have specified conditions in identifying bad sensors. A sensor is deemed to be bad if the sensor satisfies one of the two conditions: (C1) the sensor is in a small

unique measurement cluster or (C2) the sensor is in a small out-of-place component of a large measurement cluster.

Below we give an example of our proposed scheme based on Euclidean dsstance.

4.8.1 Example: A Sensor-Target based on Proposed Scheme of Euclidean distance

To illustrate value-based, and sign-based based on proposed scheme, we consider a simple sensor-target example. There are 20 sensors and 3 targets place on a line, numbered 0-19 and 0-3.The line has 100 spaced grid points, labeled as $0, 1, \ldots, 100$. Sensor i and target j are located at different grid points .

We model the measurement of target j obtained by sensor i as a function of the difference between their location values by using Euclidean distance to find the distance between the targets and sensors. We use step function shown in figure 4.2.

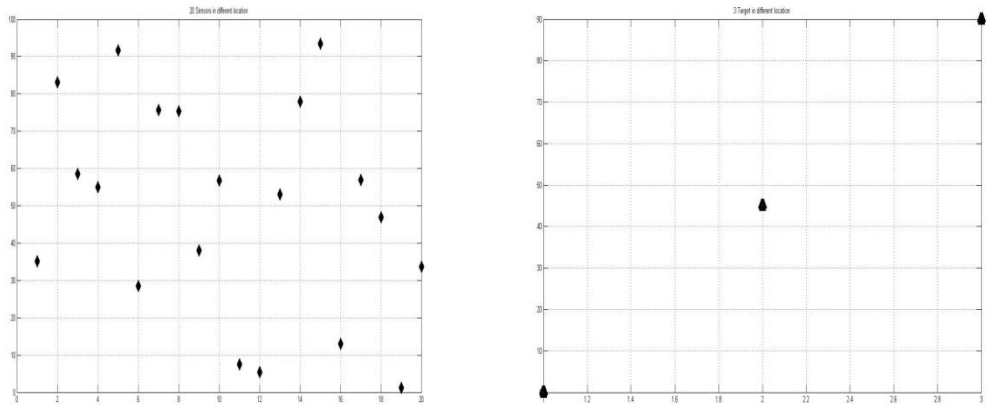

Figure 4.18: 20 Sensors in different locations in left side, and 3 targets in different locations in right side

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0.1000	0	0	0	0	1	1	0	0	0	0.5000	0	0	1	0
1	0	0.5000	1	0	0.5000	0	0	1	0.5000	0	0	1	0	0	0	0.5000	1	0	0.5000
0	1	0	0	1	0	0.5000	0.5000	0	0	0	0	0	0.5000	1	0	0	0	0	0

Figure 4.19 The sensor-target Weight matrix A

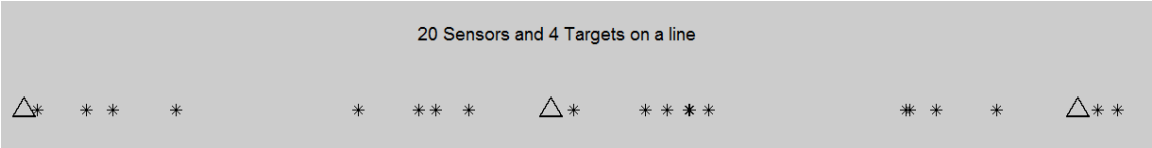

Figure 4.20 Sensors and targets are they on a line in different the location

First of all, we construct **A** sensor-target weight matrix 3×20 , by Find the difference between their location values of sensor and target measurements, on one a line. Based on **A** sensor-target weight matrix, the algorithm builds a 20×20 sensor-sensor matrix then we will examine some number of leading eigenvectors of $\boldsymbol{A^T A}$ **to** identify clusters of sensors.

The number of leading eigenvectors needed is related to the number of leading eigenvalues of $\boldsymbol{A^T A}$ which are significantly larger than the rest. For example for the weight matrix $\boldsymbol{A}$ in figure4.18, the leading three eigenvalues are 6.25, 3.75, 3.75 and the rest are zero.

Therefore, we will examine the first and the second eigenvectors of $\boldsymbol{A^T A}$.

Once $\boldsymbol{A^T A}$ a $m \times n$ sensor-sensor matrix has been constructed, the algorithm computes its eigenvalues and eigenvectors of matrix $\boldsymbol{B}$**.** After the computation of eigenvalues and eigenvectors, the algorithm find the principal and second eigenvectors of $\boldsymbol{A^T A}$, and we check that $rank(\boldsymbol{A^T A}) = 3$. Thus $\boldsymbol{A^T A}$ has four eigenvectors. Based on the principal and non-principal eigenvectors compute the value-based clustering by taken grouping according to the similarity values, at first we take a look at the principal eigenvector of figure 4.21:

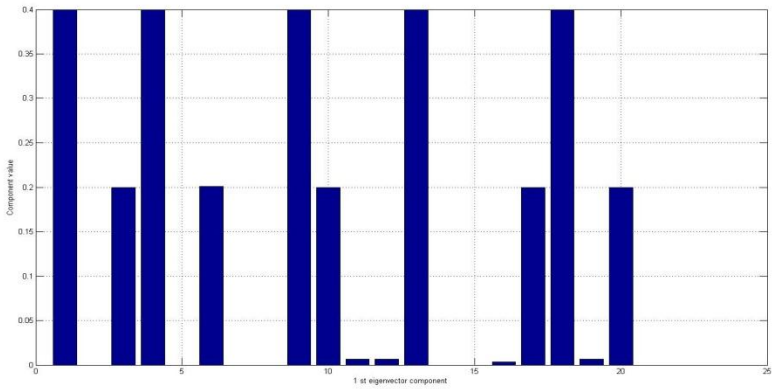

Figure 4.21 1st eigenvector of $\mathbf{A^T A}$

In the figure 4.21. The values of these components, which correspond to sensor nodes, reflect connectivity as well as edge weights of the graph. And the eigenvector implies that {2,5,7,8,14,15} is a cluster, {16} is a cluster, {11,12,19} is a cluster, {20,3,10,17} is a cluster, {6} is a cluster, and {1,4,9,13,18}, in the figure 4.21. We identify six clusters only.

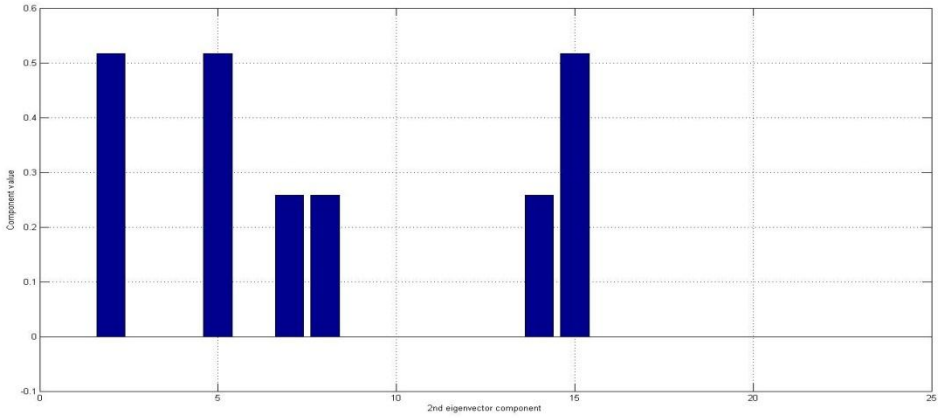

Figure 4.22 2nd eigenvector of $A^T A$

Now, we take a look at the second eigenvector of figure 4.21. It has eleven distinct values. Which imply sex clusters they are as

{ [7,8,14], [2,5,15], [20,3,10,17], [6], [1,4,9,13,18]}.

We observe from the first leading eigenvector and non-principal eigenvectors the number of the cluster is equal in the sign-value clustering.

Now we have to look at the sign-based spectral clustering.

0.399917	0	0.199959	0.399917	0	0.200627	0	0	0.399917	0.199959	0.006686	0.006686	0.399917	0	0	0.003343	0.199959	0.399917	0.006686	0.199959
4.04E-17	0.516398	7.40E-17	4.04E-17	0.516398	-7.60E-17	0.258199	0.258199	2.35E-17	-2.03E-17	7.10E-18	-1.67E-17	-6.91E-17	0.258199	0.516398	-7.07E-18	-5.43E-17	-6.82E-18	1.45E-17	-3.49E-17
0.009259	0	0.004629	0.009259	0	-0.05075	0	0	0.009259	0.004629	-0.55384	-0.55384	0.009259	0	0	-0.27692	0.004629	0.009259	-0.55384	0.004629
7	7	7	7	7	4	7	7	7	5	6	4	5	7	7	4	5	5	6	5

Figure 4.23: Weight matrix A, 3 leading eigenvectors of $\boldsymbol{A^T A}$, and the resulting clustering assignment. Sensors with eigenvector Component signs {+,+, +} are assigned to cluster 7, {+,−,−} to cluster 4, {+, −,+} to cluster 5, and {+,+,−} to cluster 6.

Now we illustrate how our spectral clustering method can identify a bad sensor.

Suppose that sensor 6 in figure 4.23 malfunctions and gets erroneous measurement

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0.5000	0	0	1	0
2	1	0	0.5000	1	0	0.5000	0	0	1	0.5000	0	0	1	0	0	0	0.5000	1	0	0.5000
3	0	1	0	0	1	0.1000	0.5000	0.5000	0	0	0	0	0	0.5000	1	0	0	0	0	0

Figure 2.24 The sensor-target Weight matrix A

0.39989	0.00803	0.19994	0.39989	0.00803	0.20075	0.00401	0.00401	0.39989	0.19994	8.73E-17	8.73E-17	0.39989	0.00401	0.00803	4.36E-17	0.19994	0.39989	3.49E-16	0.19994
0.01035	-0.5157	0.00518	0.01035	-0.5157	-0.0464	-0.2578	-0.2578	0.01035	0.00518	1.78E-16	1.78E-16	0.01035	-0.2578	-0.5157	8.92E-17	0.00518	0.01035	2.80E-16	0.00518
1.41E-16	-1.62E-16	7.03E-17	1.41E-16	-1.62E-16	1.39E-17	-8.48E-17	-8.48E-17	1.07E-16	5.33E-17	-0.5547	-0.5547	1.00E-16	-1.04E-16	-1.73E-16	-0.2774	-1.07E-16	3.44E-17	-0.5547	1.20E-16
7	4	7	7	4	5	4	4	7	7	6	6	7	4	4	6	6	7	6	7

Figure 4.25 Weight matrix A, 3 leading eigenvectors of A^TA, and the resulting clustering assignment. Sensors with eigenvector Component signs {+,+, +} are assigned to cluster 7, {+,−,−} to cluster 4, {+, −,+} to cluster 5, and {+,+,−} to cluster 6.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

1. We studied and implemented of some classical clustering algorithms like k-means and its variants.
2. We discussed the spectral clustering algorithms. We implemented all three aproches used for constructing the similarity matrix used in spectral clustering.
3. We studied two variants of spectral clustering namely Value-based and sign-based methods. We implemented these methods and illustrated with line detection problem.
4. An interesting application of sign-based clustering in validating sensors through their peers in a distributed setting is also reported along with the implementation. We saw that the similarity matrix in this application is constructed based on polarization of angles of the sensors or one-dimensional spatial distance. We introduced Euclidean distance based methods and used it along with sign-based clustering.

5.1 Future Work

Sign-based clustering is especially useful when it can identify a relatively large number of clusters close to the target numbers of clusters. In this case sign-based method can find all value-based clusters. This has to be experimentally verified with large number of sensor-targets. We discussed three types of construction of the similarity graph. It would be interesting to study which type is better for a target

application. Another direction of future work is to see the application of spectral clustering in speech separation, in particular segmentation of the spectrogram.

Reference

[1] Ulrike, Von Luxbrug. "*Tutorial Spectral Clustering*",Statistics and Computing, Vol.17, No.4.:1-32 (1 December 2007). www.springer.com.

[2] H. T. Kung and D. Vlah. "*A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks*", International Journal of Sensor Networks archive Volume 8 Issue 3/4, October 2010.

[3] H. T. Kung and D. Vlah. "*Sign-based spectral clustering*", 25th Biennial Symposium, ieeexplore.

[4] B. Matthias and S, Juri"*Spectral clustering*", Data Warehousing and Data Mining January 23, 2009.

[5] A´ ngela Ferna´ndez Pascual. "*Advanced methods for dimensionality reduction and lustering: Laplacian Eigenmaps and Spectral Clustering*" December 13, 2010.

[6] C. Elkan " *Using the trangle inequality to accelerate k-means*" in procedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

[7] J. Han and M. Kamber, "*Data Mining Concepts and Techniques*" Second Edition.

[8] K-medoids at http://en.wikipedia.org/wiki/K-medoids

Appendix: Mathematical Results

Graph notation

1 is the unity vector, i.e., 1 = (1, . . . , 1).

$G = (S, E)$, is the similarity graph, formed by the nodes V and the edges E.

$s = \{x_1, \dots, x_N\}$ is our set of nodes formed by the points of the dataset in the original n dimensional space.

$w = (w_{ij})$ With $w_{ij} > 0$, is the adjacency matrix of the graph.

We work with undirected graphs, so $w_{ij} = w_{ji}$.

D is the degree matrix, defined as the diagonal matrix with the degrees$d_i = \sum_{j=1}^{N} w_{ij}$.

elements.

$\overline{A} = S/A$ where $A \subset V$ is a subset of the vertices set. We define the indicator vector

$$1_A = (f_1, \dots, f_m)'$$

$$\text{Where } f_i = \begin{cases} 1 & if\ x_i \in A \\ 0 & if\ x_i \in a \end{cases}$$

To simplify notation, we use the compact expression$\{i | x_i \in A\}$.

$|A|$ represents the number of nodes in A. It is a measure based on the number of vertices of our
graph.

$Vol(A) = \sum_{i \in A} d_i$ represents the size of A. It is a measure based on the weights of the edges of
our graph.

We say that our subset A is **connected** if any two vertices in A are connected and all the intermediate points belong to A.

A is a connected component if it is connected and it no has connections with $\overline{A}$.

The sets $A_1, \dots, A_m$ form a partition of the graph if $A_i \cap A_j = \emptyset$ and $A_i \cup \dots \cup A_k = V$ where, k is the dimension of the reduced space.

Constructing similarity graphs

To construct similarity graphs from a given set$\{x_1, \dots, x_N\}$, we establish the nodes as the points of the set. To create the links between nodes we must determine the adjacency between them and the weights of these edges. The goal we want to reach by the construction of a similarity graph is to model the local neighborhood relationships between the data points.

- **The $\boldsymbol{\varepsilon}$-neighborhood graphs**: We connect all points with distance smaller than $\boldsymbol{\varepsilon}$

  $$|x_1, \dots, x_N|^2 \ < \ \boldsymbol{\varepsilon}.$$

  When the distances between all connected points are of the same scale the weights will not incorporate more information, so we usually construct no weighted graphs. The principal advantage of this method is that it is geometrically motivated, as it is the natural measure between points. But in graphs very connected is more difficult to choose an appropriate$\boldsymbol{\varepsilon}$**.**
- **The *k*-nearest neighbor graphs:** We connect x_i with its *k* nearest neighbors. If we construct our graph in this way, i.e,$(x_i, x_j) \in E \ if \ x_i \in K$ nearest neighbors of x_i, we obtain a symmetric graph. If we prefer to work with an asymmetric graph, we could modify slightly this method in one of the following ways:
  - Option 1: The usually called k-nearest neighbors graph is constructed without considering the direction, i.e., $(x_i, x_j) \in E$

$if\ x_i \in K$ K nearest neighbors of x_i, or $x_i \in K$ nearest neighbors of x_j.

- Option 2: The graph called mutual *k-nearest neighbors graph* is constructed connecting $x_i\ and\ x_j$ only if $if\ x_j \in K$ K nearest neighbors of x_i and $x_i \in K$ nearest neighbors of x_j.

We weight the edges by the similarity between adjacent points. This relation is given by the similarity matrix S, The advantage of this method is that the neighbors of each point are easier to choose and the resulting graph is always connected.

- **The fully connected graphs:** We connect each point with all the other nodes of our graph. In this case, we weight the connected nodes with S_{ij} for each edge. We only use the similarity function with this purpose if it contributes with local information. An example of this kind of functions is the exponential decay function:

$$\boldsymbol{s}(x_i, \quad x_j) = \boldsymbol{exp}(-|x_i - x_j|^{\boldsymbol{2}}/(\boldsymbol{2\sigma^2})$$

Where σ controls the change of our neighborhood.

Laplacian Graphs

Laplacian Graphs are one of the main tools for Laplacian eigenmaps and Spectral Clustering. In this appendix we are going to explain the different types of Laplacian Graphs and their most important properties For this purpose, we assume that our graph G is an undirected weighted graph and the weighted matrix W has positive entries ($w_{ij} = w_{ji} \geq 0$). We classify the Laplacian Graphs in unnormalized and normalized Laplacian graph.

- **Unnormalized Laplacian Graph**

  We define an Unnormalized Laplacian Graph L as follows

$$\boldsymbol{L = D - W}.$$

  This kind of graphs are characterized by a series of properties describe in the

Following proposition.

Proposition 1 of the matrix L satisfies the following properties:

1. For ever vector $f \in R^n$ we have
$$\overline{f}Lf = \frac{1}{2}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2. \quad (1.1)$$
2. L is always symmetric and positive semi-definite ($\forall i, \lambda_1 \geq 0$)
3. The smallest eigenvalue of *L* is called the spectral gap.
4. The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector 1.
5. The multiplicity of 0 as an eigenvalue of *L* is the number of connected components of graph.

Proof We prove each property separately.

1. We proof that the expression1.1

$$\overline{f}Lf = \overline{f}Df - \overline{f}Wf$$
$$= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$
$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{i=1}^{n} d_i f_i^2\right)$$
$$= \frac{1}{2}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2.$$

- **Normalized Laplacian Graph**

  We are going to study two types of Normalized Laplacian Graphs. The first one, the Symmetric Laplacian is given by the expression

$$L_{Sym} = D^{-1/2}D^{-1/2} = I - D^{-1/2}WD^{-1/2}$$

L_{sym} has the property of being a symmetric matrix.

The second one is the called RandomWalk Laplacian, as it is closely related with random walks.

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