University of California, Berkeley Mathematics Department

Bachelor Honors Thesis

Spectral Clustering and Its Application to Image Segmentation

Student

Tram Nghi Pham, *University of California, Berkeley*

tram.pham18@berkeley.edu

Thesis Supervisor

Beresford Parlett, *University of California, Berkeley*

parlett@math.berkeley.edu

Date: Oct 15, 2017

Abstract

Clustering is the task of grouping a set of objects that share similar patterns into the same classes. Today, the clustering method is the most widely used approach for exploratory data analysis; hence, many data clustering algorithms have been extensively developed. Although the human eye can instantly recognize clusters in already distinct data, for example, the geometric shapes of two spirals or two moons, patterns in some real-world data cannot be detected just by looking at them. Conventional clustering algorithms like k-means are easy to understand and often gives satisfying results. However, k-means fails to detect the patterns when data is not differentiated by the pairwise distances between points. One of the most important developments in past decades toward overcoming the disadvantages of traditional algorithms is spectral clustering, a graph-based algorithm that relies on the eigenstructure of a similarity matrix, hence the name spectral clustering. It is relatively simple to implement and can be solved efficiently by using basic linear algebra software. In this thesis, which is mainly based on the work of Ulrike Luxburg, we want to give a comprehensive overview of the basis of spectral clustering and its application to image segmentation.

Acknowledgments

There are many people without whom this project would not have been possible. I first wish to express my gratitude for my thesis supervisor, Prof. Beresford Parlett, for his inspiration, guidance, and helpful critiques of this research work.

I would like to recognize the The Berkeley Segmentation Dataset and Benchmark for making their sample data available.

Finally, I wish to thank my parents, for their unconditional love to me and support me in every way and without whom I would never made it to the point of writing this.

Declaration of Academic Integrity

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such. This paper has neither been previously submitted to another authority nor has it been published yet.

10/15/2017 [illegible]

University of California, Berkeley
Mathematics Department

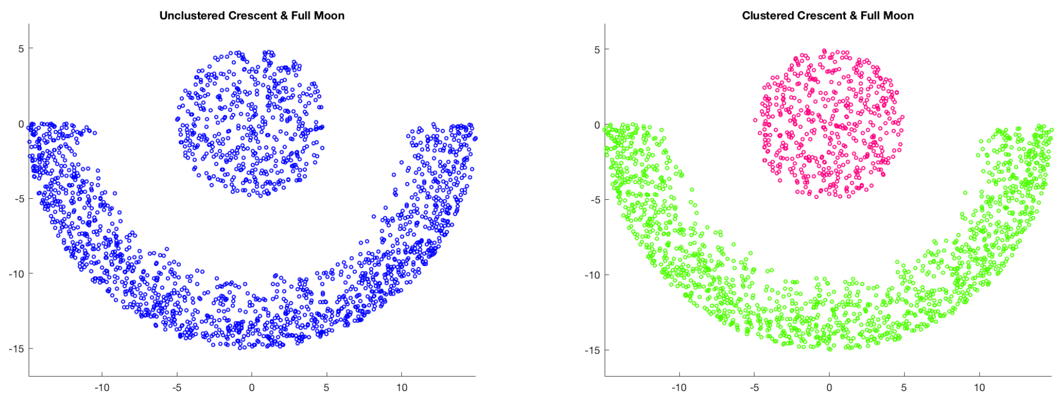
Contents

List of Figures

Chapter 1

Introduction

The objective of data clustering is to divide the data into groups that represent different patterns within the data. The human eye can easily detect clusters in data that contains two well-defined groups (see Fig. 1.1).

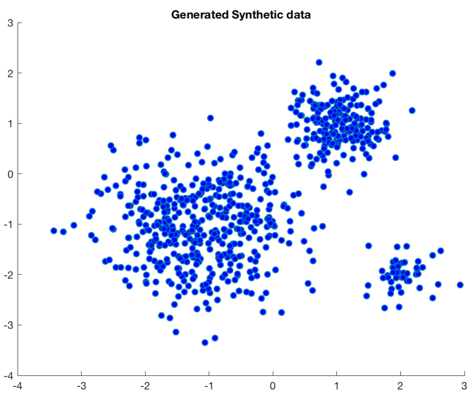


(a) (b)

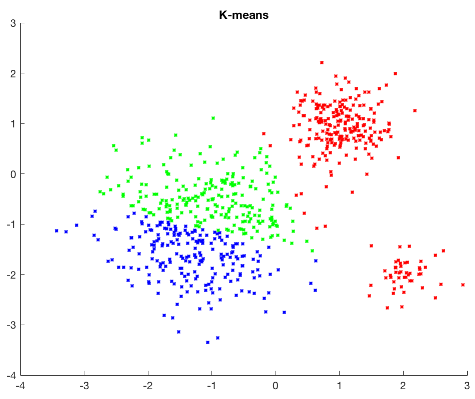
Figure 1.1: A well-separated data (a) Unclustered Crescent & Full Moon and (b) Clustered Crescent & Full Moon.

A traditional clustering algorithm like k-means can correctly detect the pattern of the sample data in Fig.1.1. However, real-world data is not as well-separated as is the data in Fig.1.1. Fig.1.2 shows simple synthetic data that contains clusters that k-means fails to detect. This is where spectral clustering comes into play. Spectral clustering is a graph-based technique based on the structure of eigenstructure of a similarity matrix. The similarity matrix is derived by evaluating the similarity relations of each pair of points in the data set. An analysis of the eigenstructure is partitioned into disjointed clusters; high-similarity points are grouped together and low-similarity points are separated from each other in different clusters. Unlike other traditional clustering methods such as k-means, spectral clustering has no restrictions on the structure of data, and usually outperforms tradition algorithms. The main idea of spectral clustering is to transform the data in such a way that it would become easier to apply the k-means algorithm to it. However, there are several factors, for example, kernel parameter choices, that determine the performance of spectral clustering, which is still an ongoing research topic today. Spectral clustering has found increasing support in many applications such as data mining, pattern recognition, image segmentation, speech separation. In this thesis, we will focus on giving a comprehensive introduction to the basis of spectral clustering, and its application to image segmentation.

The structure of the rest of the report is as follows. In Chapter 2, we will briefly introduce the basic concepts and terminology needed to derive and understand spectral clustering, including eigenstructure, conventional clustering, and graph theory. In Chapter 3, we will derive spectral clustering based on theory covered in Chapter 2. We will also discuss spectral clustering from graph-based methods that lead to different spectral algorithms. In Chapter 4, we will describe spectral clusterings implementation using MATLAB and run it on synthetic data to see its advantages and disadvantages. We will also introduce image segmentation and how spectral clustering can potentially be utilized in this application. Chapter 5 is dedicated to experiments on and the results of image segmentation. We will carefully describe the simulations and discuss the outcomes of experiments testing aspects of spectral clustering. We will end this section by discussing potential research directions. A brief conclusion is in Chapter 6.



(a)



(b)

Figure 1.2: (a) Original data and (b) Clustered data. Obviously, k-means fails to cluster the data. K-means splits data into 3 equal volume regions. As we can see, k-means is insensitive to the differing cluster density. In fact , k-means cannot adapt to the different cluster densities, even when the clusters are spherical, have equal radii, and well-separated.

Chapter 2

Background and Terminology

In this section, we will briefly introduce the basic concepts and terminology needed to prepare you for the derivation of spectral clusterings algorithm. We will first briefly review eigenvalues and eigenvectors of matrices and then look at the traditional k-means clustering algorithm, which will be used to cluster data after applying spectral clustering. However, this section is dedicated to mathematical contents used by spectral clustering: similarity graphs and Laplacians graphs and their properties. Finally, we will explain how the algorithm actually works regarding a graph-cut point of view. Solving different graph-cut problems will lead us to different spectral clustering algorithms we are interested in.

2.1 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors arise prominently in the analysis of a wide-range application such as stability analysis, vibration analysis, atomic orbitals, facial recognition, and matrix diagonalization. We will mainly relate eigenvalue and eigenvector in the context of matrix linear transformation in this section. However, since they are not the objectives of this thesis, we will keep this section short and rather reference to [4] for further details.

Definition 1: Let $A \in R^{n \times n}$ be a real-valued, square matrix. A scalar λ is called an eigenvalue of the matrix A if there is a nontrivial solution x of $Ax = \lambda x$. Such an x is called an eigenvector corresponding to the eigenvalue λ.

A matrix A is called symmetric matrix if $A = A^T$. An elegant proof from any linear algebra text can show that symmetric matrices have at least one real eigenvalue. This property will be as of our interest since we will be mostly looking at symmetric matrices later. The next theorem shows how eigenvalue and eigenvector can be computed.

Definition 2: A scalar λ is an eigenvalue of an $n \times n$ matrix A if and only if λ satisfies the *characteristic equation*

$$det(A - \lambda I) = 0$$

In other words, eigenvalues can be found by solving the characteristic polynomial. The eigenvector corresponding to eigenvalue λ can be solved by solving the linear system of equation $(A - \lambda I)x = 0$

In practice, eigenvalues of large matrices are not computed using the characteristic polynomial. For the large matrices, computing the polynomial becomes expensive in itself and exact roots of a high-degree polynomial can be difficult to compute and express. We will soon face large sparse matrices in which most of the elements are zero as we approach similarity graph. These sparse matrices can be handled efficiently by using, for example, conjugate gradient method and GMRES methods. For futher details in this topic, we would rather reference to [4].

2.2 Conventional Clustering: K-means

K-means algorithm is one of most powerful traditional methods in clustering data. It works by discovering the center or centroid of the clusters. The algorithm finds k centroids within a dataset which each corresponds to a cluster. The k-means algorithm first starts with k centroids $\mu_1, \mu_2, ..., \mu_k$ chosen randomly, then alternate between these two steps until convergence: {

1. For every i, set:
$$c^{(i)} := arg\min_j ||x^{(i)} - \mu_j||$$

2. For each j, set
$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{1\{c^{(i)} = j\}}$$

}

The inner-loop of the algorithm above carries out two steps repeatedly: (1) grouping points into clusters, where each cluster contains all of the closet points to the same centroid, and (2) recomputing the new centroid for each new cluster. Fig. 2.1 shows an illustration of k-means procedure.

Although the algorithm is simple itself, there are several questions left to answer. First, k-means is an iterative method, we need a stopping condition. This is usually done by setting the maximum number of iterations or terminating the loop when the new centroid is within an ϵ of the old one. Second, since k-means works mainly based on the distance from each points to the centroid, we need to define a distance metric. Usually, Euclidean or other prevalent metrics will be used, but the choice of metric depends on the structure of data. Third, choosing the initial centroids is a crucial step in k-means that strongly effects on the outcome; therefore, a random centroid may not give the best result. Lastly, we need a method to determine the quality of the clusters. Let us consider the distortion function $J(c, \mu)$ as follows,

$$J(c, \mu) = \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}|| \tag{2.1}$$

J measures the sum of squared distances between each points to the cluster centroid to which it has been assigned. Notice that the distortion function is a non-convex

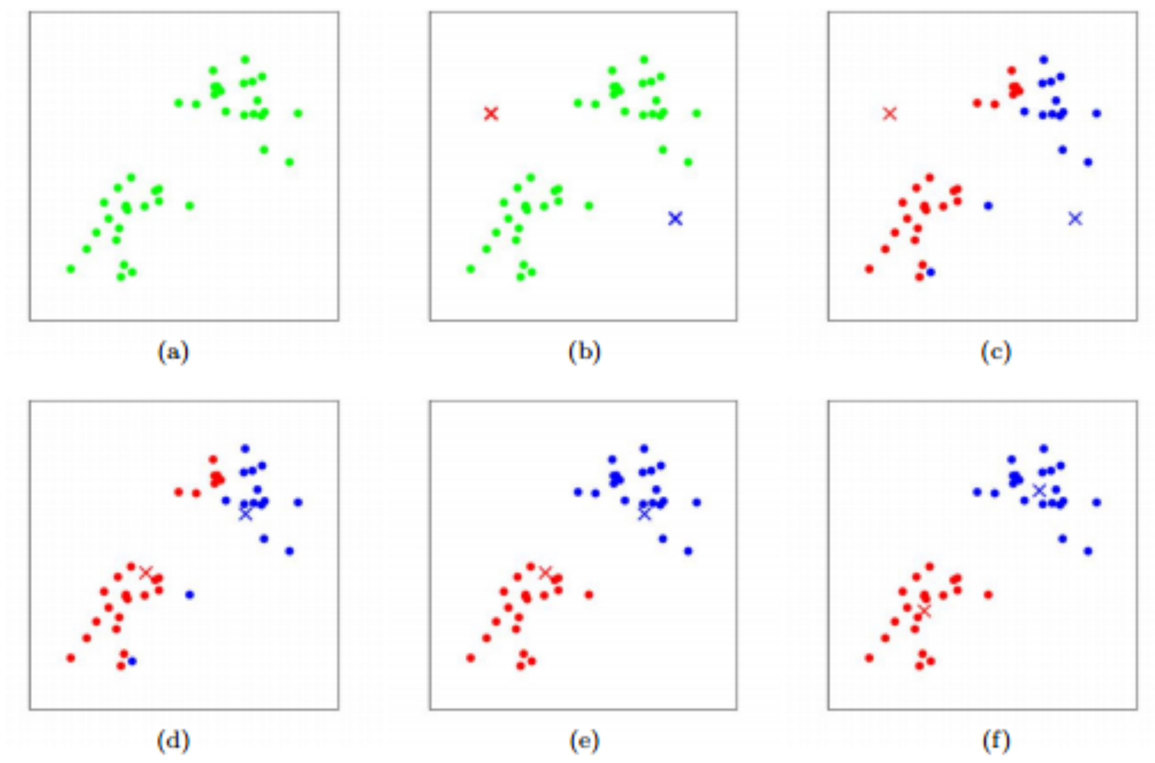


Figure 2.1: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means. In each iteration, we assign each training example to the closest cluster centroid (shown by ”painting” the training examples the same color as the cluster centroid to which is assigned); then we move each cluster centroid to the mean of the points assigned to it. Images courtesy of Michael Jordan. Ref: Andrew Ng.

function, hence it doesn't guarantee to converge to the global minimum. To avoid the issue that k-means may get stuck in a local minimum, we can run k-means multiple times and out of all the different clusterings found, we will pick the one that gives the lowest distortion $J(c, \mu)$.

2.3 Similarity Graph

We will now move on to the main concept of graphs that will lead us to the Spectral Clustering later on.

2.3.1 Graph Notation

Given a set of data points $x_1, ..., x_n$, we want to formulate the relation between all pairs of data points in such a way that allows us to describe and understand these crucial information freely.

Definition 3: *(**Graph**) Let $G = (V, E)$ denotes a (directed) graph with a non-empty finite set $V = \{v_1, v_2, ..., v_3\}$ of vertices and a set $E \subset V \times V$ of edges between* *such vertices. Edges correspond to the notion of similarity between points. If two vertices v_i and v_j are connected by an edge, we denote it by e_{ij}. Moreover, if E is symmetric, that is $e_{ij} \in E \iff e_{ji} \in E$ for all edges, then G is an undirected graph.*

Figure 2.4 shows examples for different types of graphs. Although the similarity between vertices can be described by them being connected by an edge, the directed or undirected graph does not enable us to understand a deep level of information about how closely they are related. Hence, we need a more intuitive way to represent these information. Let us introduce weighted graphs.

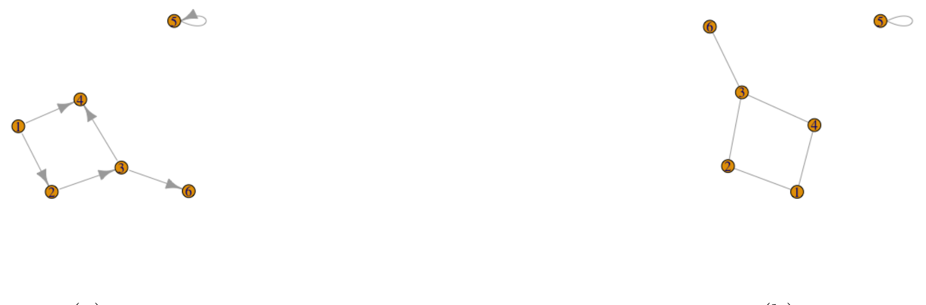


Figure 2.2: Example of (a) Directed Graph and (b) Undirected Graph

Definition 4: *(**Weighted graph**) Let $G' = (V, E')$ be a graph as defined above with E being replaced by $E' = E \times R_0^+$. E' denotes the set of edges whose each e_{ij} now carries a non-negative weight w_{ij}. Then G' is called a weighted graph.*

Definition 5: *(**Adjacency matrix**) The weighted adjacency matrix of G' is the matrix $W = (w_{ij})_{i,j=1,\ldots,n}$. Two vertices x_i and x_j are connected if there weight/similarity w_{ij} is greater than a certain threshold.If $w_{ij} = 0$, then v_i and v_j are not connected by an edge. As G' is an undirected graph, we have $w_{ij} = w_{ji}$ for all weights which shows that W is a symmetric matrix.*

Fig. 2.4 shows an illustration of an adjacency matrix.

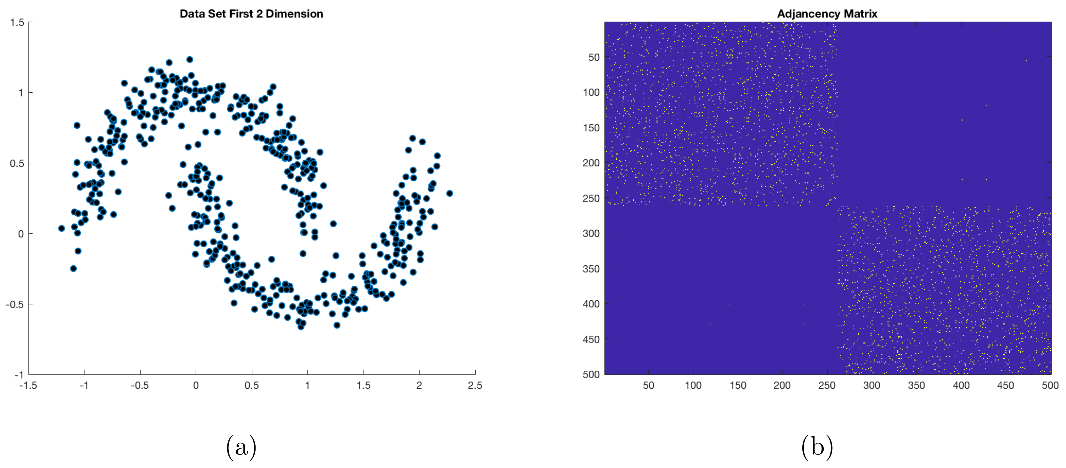


Figure 2.3: (a) Original Data and (b) Adjacency Matrix. The two clusters can be well-represented using adjacency matrix.

Definition 6: *(**Degree Matrix**) Let G be an undirected, weighted graph as described above. The degree of vertex v_i of G is defined as*

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

which tells us how many points each point is connected to. The degree matrix D is defined as a diagonal matrix whose diagonal entries are $d_1, ..., d_n$.

For convenience, we introduce the short hand notation $\{i|v_i \in A\}$ for a subset $A \subset V$. In addition, for two not necessarily disjoint sets $A, B \subset V$, we define

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

We will mainly work on graph partition which requires us to measure the size of a subset $A \subset V$. One simple way is to consider the total number of vertices in A, denoted as $|A|$. Another way is, however, to sum over the weights of all edges in A, called the volume of A and denoted by $\mathbf{vol(A)} := \sum_{i \in A} d_i$

Intuitively, we want to use graph to visualize the connection between data points (vertices), and hence, to explore the underlying structure of the data. In the ideal case, we hope to end up having all points in different clusters infinitely far apart. This can be described by well-seperated partitions in the graph, which we now define as graph connected components.

Definition 7: *(**Connected Components**) Let G be an undirected, weighted graph as defined above and a subset $A \subset V$. We say A is connected if any two vertices in A can be joined by a path such that all intermediate points also lie in A. Moreover, A is called a connected component if it is connected and if there are no connections between the verticies in A and its complement $A^c := V \setminus A$.*

The nonempty subset $A_1, ...A_k$ of V will form a partition of the graph if $A_i \cap A_j = 0$ and $A_1 \cup A_2 \cup ... \cup A_k = V$. We will introduce the last concept of graph notation before moving on to similarity graph which plays an important in the derivation of the spectral clustering algorithm.

Definition 8: *(**Indicator Vector**) Let A be a subset of V as defined above. The indicator vector of A is defined by $\mathbf{1}_A = (f_1, ..., f_n)' \in \mathbf{R}^n$ with*

$$f_i = \begin{cases} 1 & \text{if } v_i \in A \\ 0 & \text{otherwise} \end{cases}$$

2.3.2 Similarity Graph

After introducing some basic concepts of graph, we will now come to one of the most useful tools used to describe the pairwise similarity of points in a given data set.

Definition 9: *(**Graph Similarity**) Given a set $D = \{x_1, ..., x_n\}$ of data points and assume that we have values s_{ij} describing the similarity of x_i and x_j for all those* *data. We then construct the graph $G = (V, E)$ by using the data points as vertices and weighting each edge with corresponding similarity, i.e $w_{ij} := s_{ij}$. Then, G is called the similarity graph of this dataset.*

Before we proceed to discuss several common types of constructing similarity graph, we need to point out two things: First, we will refer to both graph itself and its adjacency matrix as similarity graph since they both represent the pairwise distance between data points. Second, a set of n data points will result in a $n \times n$ similarity matrix and n usually a large number. Having said that, we need to consider technological limits in computer memory. We will therefore alter the shape of similarity matrix in such a way that can bycome the obstacles. The idea is to make the matrix sparse, which means that most of the entries are zeros. This will make the computation more efficient using a MATLAB's procedure called *Compressed Sparse Column*. In the next section, we will focus on discussing several popular constructions of the similarity matrix which lead us to desired outcomes. The goal of constructing similarity graph is to model the relation between each point and its neighborhood. There are different constructions of similarity graph, most widely used ones are ϵ - neighborhood graph, k-nearest neighbor graphs, and the fully connected graph.

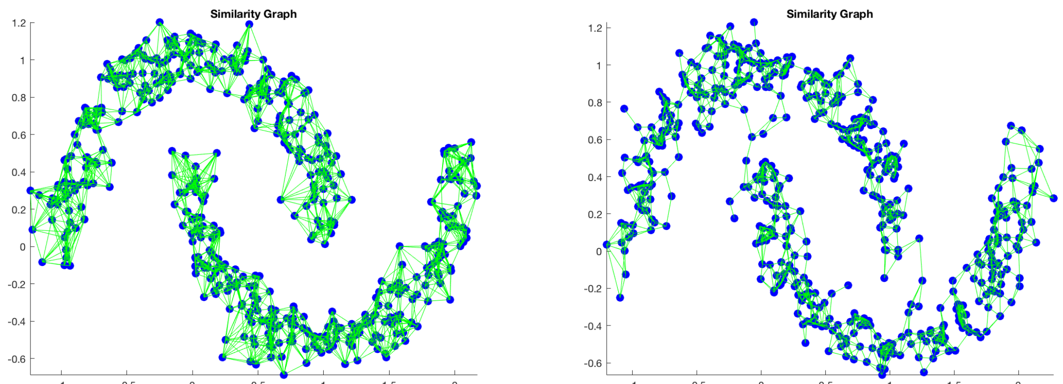
1. **Full Similarity Graph**
   The easiest way to construct the similarity graph is to simply connect all points with positive vertices v_i and v_j if $w_{ij} \geq 0$. Notice that the weight w_{ij} measures the similarity of vertex v_i to its neighbor v_j. This can be done using the Gaussian similarity function
   $$w_{ij} := exp(-\frac{d(x_i, x_j)^2}{2\sigma^2})$$
   where the parameter σ controls how rapidly the similarity W_{ij} falls off with the distance between v_i and v_j. We will later describe a method to choose σ automatically. However, the disadvantage of this type of graph is that it lacks the sparse structure that we desire; hence, it is not recommended for large data sets.

2. **k-Nearest Neighbors Similarity Graph**

   Here we connect the vertex v_i with vertex v_j if v_j is among the k nearest neighbors of v_i for a fixed number k. However, this will result in a directed graph. To convert it to undirected graph, we can use one of the following ways:

   - *Normal k-Nearest Neighbors:* Connect the two vertices if either one of them is among the k nearest neighbors of the other one.
   - *Mutual k-Nearest Neighbors:* Connect the two vertices if v_i them is among the k nearest neighbors of v_j and v_j is among the k nearest neighbors of v_i.
     In both cases, we use the similarity w_{ij} to weight the edge.

3. **ϵ-Neighborhood Similarity Graph:** Connect all vertices whose pairwise similarity are smaller than ϵ. This graph is considered as an unweighted graph since all connected points are roughly on the same scale and at most ϵ.



(a) (b)

Figure 2.4: Both pictures show a k-nearest neighbors similarity graph for two half-moons data. Two connected components in (a) shows promising clusters compared to one connected component in (b).

2.3.3 Graph Laplacians

Graph Laplacians are most important matrices in study of spectral clustering. Spectral Graph theory is the field of these matrices because they hold many useful information about the graphs. We will briefly discuss some main properties of graph Laplacians that are necessary for our proofs later.

Definition 10: *The unnormalized graph Laplacian is defined as*

$$L = D - W$$

From this point, eigenvalues will always be ordered in an increasing order. By "the first k eigenvectors", we refer to the eigenvectors of the k smallest eigenvalues.

Proposition 1: *(Properties of L) Let L be a graph Laplacian as described above. Then*

1. For every vector $f = (f_1, ..., f_n)' \in \mathbf{R}$, we have

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

2. L is symmetric and positive semi-definite.
3. The smallest eigenvalue of L is 0, a corresponding eigenvector is the constant one vector $\boldsymbol{1}$.
4. L has n non-negative, real-values eigenvalues $0 = \lambda_1 \leq \ \lambda_2 \leq ... \leq \lambda_n$.

Proof:

1. $f'Lf = f'(D-W)f = f'Df - f'Wf$

$$= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} = \frac{1}{2}(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{i=1}^{n} d_j f_j^2)$$

$$= \frac{1}{2}(\sum_{i,j=1}^{n} w_{ij} f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{i,j=1}^{n} w_{ij} f_j^2)$$

$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_j^2 - 2f_{ij} + f_j^2) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

In particular, since w_{ij} is non-negative for all vertices, this shows that $f'Lf \geq 0 \ \forall f \in \mathbf{R}^n$.

2. Notice that we assume the graph to be undirected, W is symmetric and D is a diagonal matrix; hence, L is symmetric. As $f'Lf \geq 0 \ \forall f \in \mathbf{R}^n$, it follows that L is symmetric and positive semi-definite.

3. We want to show that $L\mathbf{1} = \mathbf{0}$ which can be rewritten as $D\mathbf{1} = W\mathbf{1}$. Since the sum of the i-th row of W is exactly what we defined to be the i-row of D, which proves that $L\mathbf{1} = 0$

4. This is straightforward since we have proven that L is symmetric and positive-definite.

For further details in unnormalized Laplacian graph, we would rather refer to [9].

Proposition 2: *Let G be an undirected, weighted graph. The geometric multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, ..., A_k$ in the graph. The eigenspace of this eigenvalue is spanned by the indicator vectors $\mathbf{1}_{A_1}, ..., \mathbf{1}_{A_k}$ of those components.*

Proof: Let we start by looking at the case of a connected graph, i.e $k = 1$, and the generalization for arbitrary k will be followed from there. Suppose f is an eigenvector of eigenvalue 0, i.e $Lf = 0$, then by proposition 1, we have

$$0 = f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

The sum is 0 if all terms $w_{ij}(f_i - f_j) = 0$. Since $w_{ij} > 0$ for all connected verticies v_i and v_j, we can conclude that $f_i = f_j$. Hence, f is constant for all the verticies in the graph that can be connected and since we assume the whole graph is connected, f is constant on the whole graph. Since the multiple of an eigenvector is also an eigenvector, we see that $f = \mathbf{1}$ is the only eigenvector of eigenvalue 0. It is obvious to see that f is the indicator vector for the connected component.

Now assume that there are k connected components in the graph. For convenience, we assume that the points in our data set are ordered according to which cluster they are in. By doing that, we can achieve the adjacency matrix W in a block diagonal form as follows

$$L = \begin{bmatrix} L_1 & 0 & 0 & \dots & 0 \\ 0 & L_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & L_k \end{bmatrix}$$

Each one of the block L_i represents a graph Laplacian and therefore a graph. More precisely, it represents the i-th connected component subgraph of G. Since L is block diagonal, its eigenvalues and eigenvectors are the union of the eigenvalues and eigenvectors of its blocks by padding appropriately zeros in other entries. We have shown that each L_i has the constant vector $\mathbf{1}$ as its eigenvector with eigenvalue 0. Hence, L has as many eigenvalues 0 as there are connected components and the corresponding eigenvectors are the indicator vectors of the connected components.

2.3.4 The Normalized Graph Laplacian

So far we have discussed unnormalized graph Laplacian, we will now move on to normalized graph Laplacian, which will eventually lead us to different algorithms of spectral clustering.

Definition 11: *We define the two normalized graph Laplacian as follows,*

$$L_{sym} := D^{-1/2} L D^{-1/2}$$
$$L_{rw} := D^{-1} L$$

where L_{sym} stands for symmetric matrix and L_{rw} is related to random walk. We also note that in order for these matrices to be well-defined, we need $det(D) = \Pi_i d_i \neq 0$, which requires $d_i \neq 0$.

Proposition 3: Let L_{sym} and L_{rw} be defined as above, then we have the following properties:

1. For every vector $f = (f_1, ..., f_n)' \in \mathbf{R}$, we have

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

.

2. λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvalue of L_{sym} with eigenvector $w = D^{\frac{1}{2}} u$.

3. λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigenproblem

$$Lu = \lambda Du$$

4. 0 is an eigenvalue of L_{rw} with the constant one vector $\mathbf{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{\frac{1}{2}}\mathbf{1}$.

5. L_{sym} and L_{rw} are positive semi-definite and have n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq ... \leq \lambda_n$.

Proof:

1. This can be proved similarly to part(1) in proposition 1 by replacing f_i with $\frac{f_i}{\sqrt{d_i}}$ and $f'D^{-\frac{1}{2}}$ is just the scaled version of f'.

2. Suppose we have $L_{sym}w = \lambda w$, it follows that $D^{\frac{-1}{2}}L_{sym}w = D^{\frac{-1}{2}}\lambda w$. Notice that
$$D^{\frac{-1}{2}}L_{sym} = D^{\frac{-1}{2}}D^{\frac{-1}{2}}LD^{\frac{-1}{2}} = D^{-1}LD^{\frac{-1}{2}} = L_{rw}D^{\frac{-1}{2}}$$
Substituting $L_{rw}D^{\frac{-1}{2}}$ to $L_{sym}w = \lambda w$, we have $L_{rw}D^{\frac{-1}{2}}w = \lambda D^{\frac{-1}{2}}w$. By letting $u = D^{\frac{-1}{2}}w$, we complete the proof.

3. Multiplying D on the left of both side of $L_{rw}u = \lambda u$ and notice $DL_{rw} = DD^{-1}L = L$, we get $Lu = \lambda Du$

4. Proposition shows that $L\mathbf{1} = 0$. It follows that $L_{rw}\mathbf{1} = D^{-1}L\mathbf{1} = D^{-1}0 = 0$. That means $\mathbf{1}$ is an eigenvector to eigenvalue 0. The second statement follows directly from 2.

5. The statement about L_{sym} follows from 1 and L_{rw} follows from 2 as in proposition 1.

We will close this section with one last proposition before we move on to the most interesting part, spectral clustering algorithm.

Proposition 4: (Number of connected components and spectra of L_{sym} ***and*** L_{rw}***)*** *Let G be an undirected, weighted graph as defined in earlier section. Then the geometric multiplicity k of the eigenvalue 0 of both* L_{sym} *and* L_{rw} *equals the number of connected components* $A_1, ..., A_k$ *in the graph.*

For L_{rw}, the eigenspace of 0 is spanned by the indicator vectors $\mathbf{1}_{A_i}$ of those components. For L_{sym}, the eigenspace of 0 is spanned by the vectors $D^{\frac{1}{2}}\mathbf{1}_{A_i}$.

Proof: This proof is analogous to proposition 2 by using the properties from proposition 3.

Chapter 3

Spectral CLustering Algorithm

3.1 Graph Cuts

Up to this point, we have developed some solid theory and background about the graph and their properties. Our main goal of doing that is try to reformulate our clustering problem in the form of similarity graph. As a result, we are now able to repharse our original clustering problem in the graph language as follows: After presenting data points in the form of similarity graph, we look for partitions in the graph according to the similarity between vertices i.e the points within the same group are highly similar whereas points in various group are dissimilar to each other. Some representative graph-cut methods are Min-Max Cut, Ratio-Cut, and Normalized Cut. In our case, normalized cut will be focused on.

Definition 12: *(**MinCut**) Let W be the adjacency matrix of a given similarity graph and $k \in \boldsymbol{N}$. Then, the MinCut problem consists of choosing a partition $A_1, ...A_k$ such that*

$$cut(A_1, ..., A_k) := \frac{1}{2}\sum_{i=1}^{k} W(A_i, A_i^c) \tag{3.1}$$

for $A_1, ..., A_k \subset V$ is minimized.

Unfortunately, solving the mincut problem usually results to isolated vertices since MinCut cuts with lesser weight than the ideal cut. This is totally opposite to what we would like to see. In order to overcome this obstacles, we recall the two ways of measuring the size of such partitions. This leads us to two graph cut problems that circumvent the mincut disadvantages by making the sets $A_1, ..., A_k$ "reasonably large".

Definition 13: *Given the adjacency matrix W as above, the RatioCut problem consists of minimizing*

$$RatioCut(A_1, ..., A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, A_i^c)}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, A_i^c)}{|A_i|} \tag{3.2}$$

Definition 14:*Given the adjacency matrix W as above, the NCut (normalized* *cut) problem consists of minimizing*

$$NCut(A_1, ..., A_k) := \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i, A_i^c)}{vol(A_i)} = \sum_{i=1}^{k}\frac{cut(A_i, A_i^c)}{vol(A_i)} \tag{3.3}$$

Both cut problems approach the cut such that weight of edges connecting verticies in A_i to vertices in A_j are minimized *and* size of A_i and A_j are very similar. However, they did it in two different ways: whereas RatioCut takes the number of vertices into account, Ncut considers the edge weights. Unfortunately, unlike k-means algorithm, [13] shows that these two graph cut problems are NP-hard. Therefore, instead of solving them directly, we will rely on approximating the solutions, which is the motivation for Spectral Clustering.

3.2 Spectral Clustering

We finally have enough tools to derive the spectral clustering algorithm. We will do that by approximating the RatioCut and NCut problems in the previous section. In fact, spectral clustering is a way to solve the relaxed version of those problems in the sense of switching from discrete to a continuous problem. Later on, we will experience two different algorithms resulted from relaxing RatioCut and Ncut. We will end this section by discussing some practical issues and introducing its applications.

3.2.1 Unnormalized Algorithm - RatioCut Problem

Let we start with the case of two clusters, i.e $k = 2$. The generalization of arbitrary k can be followed easily from that.

Approximating RatioCut for $k = 2$

For $k = 2$, our objective is to solve

$$\min_{A\subset V} RatioCut(A, A^c). \tag{3.4}$$

For a subset $A \subset V$, we define vector $f \in R^n$ as

$$f_i = \begin{cases} \sqrt{\frac{|A^c|}{|A|}} & \textit{if } v_i \in A \\ -\sqrt{\frac{|A|}{|A^c|}} & \textit{if } v_i \in A^c \end{cases} \tag{3.5}$$

Proposition 5: Let A be a subset of V. Consider the Laplacian graph L of the given graph and f as defined above. Then

$$f'Lf = |V|.RatioCut(A, A^c)$$

Proof: From proposition 1,

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

$$= \frac{1}{2}\sum_{i\in A, j\in A^c} w_{ij}\Big(\sqrt{\frac{|A^c|}{|A|}} + \sqrt{\frac{|A|}{|A^c|}}\Big)^2 + \frac{1}{2}\sum_{i\in A^c, j\in A} w_{ij}\Big(\sqrt{-\frac{|A^c|}{|A|}} - \sqrt{\frac{|A|}{|A^c|}}\Big)^2$$

$$= \frac{1}{2}\Big(\sum_{i\in A, j\in A^c} w_{ij} + \sum_{i\in A^c, j\in A} w_{ij}\Big)\Big(\frac{|A^c|}{|A|} + \frac{|A|}{|A^c|} + 2\Big)$$

$$= cut(A, A^c)\Big(\frac{|A^c|}{|A|} + \frac{|A|}{|A^c|} + 2\Big) = cut(A, A^c)\Big(\frac{|A| + |A^c|}{|A|} + \frac{|A| + |A^c|}{|A^c|}\Big)$$

$$= \frac{cut(A, A^c)}{|A|}\frac{(|A| + |A^c|)}{|V|} + \frac{cut(A, A^c)}{|A^c|}\frac{(|A| + |A^c|)}{|V|}$$

$$= |V| RatioCut(A, A^c)$$

In addition, we notice that

$$f\mathbf{1} = \sum_{i=1}^{n} f_i = \sum_{i\in A}\sqrt{\frac{|A^c|}{|A|}} - \sum_{i\in A^c}\sqrt{\frac{|A|}{|A^c|}} = |A|\sqrt{\frac{|A^c|}{|A|}} - |A^c|\sqrt{\frac{|A|}{|A^c|}} = 0 \tag{3.6}$$

by the definition of the inner product, the vector f defined in equation 3.5 is orthogonal to the constant one vector $\mathbf{1}$.
Furthermore,

$$||f||^2 = \sum_{i=1}^{n} f_i^2 = |A|\frac{|A^c|}{|A|} + |A^c|\frac{|A^|}{|A^c|} = |A| + |A^c| = n \tag{3.7}$$

By combining proposition 5, equation 3.6, and 3.7, we can now rewrite the problem in 3.4 as

$$\min_{A\subset V} f'Lf \ \ with \ \begin{cases} f \ \ in \ 3.5 \ and \ f\perp 1 \\ ||f|| = \sqrt{n} \end{cases} \tag{3.8}$$

Still, the problem, of course, is NP-hard. We will now get rid of the discreteness of the problem by allowing f to take arbitrary values than just two particular values. By doing so, we eventually get a relaxed version of the RatioCut problem:

$$\min_{f\in R^n} f'Lf \ \ with \ \begin{cases} f\perp 1 \\ ||f|| = \sqrt{n} \end{cases} \tag{3.9}$$

The problem given in 3.9 can be solved using *Rayleigh-Ritz Theorem*, which says that for a symmetric $A \in R^{m\times m}$, the smallest and largest eigenvalues of A are given by

$$\lambda_{min} = \min_x \ \{\frac{x'Ax}{x'x}| \ \ 0 \neq x \in R^m\} \tag{3.10}$$

$$\lambda_{max} = \max_x \ \{\frac{x'Ax}{x'x}| \ \ 0 \neq x \in R^m\} \tag{3.11}$$

A rigorous proof of Rayleigh-Ritz Theorem can be found at [5].

At the first glance, we may hope that the solution of equation 3.9 would be given by the eigenvector corresponding to the smallest eigenvalues. Unfortunately, from Proposition 1, this eigenvector is the constant one vector $\mathbf{1}$, which obviously cannot be a solution of 3.9 since it violates the condition $f \perp \mathbf{1}$

Proposition 6: *The solution to the equation 3.9 is given by the eigenvector corresponding to the second smallest eigenvalue of L.*

Proof. From proposition 1, we have L is symmetric, hence we find n eigenvectors that are orthogonal to each other and have norm $\sqrt{n}$. This guarantees that all conditions in equation 3.9 are satisfied. In fact, all of the eigenvectors but $\mathbf{1}$ meet the conditions. However, the Rayleigh-Ritz shows that the eigenvector corresponding to second smallest eigenvalue of L is indeed the solution.

The last step we need to do is to re-transform the real-valued solution vector f of the relaxed problem into a discrete indicator vector. This can be done by consider coordinates f_i as points in R and cluster them into 2 clusters by setting

$$\begin{cases} v_i \in A & if \;\; f_i \in C \\ v_i \in A^c & if \;\; f_i \in C^c \end{cases} \tag{3.12}$$

Approximating RatioCut for $k > 2$

The general idea is the same as in $k = 2$, but we need to make few changes. Given a partition $A_1, ..., A_k \subset V$, let $h_j = (h_{1,j}, ..., h_{n,j})'$ be defined as

$$h_{ij} = \begin{cases} \frac{1}{\sqrt{|A_j|}} & if \;\; v_i \in A_j \\ 0 & if \;\; v_i \notin A_j \end{cases} \tag{3.13}$$

for all $i \in \{1, ..., n\}$ and $j \in \{1, ..., k\}$ and let $H \in R^{n \times k}$ be a matrix containing those vectors as columns.

Proposition 7: *With the new conditions defined as above, we have*

1. $h_i^{'} L h_i = \frac{cut(A_i, A_i^c)}{|A_i|}$
2. $h_i^{'} L h_i = (H' L H)_{ii}$

Proof. The proof is similar to the proof of proposition 5.

In addition, we also notice that

$$RatioCut(A_1, ..., A_k) = \sum_{i=1}^{k} h_i^{'} L h_i = \sum_{i=1}^{k} (H' L H)_{ii} = tr(H' L H) \tag{3.14}$$

Combining proposition 7, equation 3.14, and the fact that the columns of H are orthonormal, we can rewrite the RatioCut problem to

$$\min_{A_1, ..., A_k} tr(H' L H) \;\; with \begin{cases} H \;\; as \; defined \; in \; 3.13 \\ H' H = I \end{cases} \tag{3.15}$$

By allowing the entries of H to take arbitrary values, we relax the problem in the same manner as $k = 2$. This leads us to relaxed RatioCut problem:

$$\min_{H \in R^{n \times k}} tr(H'LH) \text{ } with \text{ } H'H = I \tag{3.16}$$

The solution for 3.16 is given using another version of *Rayleigh-Ritz Theorem*, which is stated as below

Theorem: Let $A \in R^{m \times m}$ be a symmetric matrix with eigenvalues $\lambda_1 \leq ... \leq \lambda_m$ and let $v_1, ..., v_m$ be the corresponding orthonormal eigenvectors. Then the solution of the optimization problem

$$\min_{X \in R^{m \times n}} tr(X'AX) \text{ } with \text{ } X'X = I$$

for some $n \in 1, ..., m$ is given by the matrix X containing the first n eigenvectors as columns.

Proof. A rigorous proof can be found at [5].

The last step is the same as in $k = 2$ in which, we apply k-means on the rows of H to transform the real-valued problem to a discrete problem to give us the clustering. This leads to the unnormalized spectral clustering algorithm as presented below.

Unnormalized Spectral Clustering

Input: Given a set of points $S = s_1, ..., s_n$ to cluster into k groups

1. Form the similarity graph with adjacency matrix $W \in R^{n \times n}$ by one of the ways described in section 2.
2. Compute the unormalized graph Laplacian $L = D - W$
3. Compute the first k eigenvectors $u_1, u_2, ...u_k$ of L
4. Form a matrix $U \in R^{n \times k}$ containing the vectors $u_1, u_2, ...u_k$ as columns.
5. For $i = 1, ..., n$, let $y_i \in R^k$ be the vector corresponding to the i-th row of matrix U.
6. Cluster the points $(y_i)_{i=1}^n$ into clusters $C_1, ...C_k$ using the k-means algorithm

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

3.3 Normalized Algorithms - NCut Problem

The previous section shows that RatioCut leads us to an algorithm using the unnormalized graph Laplacian. Likewise, we will now show that approximating the NCut problem will lead to another algorithm using normalized graph Laplacians.

Approximating NCut for k = 2

In the case $k = 2$, let us define the indicator vector as

$$f_i = \begin{cases} \sqrt{\frac{vol(A^c)}{vol(A)}} & if \;\; v_i \in A \\ -\sqrt{\frac{vol(A)}{vol(A^c)}} & if \;\; v_i \in A^c \end{cases} \tag{3.17}$$

Similarly, We will now show that

1. $(Df)'\mathbf{1} = 0$
2. $f'Df = vol(V)$

Proof. For the first equation

$$(Df)'\mathbf{1} = \sum_{i=1}^{n} d_i f_i = \sum_{i \in A} d_i \sqrt{\frac{vol(A^c)}{vol(A)}} - \sum_{i \in A^c} d_i \sqrt{\frac{vol(A)}{vol(A^c)}}$$

$$= \sqrt{\frac{vol(A^c)}{vol(A)}} vol(A) - \sqrt{\frac{vol(A)}{vol(A^c)}} vol(A^c)$$

$$= \sqrt{vol(A^c)vol(A)} - \sqrt{vol(A)vol(A^c)} = 0$$

For the second equation,

$$f'Df = \sum_{i=1}^{n} f_i^2 d_i$$

$$= \sum_{i \in A} d_i \frac{vol(A^c)}{vol(A)} + \sum_{i \in A^c} d_i \frac{vol(A)}{vol(A^c)}$$

$$= vol(A^c) + vol(A) = vol(V)$$

Proposition 8: With two properties defined above, we have

$$f'Lf = vol(V)Ncut(A, A^c)$$

Proof. From proposition 1, we have that

$$f'Lf = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 = \frac{1}{2} \sum_{i \in A, j \in A^c} w_{ij} \left(\sqrt{\frac{vol(A^c)}{vol(A)}} + \sqrt{\frac{vol(A)}{vol(A^c)}} \right)^2 +$$

$$\frac{1}{2} \sum_{i \in A^c, j \in A} w_{ij} \left(-\sqrt{\frac{vol(A^c)}{vol(A)}} - \sqrt{\frac{vol(A)}{vol(A^c)}} \right)^2$$

$$= \frac{1}{2}\Big(\sum_{i \in A, j \in A^c} w_{ij} + \sum_{i \in A^c, j \in A} w_{ij} \Big) \Big(\frac{vol(A^c)}{vol(A)} + \frac{vol(A)}{vol(A^c)} + 2 \Big)$$

$$= cut(A, A^c)\Big(\frac{vol(A^c)}{vol(A)} + \frac{vol(A)}{vol(A^c)} + 2 \Big) = cut(A, A^c)\Big(\frac{vol(A^c) + vol(A)}{vol(A)} + \frac{vol(A) + vol(A^c)}{vol(A^c)} \Big)$$

$$= vol(V) NCut(A, A^c)$$

Using the first two properties and proposition 8, we can rewrite the problem as

$$\min_{A} f'Lf \;\; with \begin{cases} f \;\; in \; 3.17 \;\; and \;\; Df \perp 1 \\ f'Df = vol(V) \end{cases} \tag{3.18}$$

We will allow f to take arbitrary values, thus relaxing the problem to

$$\min_{f \in R^n} f'Lf \;\; with \begin{cases} Df \perp 1 \\ f'Df = vol(V) \end{cases} \tag{3.19}$$

But we are not done yet, now we substitute $g := D^{1/2} f$ which makes the problem become

$$\min_{g \in R^n} g' D^{\frac{-1}{2}} L D^{\frac{-1}{2}} g \;\; with \begin{cases} g \perp D^{\frac{1}{2}} 1 \\ ||g||^2 = vol(V) \end{cases} \tag{3.20}$$

Observe that $D^{\frac{-1}{2}} L D^{\frac{-1}{2}} = L_{sym}$, whose first eigenvector, according to Proposition 3, is $D^{\frac{1}{2}}\mathbf{1}$. Since $vol(V)$ is a constant, we can once again use Rayleigh-Ritz theorem to solve for g. In fact, the solution g of 3.20 is the second eigenvector of L_{sym}. Hence, by proposition 3 again, we have that $f = D^{\frac{-1}{2}} g$ is the second eigenvector of L_{rw}. Now the rest is just like in the unnormalized case. We will cluster the coordinates $f_i \in R$ using 2-means algorithm.

Approximating NCut for $k > 2$

For arbitrary k, we define the indicator vectors $h_j = (h_{1,j}, ..., h_{n,j})'$ by

$$h_{ij} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}} & if \;\; v_i \in A_i \\ 0 & if \;\; v_i \notin A_j \end{cases} \tag{3.21}$$

for all $i \in \{1, ..., n\}$ and $j \in \{1, ..., k\}$ and let $H \in R^{n \times k}$ be a matrix containing those vectors as columns.

Proposition 9: Given h and H as described above, we have

1. $H'H = I$

2. $h_i' D h_i = 1$

3. $h_i' L h_i = \frac{cut(A_i, A_i^c)}{vol(A_i)}$

Proof. The proof for each equation can be done in the same manner as the ones before.

Using proposition 9, we can now rewrite the Ncut problem as following

$$\min_{A_1,\ldots,A_k} tr(H'LH) \ with \begin{cases} H \, as \ defined \ above \\ H'DH = I \end{cases} \tag{3.22}$$

Now substitute $T = D^{\frac{1}{2}}H$ and allow H to take arbitrary values, the problem becomes

$$\min_{T \in R^{n\times k}} tr(T'D^{\frac{-1}{2}}LD^{\frac{-1}{2}}T) \ with \ T'T = I$$

Observe that solution T is given by the matrix containing the first k eigenvectors of L_{sym} as columns. Re-substituting $H = D^{\frac{-1}{2}}T$ and using the proposition 3 show that H is given by the matrix containing the first k eigenvectors of L_{rw} as columns. Finally, running k-means on the rows of H to get discrete indicator vectors and obtain the clusters. This yields the normalized spectral clusering algorithm according to Shi and Malik (2000).

Normalized Spectral Clustering according to Shi and Malik (2000)

Input: Given a set of points $S = s_1, ..., s_n$ to cluster into k groups

1. Form the similarity graph with adjacency matrix $W \in R^{n\times n}$ by one of the ways described in section 2.
2. Compute the unormalized graph Laplacian $L = D - W$
3. Compute the first k eigenvectors $u_1, u_2, ...u_k$ of the normalized graph Laplacian L_{rw}
4. Form a matrix $U \in R^{n\times k}$ containing the vectors $u_1, u_2, ...u_k$ as columns.
5. For $i = 1, ..., n$, let $y_i \in R^k$ be the vector corresponding to the i-th row of matrix U.
6. Cluster the points $(y_i)_{i=1}^n$ into clusters $C_1, ...C_k$ using the k-means algorithm

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

The next algorithm shows a different version of normalized spectral clustering introduced by Jordan and Ng, which uses eigenvectors of L_{sym} rather than L_{rw}. An additional step is required to normalize the row of the matrix U which contains the eigenvectors as columns. This step can be explained using the pertubation theory, but we will not discuss it in details here and rather refer to [12].

Normalized Spectral Clustering according to Ng, Jordan, and Weiss (2002)

Input: Given a set of points $S = s_1, ..., s_n$ to cluster into k groups

1. Form the similarity graph with adjacency matrix $W \in R^{n \times n}$ by one of the ways described in section 2.
2. Compute the unnormalized graph Laplacian $L = D - W$
3. Compute the first k eigenvectors $u_1, u_2, ...u_k$ of the graph Laplacian L_{sym}
4. Form a matrix $U \in R^{n \times k}$ containing the vectors $u_1, u_2, ...u_k$ as columns.
5. Form the matrix $T \in R^{n \times k}$ from U by normalizing the rows, i.e, set

$$T = (t_{ij})_{i,j=1}^{n} \; with \; t_{ij} = \frac{w_{ij}}{\left(\sum_k u_{jk}^2\right)^{\frac{1}{2}}}$$

6. For $i = 1, ..., n$, let $y_i \in R^k$ be the vector corresponding to the $i - th$ row of matrix U.
7. For $i = 1, ..., n$, let $y_i \in R^k$ be the vector corresponding to the i-th row of matrix T.
8. Cluster the points $(y_i)_{i=1}^n$ into clusters $C_1, ...C_k$ using the k-means algorithm

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

Chapter 4

Spectral Clustering: Toy Data and Practical Issues

In this section, we want to discuss some of the practical issues which come up when we actually implement the spectral clustering. We have seen that spectral clustering works with even more parameters than just the number of clusters. Roughly, efficient way to estimate those parameters is still an ongoing research topic today. However, we should note that many other clustering algorithms also have the same problems. We will now take a closer look on each of the issues.

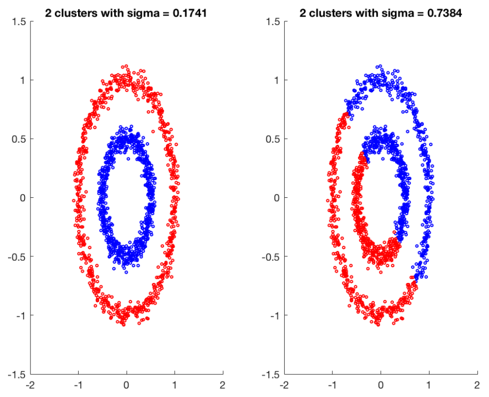
4.0.1 Choice of σ

Recall that the first step in spectral clustering is to calculate the adjacency matrix which represents the similarity between data points. This can be done using the Gaussian similarity function

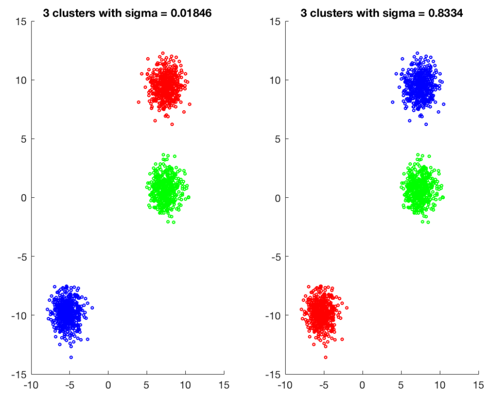
$$w_{ij} := exp(-\frac{d(x_i, x_j)^2}{2\sigma^2})$$

where the parameter σ controls how rapidly the similarity W_{ij} falls off with the distance between v_i and v_j.

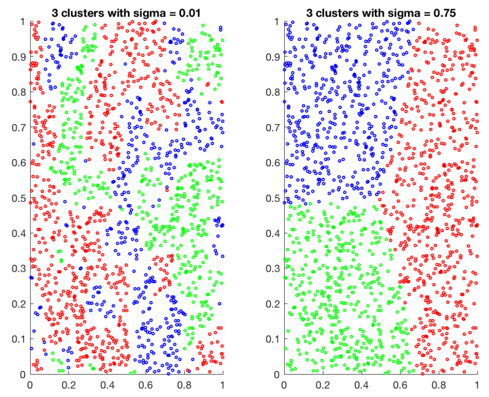
In this section, we will analyze the effect of σ on the clustering. Let us take a look at some clusters produced by spectral clustering using different values of σ in Figure 4.1.



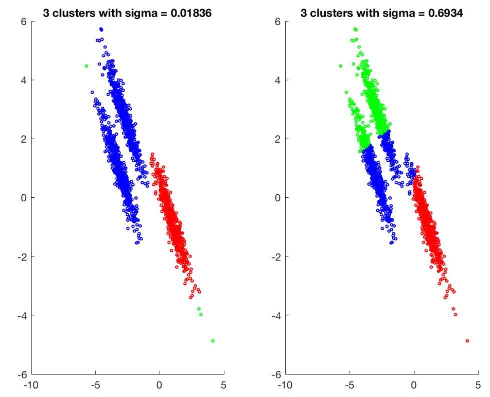
(a)



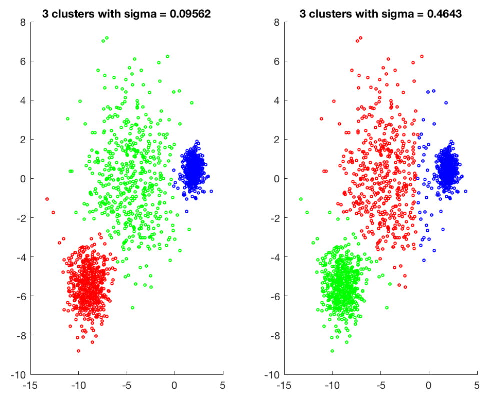
(b)



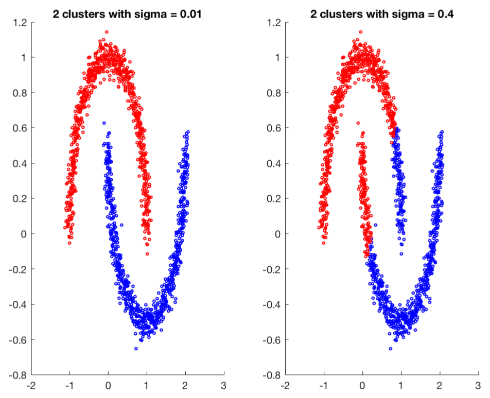
(c)



(d)



(e)



(f)

Figure 4.1: **Spectral Clustering with various σ values**. The optimal σ for each data turned out to be different. This highlights the impact of σ value has on the clustering quality. By slightly changing the σ value, there can be a huge difference in the clustering results.

We will see this issue again when we approach image segmentation in section 5.

Chapter 5

Image Segmentation

Spectral clustering has become an increasingly adopted tool in a variety of research areas. At the end of the last section, we introduced some of its applications-namely, signal theory and educational data mining. In this section, our main focus is image segmentation. In image segmentation, we try to divide image into certain segments or detect edges. A common challenge with image segmentation methods based on spectral clustering is scalability. Let us consider each pixel of an n-by-n pixels image as a data point. Other attributes like RGB values or intensity can also be considered. Therefore, we will end up having n^2 data points to work with. Since spectral clustering involves the eigendecomposition of a pairwise similarity matrix, we will soon face a matrix of size n^2-by-n^2. Even for low-resolution images, this still forces us to deal with very big datasets. Unfortunately, down-sizing the image, however, will cause a loss of finer details and can lead to less accurate segmentation results. A common approach to solve this problem is to applying spectral clustering algorithm to smalls blocks of the image and then merge the resulting segments with methods like a stochastic ensemble consensus. The details of this method can be found in [3]. Having said that, we will first explain how spectral clustering is used on a small-scale image to gain the intuition of how image segmentation works.

5.1 k-means Approach

Suppose we want to use the k-means algorithm described in section 2.2 to segment the image. Consider, for example, the case of a 500-by-500 pixels image. We store each pixel as a data point using its horizontal and vertical position and the RGB channel values, ranging from 0 to 1, as attributes, giving us a 5-dimensional dataset. If we apply k-means on this dataset, using unmodified Euclidean metric would not be a good idea since the pixel position attributes have a much larger range than the RGB values. This would make the position of the pixel have a greater effect on the distance than the color values. Hence, segmenting the picture into certain areas will lead to bad results.

5.2 Spectral Clustering Approach

Again, we consider each pixel in an image as a data point. We will stack n^2 pixels in an n-by-n pixels image as a column vector. One should be aware that two pixels that are far apart but have the same RGB color should not be connected in the similarity graph. If we compute the similarity matrix by computing the similarity between a pixel and all other pixels, then two pixels with the same RGB values will certainly have high similarity despite their positions in the picture. To avoid this issue, we will build a similarity matrix connecting pixels within radius R. In other words, each pixel is connected to its neighboring pixels within radius R.

Let D be a distance matrix computed using block processing for the first color channel. D is a matrix of size number-of-pixels $\times$ number-of-pixels (i.e n^2-by-n^2) defined as follows,

$$D(i,j) = \begin{cases} (im(i) - im(j))^2 & j \textit{ is within radius } R \textit{ of } i \\ 0 & \textit{otherwise} \end{cases} \tag{5.1}$$

where i and j are pixel locations, $im(i)$ is the intensity of pixel i.

Note that if there is more than one color channel, we will sum all the distances since all channels have the same fill pattern in the distance.

The similarity matrix W is then calculated using Gaussian similarity function defined in section 2 as

$$w(i,j) = \begin{cases} e^{-\frac{d(i,j)^2}{2\sigma^2}} & j \textit{ is within radius } R \textit{ of } i \\ 0 & \textit{otherwise} \end{cases} \tag{5.2}$$

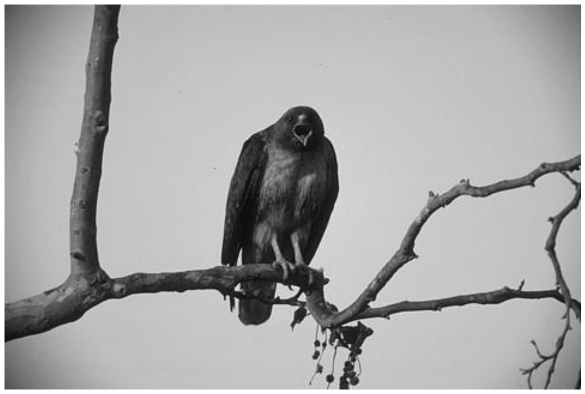
where $w(i,j)$ is the similarity between pixel i and pixel j.

The rest of the algorithm can be proceeded using spectral clustering method introduced by Jordan & Ng.

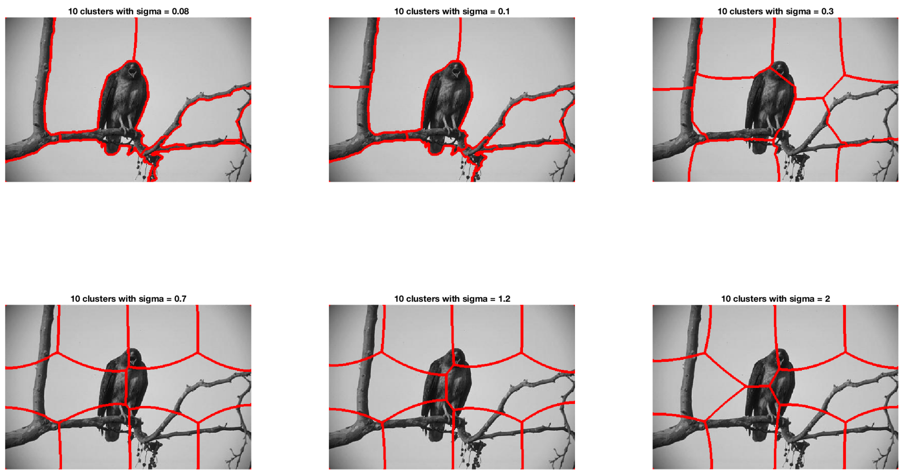
There are two things we need to point out here: first, in order to get the best result in the last step of the algorithm in which we apply k-means to eigenvectors, we will run k-means multiple times, and out of all the different clusterings found, we pick the one with the least distortion. In addition, the first centroid is picked randomly, and the other centroids are selected to be farthest from previous centroids (see [1]). Second, we need to determine the scaling parameter σ as describe in section 4.0.1. The selection of σ is usually done manually. Ng at al. [1] suggested selecting σ automatically by running the algorithm repeatedly for different number of values of σ and choosing the one which provides the least distortion. Though it is better than choosing σ manually, we should note that doing so will increase the computational time.

5.3 Experimental Results

We tested the algorithm on some images available on The Berkeley Segmentation Dataset and Benchmark. We chose to use the gray bird and airplane pictures. In Figure 5.1, we use the spectral clustering introduced by Jordan and Ng.

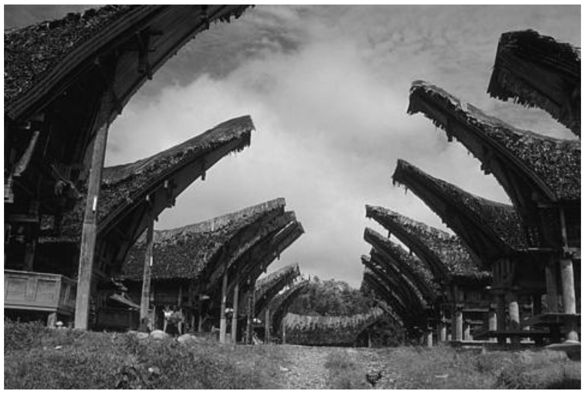


(a)

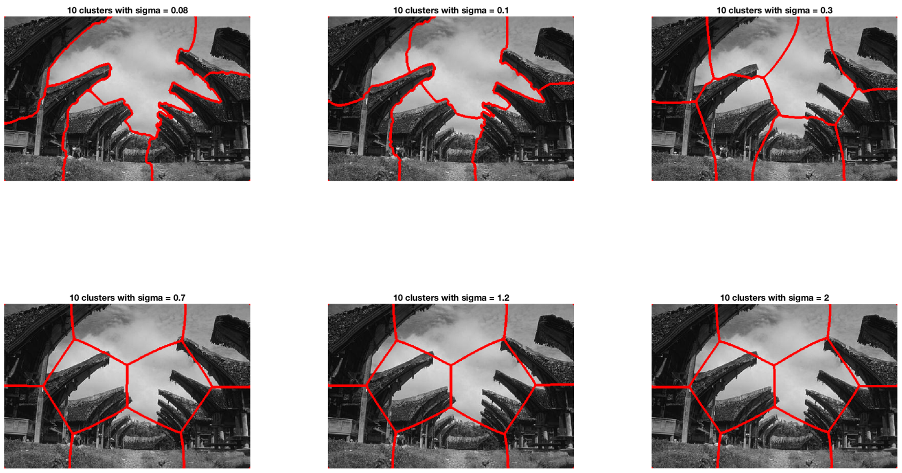

10 clusters with sigma = 0.08
10 clusters with sigma = 0.1
10 clusters with sigma = 0.3
10 clusters with sigma = 0.7
10 clusters with sigma = 1.2
10 clusters with sigma = 2

(b)

Figure 5.1: (a) shows a original picture of a bird in gray scale. Using the different σ values, we set the cluster number to 7 and changing the value of σ. The optimal σ is 0.08. The clustering result starts getting worse as σ gets larger as we can see in (b).



(a)


10 clusters with sigma = 0.08
10 clusters with sigma = 0.1
10 clusters with sigma = 0.3
10 clusters with sigma = 0.7
10 clusters with sigma = 1.2
10 clusters with sigma = 2

(b)

Figure 5.2: (a) shows a original picture of a houses in gray scale. Using the different σ values, we set the cluster number to 10 and changing the value of σ. The clustering result starts getting worse as σ gets larger as we can see in (b).

Chapter 6

Summary and Future Work

Appendix A

Abbreviations

Bibliography

[1] M. A.Ng and Y.Weiss, *On spectral clustering: Analysis and an algorithm.*, 2011. http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf, Accessed 27 June 2017.

[2] L. H. D. Yan and M. I. Jordan, *Fast approximate spectral clustering*, 15th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), 2009. https://people.eecs.berkeley.edu/~jordan/papers/yan-huang-jordan-kdd09.pdf.

[3] A. W. F. Tung and D. A. Clausi, *Enabling scalable spectral clustering for image segmentation*, vol. 43, Pattern Recognition, 2010. https://people.eecs.berkeley.edu/~jordan/papers/yan-huang-jordan-kdd09.pdf.

[4] G. H. Golub and C. F. V. Loan, *Matrix computation*, 1991.

[5] R. A. Horn and C. R. Johnson, *Topics in matrix analysis*, 1985. http://tinyurl.com/zvt37j8, Accessed 26 June 2016.

[6] E. Kandogan, *Visualizing multi-dimensional clusters, trends, and outliers using star coordinates*, In the proceedings of ACM SIGKDD, (2011), pp. 107–116. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.8909&rep=rep1&type=pdf.

[7] P. N. M. Mahajan and K. R. Varadarajan, *The planar k-means problem is np-hard*, WALCOM, 2009.

[8] M. Meila and J. Shi, *A random walks view of spectral segmentation.*, In 8th International Workshop on Artificial Intelligence and Statistics (AISTATS)., 2001.

[9] B. Mohar, *The laplacian spectrum of graphs*, vol. 2, In Graph theory, combinatorics, and applications, p. 871898.

[10] S. V. R. Kannan and V.Vetta, *On spectral clustering good, bad and spectral*, 2000. https://www.cc.gatech.edu/~vempala/papers/jacm-spectral.pdf, Accessed 24 August 2017.

[11] J. Shi and J. Malik, *Normalized cuts and image segmentation*, August 2000. http://www.learningaboutelectronics.com/Articles/Low-pass-filter.php, Accessed 26 July 2016.

[12] U. von luxburg, *A tutorial on spectral clustering*, 2007. `https://www.cs.cmu.edu/~aarti/Class/10701/readings/Luxburg06_TR.pdf`.

[13] D. Wagner and F. Wagner, *Between mincut and graph bisection*, vol. 43, Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS), p. 744750.

[14] Y.Weiss, *Segmentation using eigenvectors: A unifying view*. `https://www.cs.huji.ac.il/~yweiss/iccv99.pdf`, Accessed 26 August 2017.