Spectral Partitioning and its Application to Image Segmentation

Michael Agbo Tettey Soli

10249232

THIS THESIS IS SUBMITTED TO THE UNIVERSITY OF GHANA, LEGON IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MPHIL MATHEMATICS DEGREE

July, 2015

DECLARATION

This thesis was written in the Department of Mathematics of the University of Ghana, Legon from August, 2014 to June, 2015 under the supervision of Dr. Margaret McIntyre and Dr. Douglas Adu-Gyamfi of the University of Ghana, Legon.

I hereby declare that, except where due acknowledgement is made, this work has not been presented wholly or in part for the award of a degree at the University of Ghana or any other institution.

.....
Student: Michael Agbo Tettey Soli

.....
Supervisor: Dr. Margaret McIntyre

.....
Co-Supervisor: Dr. Douglas Adu-Gyamfi

ABSTRACT

The properties of graphs can be studied via the algebraic characteristics of its adjacency or Laplacian matrix. The second eigenvector of the graph Laplacian is one very useful tool which provides information as to how to partition a graph. In this thesis, we study spectral clustering and how to apply it in solving the image segmentation problem in computer vision.

DEDICATION

This work is dedicated to my parents Mr. and Mrs. Ebenezer D. Soli, my troublesome siblings Mrs. Vera Lassey Fiador, Cynthia and Leticia Soli and to all my friends and loved ones who have supported and continue to support me.

ACKNOWLEDGEMENTS

I would have never been able to finish my thesis without the guidance of my team of supervisors, help from friends, and support from my family and loved ones.
My deepest and foremost gratitude goes to the Almighty God for the grace He has placed on my life and seeing me through my studies with strength and favour.
I am specially grateful to my team of Supervisors Dr. Margaret McIntyre, Dr. Douglas Adu-Gyamfi and Mr. Eyram Schwinger for the relentless support they offered me. I am also thankful to the CARNEGIE Foundation and Prof. F.K. Allotey for their financial support. I am thankful to Dr. Edward Prempeh, Dr. Prince Osei-Koree, Mr. Peter Acquaah and especially Dr. Ralph Twum for his support from coursework through to thesis. I also express deep gratitude to members of my MPhil class Justice, Thomas, Leonard, Kingsley, Precious, Grace, Majeed, Christian and Issac for their individual contributions towards the successful completion of my thesis.

Contents

Chapter 1

Introduction

Many real-world situations can conveniently be described by means of a diagram consisting of a set of points together with lines joining certain pairs of these points. For example, the points could represent people, with lines joining pairs of friends; or the points might be communication centres, with lines representing communication links. In such diagrams one is mainly interested in whether or not two given points are joined by a line; the manner in which they are joined is immaterial. A mathematical abstraction of situations of this type gives rise to the concept of a graph.
Due to its inherent simplicity, graph theory has a wide range of applications in engineering, in the physical, social and biological sciences, in linguistics and in numerous other areas. A graph can be used to represent any physical situation involving discrete objects and relationships among them.
Example: The link structure of a website can be represented by a directed graph, in which the vertices represent web pages and directed edges represent links from one page to another.

In this thesis, we study the concept of spectral clustering and its application to the image segmentation problem in computer vision. Image segmentation is described as one of the most important problems in computer vision. It serves as one of the preparatory steps in getting an image ready for image analysis. Generally, image segmentation is the partitioning of an image into groups called segments such that the pixels belonging to the same partition have similar characteristics and differ from pixels from other segments. Although image segmentation is a simple problem, there is no single best known technique for solving this.

We begin by introducing basic concepts in graph theory, matrix representation of graphs in the sections of Chapter 1. We review relevant linear algebra and matrix analysis needed for our work in Chapter 2. We examine the concept of spectral clustering in Chapter 3 using

the tools we reviewed in chapter 2. Chapter 4 focuses on the graph partitioning problem and some methods used in solving it. We discuss the image segmentation problem in Chapter 5 and model the image segmentation problem as a graph partitioning problem under spectral lens. In chapter 6, we implement the image segmentation algorithm in matlab using the Ncut segmentation code provided by Timothee Cour, Stella Yu and Jianbo Shi. We also examine the solution and provide possible improvements to the code in terms of quality of segments produced and the running time of the code.

1.1 Basic Definitions And Overview

Definition 1.1.1 (Graph)**.** A graph $G = (V, E)$ consists of a set of objects $V = \{v_1, v_2, ...\}$ called *vertices* and another set $E = \{e_1, e_2....\}$, whose elements are called *edges*, such that each edge e_k is identified with an unordered pair $\{v_i, v_j\}$ of vertices.

The vertices v_i, v_j associated with the edge e_k are called *end vertices* of e_k. The most common representation of a graph is by means of a diagram, in which the vertices are represented as points and each edge as a line segment joining its end vertices.

The above definition of a graph permits an edge to be associated with a vertex pair $\{v_i, v_i\}$. Such an edge having the same vertex as its end vertices is called a self loop. Formally:

Definition 1.1.2 (Self-Loop)**.** If $G = (V, E)$ is a graph and $v_i \in V$ and $e = \{v_i, v_i\}$, then the edge e is called a *self-loop*. That is any edge that is a single element subset of V is called a *self-loop*.

Definition 1.1.3 (Vertex Adjacency)**.** Let $G = (V, E)$ be a graph. Two vertices v_1 and v_2 are said to be *adjacent* if there exists an edge $e \in E$ such that $e = \{v_1, v_2\}$. A vertex v is *self-adjacent* if $e = \{v\}$ is an element of E.

Definition 1.1.4 (Edge Adjacency)**.** Let $G = (V, E)$ be a graph. Two edges e_1 and e_2 are said to be adjacent if there exists a vertex $v \in V$ such that v is an element of both e_1 and e_2 (as sets). An edge e is said to be adjacent to a vertex v if v is an element of e as a set.

Definition 1.1.5 (Neighbourhood)**.** Let $G = (V, E)$ be a graph and let $v \in V$. The *neighbours* of v are the set of vertices that are adjacent to v.

Definition 1.1.6 (Incidence)**.** Let $G = (V, E)$ be a graph and let $v_1, v_2 \in V$. If v_1 and v_2 are the end vertices of the edge $e \in E$, then e is said to be *incident* on v_1 and v_2. Similarly, v_1 and v_2 are said to be *incident* on e.

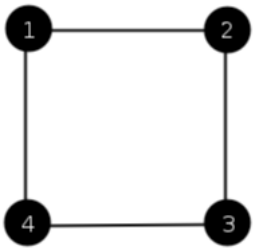


Figure 1.1: A graph represented by a diagram in which vertices are represented by points and edges are represented by lines connecting vertices.

Example 1.1.7. Consider the set of vertices $V = \{1, 2, 3, 4\}$. The set of edges $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}\}$. Then the graph $G = (V, E)$ has four vertices and four edges. It is usually easier to represent this graphically. See figure 1.2 for a visual representation of G.

These visualisations are constructed by representing each vertex as a point and each edge as a line connecting the vertex representations that make up the edge. That is , let $v_1, v_2 \in V$, then there is a line connecting the points v_1 and v_2 if and only if $\{v_1, v_2\} \in E$. In this example, the neighbourhood of vertex 1 is vertices 2 and 4 and vertex 2 is adjacent to vertices 1 and 3.

Definition 1.1.8. (Order) The *order* of a graph $G = (V, E)$ is the number of elements in the vertex set V of the graph. This is the cardinality of the vertex set.

Definition 1.1.9. (Degree) Let $G = (V, E)$ be a graph and let $v \in V$. The *degree* of the vertex v, denoted d_v is the number of edges $e \in E$ incident on v.

If a pair of vertices $v_1, v_2 \in V$ are connected by more than one edge, the edges connecting v_1 and v_2 are known as *parallel edges*. In this case, the edge set E is a multiset.

Let $G = (V, E)$ be a graph. The graph is said to be finite if it has a finite number of vertices as well as a finite number of edges. For our study we will be working with finite graphs only.

Definition 1.1.10. Let $G = (V, E)$ be a graph. A vertex $v \in V$ having no incident edge is called an *isolated vertex*. That is a vertex $v \in V$ of degree zero.

Definition 1.1.11. (Simple Graphs) A graph $G = (V, E)$ is said to be *simple* if it has no *self-loops* or *parallel edges*.

Definition 1.1.12. (Undirected Graph) Let $G = (V, E)$ be a graph and let $v_1, v_2 \in V$, $e = \{v_1, v_2\} \in E$. The graph G is said to be *undirected* if the elements in its edge set have no orientation. That is the edge $e = \{v_1, v_2\} = \{v_2, v_1\}$.

Definition 1.1.13. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the graph G_1 is said to be

- a *subgraph* of $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$, *i.e.* G_1 can be obtained from G_2 by deleting some vertices (with all incident edges) and some edges.
- a *spanning subgraph* of G_2 if $V_1 = V_2$, *i.e.* G_1 can be obtained from G_2 by deleting some edges but not vertices;
- an *induced subgraph* of G_2 if every edge of G_2 with both end points in V_1 is also an edge of G_1,*i.e.* G_1 can be obtained from G_2 by deleting some vertices (with all incident edges).

Definition 1.1.14. (Weighted Graphs) A graph $G = (V, E)$ together with a *weight function*

$$w : V \times V \to \mathbb{R}^+$$

which assigns a non-negative real weight $w(i, j)$ to each pair i, j of vertices with w satisfying the following properties:

- $w(i, j) \geq 0 \forall i, j \in V$
- $w(i, j) = w(j, i)$.

Definition 1.1.15 (Connected Graph)**.** A graph $G = (V, E)$ is said to be *connected* if every pair of distinct vertices is joined by a path. Otherwise, the graph is said to be *disconnected*.

Definition 1.1.16. (Walk) A *walk* is defined as a finite alternating sequence of vertices and edges, beginning and ending with vertices, such that each edge is incident to the vertices preceding and following it with no edge traversed more than once.

It is possible for a walk to begin and end at the same vertex. Such a walk is known as a *closed walk*. A walk that is not closed is known as an *open walk*.

Definition 1.1.17. An open walk in which no vertex appears more than once is called a *path*.

Definition 1.1.18. Let $\{X_k\}_{k=1}^{\infty}$ be a sequence of independent, identically distributed discrete random variables. For every positive integer m, we let S_m denote $\sum_{i=1}^{m} X_i$. The sequence $\{S_m\}_{m=1}^{\infty}$ is called a *random walk*

1.2 Matrix Representation of Graphs

Although a pictorial representation of a graph is very convenient for a visual study, other representations are better for computer processing. A matrix is a convenient and useful way of representing a graph to a computer. Matrices lend themselves easily to mechanical manipulations. Again, many known results in matrix algebra can be readily applied to study the structural properties of a graph from the algebraic viewpoint. In many applications of graph theory, such as in image segmentation, matrices also turn out to be the natural way of expressing the problem. In this section we review some of the matrix representations of a graph needed for our work.

1.2.1 Incidence Matrix

Let $G = (V, E)$ be a graph with n vertices, e edges and no self-loops. Define an n by e matrix $A = [a_{ij}]$ whose n rows correspond to the n vertices and the e rows correspond to the e edges, as follows:

The matrix element

$$a_{ij} = \begin{cases} 1, & \text{if the } j\text{th edge } e_j \text{ is incident in the } i\text{th vertex } v_i. \\ 0, & \text{otherwise.} \end{cases} \tag{1.1}$$

Such a matrix A is called the *vertex − edge incidence matrix*, or simply *incidence matrix*. The matrix A for a graph G is sometimes also written as $A(G)$.
Given any geometric representation of a graph without self-loops, we can readily write an incidence matrix. On the other hand, if we are given an incidence matrix $A(G)$, we can construct its geometric graph without ambiguity.
The following observations about the *incidence matrix* A can be readily made:

- Since every edge is incident on exactly two vertices, each column of A has exactly two 1's and it follows that the column sum is 2.

- The number of 1's in each row equals the degree of the corresponding vertex. That is the row sum equals the degree of the corresponding vertex.
- A row with all zeros therefore represents an isolated vertex.
- Parallel edges in a graph produce identical columns in its incidence matrix.
- If a graph G is disconnected and consists of components g_1 and g_2, the incidence matrix $A(G)$ of graph G can be written in block-diagonal form as follows.

$$A(G) = \begin{bmatrix} A(g_1) & 0 \\ 0 & A(g_2) \end{bmatrix},$$

  where $A(g_1)$ and $A(g_2)$ are incidence matrices of components g_1 and g_2. This observation results from the fact that no *edge* in g_1 is incident to a vertex in g_2, and vice versa. It also holds for disconnected graphs with any number of components.
- Permutation of any two rows or columns of an incidence matrix simply corresponds to the relabelling of the vertices and edges of the graph.

1.2.2 Adjacency Matrix

As an alternative to the *incidence* matrix, it is sometimes more convenient to represent a graph by its *adjacency matrix* or *connection matrix*. The *adjacency* matrix of a graph G with n vertices and no parallel edges is an n by n symmetric binary matrix $X = [x_{ij}]$ defined over the ring of integers such that:

The matrix element

$$x_{ij} = \begin{cases} 1, & \text{if there is an edge between the } i\text{th and } j\text{th vertices, and .} \\ 0, & \text{if there is no edge between them.} \end{cases} \tag{1.2}$$

Observations that can be made immediately about the adjacency matrix X of a graph G are:

- The entries along the principal diagonal of X are all 0's if and only if the graph has no self loops. A self-loop at the ith vertex corresponds to $x_{ii} = 1$.

- The definition of adjacency matrix makes no provision for parallel edges. Adjacency matrices are therefore defined for graphs without parallel edges [5].
- If a graph is *simple* (*i.e.* no self-loops or parallel edges), the degree of a vertex equals the row or column sum of X.
- Permutations of rows and of corresponding columns imply reordering of vertices making sure that the rows and columns must be arranged in the same order.
- A graph G is disconnected and is in two components g_1 and g_2 if and only if its adjacency matrix $X(G)$ can be partitioned as

$$X(G) = \begin{bmatrix} X(g_1) & 0 \\ 0 & X(g_2) \end{bmatrix},$$

where $X(g_1)$ and $X(g_2)$ are the adjacency matrices of components g_1 and g_2 respectively.

Remark. In defining the *adjacency matrix* of a graph, we can use the weight of each edge between the nodes i and j in the case of the weighed graph. We let W denote the weighted adjacency matrix.

1.2.3 Degree Matrix

Let $G = (V, E)$ be a graph with n vertices, e edges. Define an $n \times n$ matrix $D = [d_{ij}]$ whose n rows and columns correspond to the n vertices as follows:

$$d_{ij} = \begin{cases} d_i, & \text{if } i = j. \\ 0, & \text{otherwise.} \end{cases} \tag{1.3}$$

Matrix D is known as the *degree matrix* which is a diagonal matrix whose (i, i)-th entry is d_i, the degree of vertex i. We can also generalize the definition of the degree matrix to make room for the case of a weighted graph. We define :

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

1.2.4 Laplacian Matrix

In a graph G, let d_i denote the degree of vertex i. We define the Laplacian for graphs without *self-loops* or *parallel edges*. Consider the matrix $L = [l_{ij}]$ defined as follows:

$$l_{ij} = \begin{cases} d_i, & \text{if i = j.} \\ -1, & \text{if the } i\text{th vertex is adjacent to the } j\text{th vertex.} \\ 0, & \text{otherwise.} \end{cases} \tag{1.4}$$

This matrix is known as the *Laplacian matrix*. Now let D denote the degree matrix whose (i,i)-th entry has the value d_i, the degree of vertex i. The *normalised Laplacian* of G is defined to be the matrix $\mathcal{L} = [\ell_{ij}]$ with entries:

$$\ell_{ij} = \begin{cases} 1, & \text{if } i = j \text{ and } d_i \neq 0. \\ -\frac{1}{\sqrt{d_i d_j}}, & \text{if the } i\text{th vertex is adjacent to the } j\text{th vertex.} \\ 0, & \text{otherwise.} \end{cases} \tag{1.5}$$

Notice that we can write

$$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

with the convention that $D^{-1}(i,i) = 0$ for $d_i = 0$. We say that i is an isolated vertex if $d_i = 0$.

Example 1.2.5. We compute the adjacency, incidence and Laplacian matrix of the graph in Figure 1.2.

- The incidence I and the adjacency A matrices of the graph are given as follows:

$$I = \left[\begin{array}{cccccccccc:c} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 4 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right] \quad A = \left[\begin{array}{ccccccccc:c} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 4 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hdashline 2 & 4 & 2 & 4 & 2 & 1 & 1 & 3 & 1 & \end{array}\right]$$

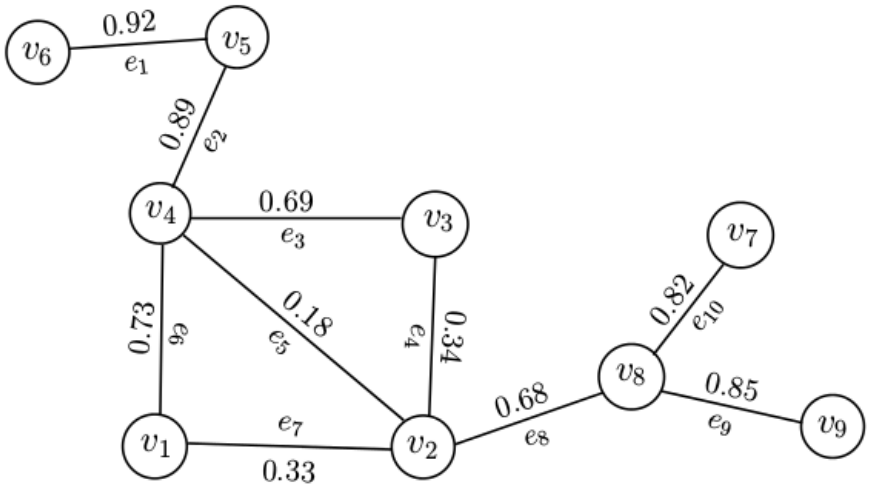


Figure 1.2: A graph on nine vertices

- The degree D and the Laplacian L matrices of the graph are given as follows:

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad L = \left[\begin{array}{ccccccccc|c} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \end{array}\right]$$

As stated earlier, the *adjacency* matrix and the *diagonal* matrix can be defined in terms of the edge weights. We can therefore write the *weighted* adjacency matrix W, the diagonal matrix D and the corresponding Laplacian matrix L as :

$$W = \begin{bmatrix} 0 & 0.33 & 0 & 0.73 & 0 & 0 & 0 & 0 & 0 \\ 0.33 & 0 & 0.34 & 0.18 & 0 & 0 & 0 & 0.68 & 0 \\ 0 & 0.34 & 0 & 0.69 & 0 & 0 & 0 & 0 & 0 \\ 0.73 & 0.18 & 0.69 & 0 & 0.89 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.89 & 0 & 0.92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.92 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.82 & 0 \\ 0 & 0.68 & 0 & 0 & 0 & 0 & 0.82 & 0 & 0.85 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 1.06 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.53 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.03 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.49 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.82 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.35 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 \end{bmatrix}$$

$$L = \begin{bmatrix} 1.06 & -0.33 & 0 & -0.73 & 0 & 0 & 0 & 0 & 0 \\ -0.33 & 1.53 & -0.34 & -0.18 & 0 & 0 & 0 & -0.68 & 0 \\ 0 & -0.34 & 1.03 & -0.69 & 0 & 0 & 0 & 0 & 0 \\ -0.73 & -0.18 & -0.69 & 2.49 & -0.89 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.89 & 1.81 & -0.92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.92 & 0.92 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.82 & -0.82 & 0 \\ 0 & -0.68 & 0 & 0 & 0 & 0 & -0.82 & 2.35 & -0.85 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.85 & 0.85 \end{bmatrix}$$

Chapter 2

Matrix Algrebra and Analysis

2.1 Linear Algebra Background

This thesis is largely about linear trends in data, exhibited when the data is cast as a matrix. As such, we now review some basics of linear algebra and develop the tools we will use in subsequent chapters. We begin with some notational and introductory material focusing on spectral features of matrices.

2.1.1 Basic Definitions

We refer to matrices frequently in this thesis, as they are the primary objects of interest, and so we use capital letters M to denote them. Single subscripts (M_j) denote column j in the matrix M, and double subscripts (M_{ij}) denote the entry in row i and column j of M. All matrices we consider will be of dimension $m \times n$, with m rows and n columns. In general m need not equal n, though in some special cases (which will be noted) the matrix may be square, and even symmetric.

2.2 Eigen Decomposition

Eigen-decomposiiton of a matrix M is defined as the factorization of a matrix into its canonical form. That is the matrix is represented in terms of its *eigenvalues* and *eigenvectors*.

Only diagonalizable matrices can be factorized this way.

Definition 2.2.1 (Similar Matrix)**.** Two matrices M and N are said to be *similar* if there exists an invertible matrix T (known as the similarity transformation matrix) such that $M = T^{-1}NT$

Definition 2.2.2 (Diagonalizable Matrix)**.** A matrix M is *diagonalizable* if it is similar to a diagonal matrix.

If $x = a + ib$ is a complex number, then we let $x^* = a - ib$ denote its *conjugate*.
If $M \in \mathbb{C}^{n\times n}$ is a square matrix, $\lambda \in \mathbb{C}$ is a scalar, $\mathbf{v} \in \mathbb{C}^n - \{0\}$ is a non zero vector and we have

$$M\mathbf{v} = \lambda\mathbf{v} \tag{2.1}$$

then we say that λ is an *eigenvalue* of M and $\mathbf{v}$ is the *eigenvector* of M corresponding to λ.

When 2.1 is satisfied, then we equivalently have

$$(M - \lambda I).\mathbf{v} = 0$$

for a non-zero vector $\mathbf{v}$, which is equivalent to

$$det(M - \lambda I) = 0. \tag{2.2}$$

The *eigenvalues* of $M \in \mathbb{C}^{n\times n}$ are the zeros of the *characteristic polynomial*

$$p(\lambda) = det(M - \lambda I)$$

and we denote the set of *eigenvectors* of the matrix M by

$$\lambda(M) = \{\lambda : det(M - \lambda I) = 0\}$$

Remark. We note that the *characteristic polynomials* of similar matrices are the same and therefore have exactly the same *eigenvalues*.

Definition 2.2.3. *A matrix $M \in \mathbb{C}^{n\times n}$ is Hermitian if $M_{ij} = M^*_{ji}$ for every i, j.*

We note here that a real symmetric matrix is always Hermitian.

Lemma 2.2.4. *If a matrix M is Hermitian, then all the eigenvalues of M are real.*

Proof. Let M be a Hermitian matrix and let λ be a scalar and x be a non-zero vector such that $Mx = \lambda x$. We will show that $\lambda = \lambda^*$, which implies that λ is a real number.
We start off by defining the following inner product operation over vectors in $\mathbb{C}^n$:

$$\langle \mathbf{v}, \mathbf{w} \rangle := \sum_i v_i^*.w_i.$$

Notice that, by definition, we have $\langle v, w \rangle = (\langle w, v \rangle)^*$ and $\langle v, v \rangle = \|v\|^2$. The lemma follows by observing that:

$$\begin{aligned} \langle M\mathbf{x}, \mathbf{x} \rangle &= \sum_i \sum_j M_{ij}^* x_j^* x_i \\ &= \sum_i \sum_j M_{ji} x_j x_i^* \\ &= \langle \mathbf{x}, \mathbf{x}M \rangle \end{aligned}$$

where we use the fact that M is Hermitian, and that

$$\langle M\mathbf{x}, \mathbf{x} \rangle = \langle \lambda\mathbf{x}, \mathbf{x} \rangle = \lambda^* \|x\|^2$$

and

$$\langle \mathbf{x}, \mathbf{x}M \rangle = \langle \mathbf{x}, \lambda\mathbf{x} \rangle = \lambda \|x\|^2$$

so that $\lambda = \lambda^*$. □

In order to relate the eigenvalues of the adjacency matrix of the graph to its combinatorial properties, we first need to express the eigenvalues and eigenvectors as solutions to optimization problems, rather than solutions to algebraic equations.
First, we observe that if a matrix M is a real symmetric matrix and λ is an eigenvalue of M, then λ admits a real eigenvector. This is because if $M\mathbf{x} = \lambda\mathbf{x}$ for some $\mathbf{x} \in C^n$, then we also have $M\mathbf{x}' = \lambda\mathbf{x}'$ where $\mathbf{x}' \in \mathbb{R}^n$ is a vector whose i-th coordinate is the real part of the i-th coordinate of $\mathbf{x}$. Now, if λ is a (real) eigenvalue of a real symmetric matrix M, then the set $\{\mathbf{x} \in \mathbb{R}^n : M\mathbf{x} = \lambda\mathbf{x}\}$ is a vector space of $\mathbb{R}^n$, called the *eigenspace* of λ.

Lemma 2.2.5. *If $\lambda \neq \lambda'$ are two distinct eigenvalues of a real symmetric matrix M, then the eigenspaces of λ and λ' are orthogonal.*

Proof. Let $\mathbf{x}$ be an eigenvector of λ and $\mathbf{y}$ be an eigenvector of λ'. From the symmetry of M and the fact that M,$\mathbf{x}$ and $\mathbf{y}$ all have real entries, we get

$$\langle M\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, M\mathbf{y}\rangle$$

but

$$\langle M\mathbf{x}, \mathbf{y}\rangle = \lambda.\langle \mathbf{x}, \mathbf{y}\rangle$$

and

$$\langle \mathbf{x}, M\mathbf{y}\rangle = \lambda'.\langle \mathbf{x}, \mathbf{y}\rangle$$

so that

$$(\lambda - \lambda').\langle \mathbf{x}, \mathbf{y}\rangle = 0$$

which implies that $\langle \mathbf{x}, \mathbf{y}\rangle = 0$, since $\lambda \neq \lambda'$ *i.e.*, $\mathbf{x}$ and $\mathbf{y}$ are orthogonal. □

If $M \in \mathbb{C}^{n\times n}$ has n independent *eigenvetors* $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ and $M\mathbf{v_i} = \lambda_i \mathbf{v_i}$, for $1 \leq i \leq n$, then M is *diagonalizable*. We see that if

$$V = [\mathbf{v_1}|\mathbf{v_2}|\ldots|\mathbf{v_n}]$$

then

$$V^{-1}MV = diag(\lambda_1, \lambda_2, \ldots \lambda_n).$$

It is important to note that not all matrices are *diagonlizable*. If $M \in \mathbb{C}^{n\times n}$ is Hermitian, then there exists a unitary matrix U such that U^*MU is diagonal. If M is a real symmetric matrix, then U may be chosen to be real and orthogonal.

If a matrix $M \in \mathbb{C}^{n\times n}$ is symmetric however, there exists an orthogonal P such that

$$P^TMP = diag(\lambda_1, \lambda_2, \ldots \lambda_n).$$

This is known as the *Shur decomposition*.

2.2.6 Rayleigh Quotient and the Min-Max Theorem

As we saw earlier, Hermitian matrices have real *eigenvalues* and they are *diagonalizable*. We review yet another important property of Hermitian eigenproblems: they obey the **min-max theorem** or the **variational principle**, which says that the eigensolutions *maximize* and *minimize* a certain quantity. This is useful because it gives an intuitive way of "guessing"

the qualitative features of the eigensolutions.

The Rayleigh Quotient

Definition 2.2.7 (Rayleigh Quotient)**.** For a real symmetric matrix M, the Rayleigh quotient of M, denoted as $R_M(.)$ is a function from $\mathbb{R}^n\backslash\{0\}$ to $\mathbb{R}$, defined as follows

$$R_M(\mathbf{v}) = \frac{\mathbf{v}^T M \mathbf{v}}{\mathbf{v}^T \mathbf{v}}.$$

We can show that the largest and smallest *eigenvalues* of a symmetric matrix M satisfy

$$\lambda_{max} = \max_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T M \mathbf{v}}{\mathbf{v}^T \mathbf{v}}$$

and

$$\lambda_{min} = \min_{\mathbf{v}\neq 0} \frac{\mathbf{v}^T M \mathbf{v}}{\mathbf{v}^T \mathbf{v}}$$

First of all, that if $\mathbf{v_i}$ is the eigenvector corresponding to the eigenvalue λ_i, then

$$\lambda_i = R_M(\mathbf{v_i}) = \frac{\mathbf{v_i}^T M \mathbf{v_i}}{\mathbf{v_i}^T \mathbf{v_i}}$$

For any $\mathbf{v} \in \mathbb{R}^n\backslash\{0\}$, $\mathbf{v}$ can be expressed in the orthonormal basis given by the eigenvectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ of M. Let $\mathbf{v} = \sum_i \alpha_i \mathbf{v_i}$ then

$$\begin{aligned} \mathbf{v}^T \mathbf{v} &= \left(\sum_{i=1}^{n} \alpha_i \mathbf{v_i}\right)^T \left(\sum_{i=1}^{n} \alpha_i \mathbf{v_i}\right) \\ &= \sum_{i,j} \alpha_i \alpha_j \mathbf{v_i}^T \mathbf{v_i} \\ &= \sum_{i=1}^{n} {\alpha_i}^2 \end{aligned} \tag{2.3}$$

Similarly,

$$\begin{aligned}\mathbf{v}^T M\mathbf{v} &= \left(\sum_{i=1}^{n}\alpha_i\mathbf{v_i}\right)^T\left(\sum_{i=1}^{n}\alpha_i M\mathbf{v_i}\right)\\ &= \left(\sum_{i=1}^{n}\alpha_i\mathbf{v_i}\right)^T\left(\sum_{i=1}^{n}\alpha_i\lambda_i\mathbf{v_i}\right)\\ &= \sum_{i=1}^{n}{\alpha_i}^2.\lambda_i. \qquad (2.4)\end{aligned}$$

It follows therefore that

$$R_M(\mathbf{v}) = \frac{\sum_{i=1}^{n}{\alpha_i}^2.\lambda_i}{\sum_{i=1}^{n}{\alpha_i}^2}.$$

Using the fact that $\lambda_i \leq \lambda_{max}$, we have that $R_M(\mathbf{v}) \leq \lambda_{max}$ for all v. Combining it with $R_M(\mathbf{v_1}) = \lambda_{max}$ and thus we get $\lambda_{max} = \max_{\mathbf{v}\neq 0}\frac{\mathbf{v}^T M\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$.

Similarly, we can observe that

$$\lambda_2 = \max_{\mathbf{v}^T\mathbf{v}=0}\frac{\mathbf{v}^T M\mathbf{v}}{\mathbf{v}^T\mathbf{v}}.$$

The Courant-Fischer theorem gives a variational formulation of the eigenvalues of a symmetric matrix, which can be useful for obtaining bounds on the eigevalues. (see [7]).

Theorem 2.2.8 (Courant-Fischer Minmax Theorem)**.** *Let M be an $n \times n$ real symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$ and corresponding eigenvectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$, then*

$$\lambda_k(M) = \max_{\dim(\mathbf{S})=k}\min_{0\neq\mathbf{y}\in\mathbf{S}}\frac{\mathbf{y}^T M\mathbf{y}}{\mathbf{y}^T\mathbf{y}}$$

for $1 \leq k \leq n$.

Proof. Let $Q^T MQ = diag(\lambda_i)$ be the *Shur decomposition* with $\lambda_k = \lambda_k(M)$ and $Q = [\mathbf{q_1}|\ldots|\mathbf{q_n}]$. Define

$$S_k = span\{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_k}\},$$

$$\max_{\dim(\mathbf{S})=k}\min_{0\neq\mathbf{y}\in\mathbf{S}}\frac{\mathbf{y}^T M\mathbf{y}}{\mathbf{y}^T\mathbf{y}} \geq \min_{0\neq\mathbf{y}\in\mathbf{S}}\frac{\mathbf{y}^T M\mathbf{y}}{\mathbf{y}^T\mathbf{y}} = \mathbf{q_k}^T M\mathbf{q_k} = \lambda_k.$$

Conversely, let S be any $k-$dimensional subspace that intersects $span\{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_n}\}$, a

subspace with dimension $n-k+1$. If $\mathbf{y}_* = \alpha_k \mathbf{q_k} + \ldots + \alpha_n \mathbf{q_n}$ is this intersection, then

$$\min_{0 \neq \mathbf{y} \in \mathbf{S}} \frac{\mathbf{y}^T M \mathbf{y}}{\mathbf{y}^T \mathbf{y}} \leq \frac{{\mathbf{y}_*}^T M \mathbf{y}_*}{{\mathbf{y}_*}^T \mathbf{y}_*} \leq \lambda_k(M)$$

Since this inequality holds for all $k-$dimensional subspaces,

$$\max_{\dim(\mathbf{S})=k} \min_{0 \neq \mathbf{y} \in \mathbf{S}} \frac{\mathbf{y}^T M \mathbf{y}}{\mathbf{y}^T \mathbf{y}} \leq \lambda_k(M),$$

which completes the proof. □

2.2.9 The symmetric Lanczos process

Suppose $A \in \mathbb{R}^{n \times n}$ is sparse, large, and symmetric and assume that a few of its largest and/or smallest eigenvalues are desired. Eigenvalues at either end of the spectrum are referred to as extremal eigenvalues. This problem can be addressed by a method attributed to Lanczos (1950).
The method generates a sequence of tridiagonal matrices $\{T_k\}$ with the property that the extremal eigenvalues of $T_k \in \mathbb{R}^{k \times k}$ are progressively better estimates of $A's$ extremal eigenvalues. The Lanczos method "learns from experience" and takes advantage of all previously computed matrix-vector products.

The symmetric Lanczos algorithm is formulated as follows (See [7]).
Given a symmetric matrix $A \in \mathbb{R}^{n \times n}$ and a unit 2-norm vector $q_1 \in \mathbb{R}^n$, the following algorithm computes a matrix $Q_k = [q_1|...|q_k]$ with orthonormal columns and a tridiagonal matrix $T_k \in \mathbb{R}^{k \times k}$ so that $AQ_k = Q_k T_k$. The diagonal and superdiagonal entries of T_k are $\alpha_1, ..., \alpha_k$ and $\beta_1, ..., \beta_{k-1}$ respectively. The integer k satisfies $1 \leq k \leq n$.

There is no loss of generality in choosing β_k to be positive. The vectors q_k are called *Lanczos vectors*.

With careful overwriting of algorithm 1 and exploitation of the formula

$$\alpha_k = q_k^T (A q_k - \beta_{k-1 q_{k-1}}),$$

Algorithm 1 Lanczos Algorithm

$k = 0, \beta_0 = 1, q_0 = 0, r_0 = q_1$

while $k = 0$ or $\beta_k \neq 0$ **do**

  $q_{k+1} = r_k/\beta_k$

  $k = k + 1$

  $\alpha_k = q_k^T A q_k$

  $r_k = (A - \alpha_k I)q_k - \beta_{k-1} q_{k-1}$

  $\beta_k = \|r_k\|_2$

end while

the whole Lanczos process 2 can be implemented with just a pair of n-vectors:

$$w = q_1, v = Aw, \alpha_1 = w^T v, v = v - \alpha_1 w, \beta_1 = \|v\|_2, k = 1$$

Algorithm 2 Lanczos Algorithm Refined

while $\beta_k \neq 0$ **do**

  for $i \leftarrow 1 : n$ **do**

    $t = w_i, w_i = v_i/\beta_k, v_i = -\beta_k t$

  end for

  $v = v + Aw$

  $k = k + 1, \alpha_k = w^T v, v = v - \alpha_k w, \beta_1 = \|v\|_2$

end while

At the end of the loop body, the array w houses Q_k and v houses the residual vector $r_k = Aq_k - \alpha_k q_k - \beta k - 1 q_{k-1}$.

Chapter 3

Spectral Theory and Clustering

As discussed in section 2 of chapter 1, given a graph $G = (V, E)$ there are various matrix representations of G. Two very relevant matrix representation of graphs to our study are the adjacency matrix A and the Laplacian matrix L. The eigenvalues and eigenvectors of these matrices expose the properties of the graph under study thereby making it a convenient tool for studying graphs. The study of the properties of a graph in relation to the eigenvalues and eigenvectors of matrices associated to the graph is know as *spectral graph theory* and the use of the spectrum of the graph Laplacian to partition a graph is known as *spectral clustering*. In this chapter we examine the Laplacian matrix and its properties and apply it to graph partitioning.

3.1 Graph Laplacians and their basic properties

The Laplacian matrices of graphs are the main tools used for spectral partitioning. In this section, we will define different graph Laplacians and examine their properties to the degree required for our work. It is therefore important to note that there are different graph Laplacians in the literature with no unique convention as to which one is known as the "graph Laplacian".

We will work with the assumption that the graph $G = (V, E)$ is a weighted, undirected graph whose weight matrix is given by $W = [w_{ij}]$ where $w_{ij} = w_{ji} \geq 0$.

We will not necessarily assume that the eigenvectors of the matrices are normalized. For instance the constant vector $\mathbf{1}$ and a multiple a$\mathbf{1}$ for some $a \neq 0$ shall be considered as one and the same. By the k-th eigenvector, we are referring to the eigenvector corresponding to the k-th eigenvalue where the k-th eigenvalue is the eigenvalue in the k-th position when the

eigenvalues are arranged in ascending order of magnitude respecting multiplicity.

3.1.1 The unnormalized graph Laplacian

The unnormalized graph Laplacian matrix is defined as

$$L = D - W$$

where D is the degree matrix given by $D = [d_{ij}] = [d_{ii}]$ and W is the weight matrix given by $W = [w_{ij}]$ with $w_{ji} = w_{ji}$ We summarize the most important facts about the Laplacian matrix needed for spectral partitioning.

Lemma 3.1.2. *Let L be the Laplacian matrix. For every vector $u \in \mathbb{R}^n$ we have*

$$u^T Lu = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2.$$

Proof. From the definition of the degree of a vertex $v_i \in V$, $d_i = \sum_{j=1}^{n} w_{ij}$

$$\begin{aligned} u^T Lu &= u^T Du - u^T Wu \\ &= \sum_{i=1}^{n} d_i {u_i}^2 - \sum_{i,j=1}^{n} w_{ij} u_i u_j \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} d_i {u_i}^2 - 2 \sum_{i,j=1}^{n} w_{ij} u_i u_j + \sum_{j=1}^{n} d_j {u_j}^2 \right) \\ &= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2. \end{aligned}$$

□

Lemma 3.1.3. *The Laplacian matrix L is positive semi-definite and symmetric.*

Proof. Since both W and D are symmetric, $L = D - W$ is also symmetric.
Also since $w_{ij} \geq 0$ and $(u_i - u_j)^2 \geq 0 \ \forall i, j$, it follows that $u^T Lu \geq 0 \ \forall u \in \mathbb{R}^n$. Therefore L is positive semi-definite. □

Lemma 3.1.4. *0 is the smallest eigenvalue of L and the corresponding eigenvector is the constant one vector* $\mathbf{1}$.

Proof. $d_i = \sum_j w_{ij}$

$$L\mathbf{1} = \begin{pmatrix} d_{11} - w_{11} & -w_{12} & \dots & -w_{1n} \\ -w_{21} & d_{22} - w_{22} & \dots & -w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -w_{n1} & -w_{n2} & \dots & d_{nn} - w_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

□

Lemma 3.1.5. *L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.*

Proof. The proof follows directly from the previous Lemmas. □

Note that the unnormalized graph Laplacian does not depend on the diagonal elements of the adjacency matrix W. Each adjacency matrix which coincides with W on all off-diagonal positions leads to the same unnormalized graph Laplacian L. In particular, self-edges in a graph do not change the corresponding graph Laplacian.

The unnormalized graph Laplacian and its eigenvalues and eigenvectors can be used to describe many properties of graphs, see [10]. One example which will be important for spectral clustering is the following, see [9]:

Proposition 3.1.6. *(Number of connected components and the spectrum of L).*
Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$ of those components.

Proof. We start with the case $k = 1$, that is the graph is connected. Assume that u is an eigenvector with eigenvalue 0. Then we know that

$$0 = u^T L u = \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2.$$

As the weights w_{ij} are non-negative, this sum can only vanish if all terms $w_{ij}(u_i - u_j)^2$ go to zero. Thus, if two vertices v_i and v_j are connected (i.e., $w_{ij} > 0$), then u_i needs to equal u_j . With this argument we can see that u needs to be constant for all vertices which can be connected by a path in the graph. Moreover, as all vertices of a connected component

in an undirected graph can be connected by a path, u needs to be constant on the whole connected component. In a graph consisting of only one connected component we thus only have the constant one vector $\mathbf{1}$ as eigenvector with eigenvalue 0, which obviously is the indicator vector of the connected component.

Now consider the case of k-connected components. Without loss of generality we assume that the vertices are ordered according to the connected components they belong to. In this case, the adjacency matrix W has a block diagonal form, and the same is true for the matrix L:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_n \end{pmatrix}$$

Note that each of the blocks L_i is a proper graph Laplacian on its own, namely the Laplacian corresponding to the subgraph of the i-th connected component. As it is the case for all block diagonal matrices, we know that the spectrum of L is given by the union of the spectra of L_i, and the corresponding eigenvectors of L are the eigenvectors of L_i, filled with 0 at the positions of the other blocks. As each L_i is a graph Laplacian of a connected graph, we know that every L_i has eigenvalue 0 with multiplicity 1, and the corresponding eigenvector is the constant one vector on the i-th connected component. Thus, the matrix L has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components. □

3.1.7 The normalized graph Laplacian

There are two matrices which are called normalized graph Laplacians in the literature. Both matrices are closely related to each other and are defined as:

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

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

We denote the first matrix by L_{sym} as it is a symmetric matrix, and the second one by L_{rw} as it is closely related to a random walk. In the following we summarize several properties of L_{sym} and L_{rw}. The standard reference for normalized graph Laplacians is [3].

The normalized Laplacians (L_{sym} and L_{rw}) satisfy the following properties:

Lemma 3.1.8. *For every vector $u \in \mathbb{R}^n$, it follows that*

$$u^T L_{sym} u = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (\frac{u_i}{\sqrt{d_i}} - \frac{u_j}{\sqrt{d_i}})^2.$$

Proof. The proof of this lemma is similar to the one proved in 3.1.2 □

Lemma 3.1.9. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvalue of L_{sym} with eigenvector $w = D^{\frac{1}{2}} u$.*

Proof. Substituting $u = D^{-\frac{1}{2}} w$ and left multiplying the equation $L_{sym} w = \lambda w$ with $D^{-\frac{1}{2}}$ yields the needed results. □

Lemma 3.1.10. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigenproblem $Lu = \lambda Du$.*

Proof. The proof of this follows directly by multiplying the eigenvalue equation $L_{rw} u = \lambda u$ with D from the left. □

Lemma 3.1.11. *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 $D^{\frac{1}{2}} \mathbf{1}$.*

Proof. The first statement is obvious as $L_{rw} \mathbf{1} = 0$, the second statement follows from 3.1.9 □

Lemma 3.1.12. *L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.*

Proof. The statement about L_{sym} follows from 3.1.8, and then the statement about L_{rw} follows from 3.1.9. □

As it is the case for the unnormalized graph Laplacian, the multiplicity of the eigenvalue 0 of the normalized graph Laplacian is related to the number of connected components (see [9]):

Proposition 3.1.13. *(Number of connected components and spectra of L_{sym} and L_{rw}) Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of both L_{rw} and L_{sym} equals the number of connected components $A_1, \ldots, A_k$ 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. The proof is analogous to the one of *Proposition* 3.1.6. □

3.1.14 Spectral Clustering Algorithm

Now we state the spectral clustering algorithm by Shi and Malik [14] known as the Normalized spectral clustering algorithm. We assume that our data consists of n "points" $x_1, \ldots, x_n$ which can be arbitrary objects.

We measure their pairwise 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 $S = (s_{ij})_{i,j=1\ldots n}$.

Algorithm 3 Normalized spectral clustering algorithm

procedure NCUT(W, k) ▷ Take a weight matrix and cut the graph into k parts
  Construct diagonal matrix D
  $L \leftarrow D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ ▷ Compute the normalized Laplacian L
  Compute top k eigenvectors $v_1, \ldots, v_k$ of L
  Form matrix $V \in \mathbb{R}^{n \times k}$ containing the vectors $v_1, \ldots, v_k$ as columns
  for $i \leftarrow 1 : n$ **do**
    $y_{ij} = V_{i,j} / \|V\{i, :\}\|$. Let $y_i \in \mathbb{R}^k$ be the normalized vector corresponding to the i-th row of V
  end for
  Cluster the points $(y_i) i = 1, ..., n \in \mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
  return Clusters $A_1, \ldots, A_k$ with $A_i = \{j | y_j \in C_i\}$.
end procedure

Chapter 4

Graph Partitioning

When modeling an application problem, computer scientists frequently use graphs as abstractions of the problem. One of the fundamental algorithmic operations performed on the graph is to cut it into smaller pieces. Even in instances where the final application concerns a different problem (such as traversals, finding paths. trees and flows), in order to reduce the complexity of the task at hand, the partitioning of large graphs becomes an important sub-task.
Graph partitioning has gained a lot of importance over the years due to its wide range of applications across multiple domains. Some of the applications of graph partitioning include:

- Scientific simulations.
- Social network analysis.
- Very Large Scale Integration (VLSI) circuit partitioning and
- Image Segmentation.

In this chapter, we review the graph partitioning problem and examine the $Ncut$ solution to this problem based on the work of Shi and Malik [14].

4.1 Graph partitioning concept

Given a graph $G = (V, E)$, the vertex set V can be partitioned into two disjoint subsets A and B such that $A \cup B = V$ and $A \cap B = \emptyset$. This is done by simply removing the edges

joining A and B. The degree of dissimilarity between the two sets can be computed as the total weight of the removed edges. In graph theoretic language, this is known as a *cut* :

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v). \tag{4.1}$$

The optimal bi-partitioning of a graph is the one that minimizes this *cut* value. Although there are an exponential number of such partitions, finding the *minimum cut* of a graph is a well studied problem, and there exist efficient algorithms for solving it [14].

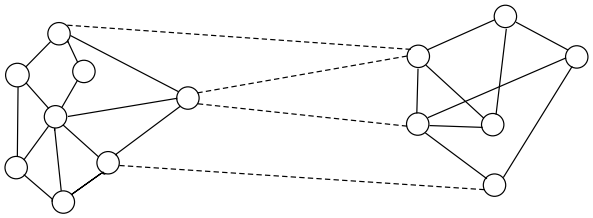


Figure 4.1: The sum of the weights of the dashed-edges represents the cut value between the nodes on the left and on the right

Wu and Leahy[15] proposed a partitioning method based on this minimum cut criterion in which they sought to partition a graph into *k-subgraphs*, such that the maximum cut across the subgroups is minimized. This problem can be efficiently solved by recursively finding the minimum cuts that bisect the existing segments.
They also noticed in their work that, the minimum cut favours cutting small sets of isolated vertices in the graph. This situation is possible since the cut defined in 4.1 increases with the number of edges going across the two partitioned parts. Figure 4.2 illustrates one such instance.

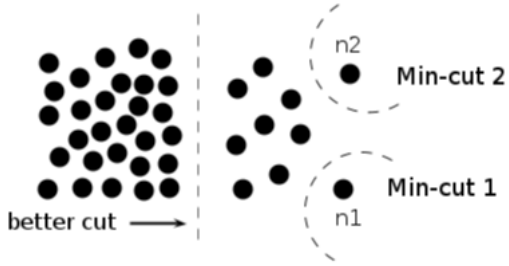


Figure 4.2: A case where minimum cut gives a bad partition

We notice that the minimum cut can produce a bad partition as shown in Figure 4.2. This is due to the fact that it attempts to cut out partitions that will produce small *cut values*. Assuming the edge weights are inversely proportional to the distance between two vertices, we see that the cut that partitions out vertex n_1 and vertex n_2 will be of a very small value.

In order to avoid this unnatural bias for partitioning out a small set of points, it is necessary to define what a good cluster or partition is. A good partition must have the following properties.

1 The total weight of the edges within a partition must be high and

2 the total weight of the edges that connect the partitions must be low

Instead of looking at the the total edge weights connecting the two partitions, Shi and Malik [14] proposed a measure that computes the cut cost as a fraction of the total edge connections to all the nodes in the graph and called this dissociation measure the *noramlized cut* ($Ncut$):

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{4.2}$$

where $assoc(A,V) = \sum_{u\in A, t\in V} w(u,t)$ is the total connections of nodes in A to all nodes in the graph and $assoc(B,V)$ is similarly defined. In this case, the optimal bi-partition is one that minimizes the $Ncut$ value. It is important to note here that the drawback of the $min-cut$ will not be experienced. This is because the cut that partitions out small isolated nodes will no longer have small $Ncut$ value. Notice that the cut value will be a large percentage of the total connections from the small set to all the other nodes.

In the instance illustrated in Figure 4.2, we see that the value of cut_1 across node n_1 will be 100 percent of the total connections from that node. In the same spirit, a measure for total normalized association within groups for a given partition can be defined:

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}, \tag{4.3}$$

where $assoc(A,A)$ and $assoc(B,B)$ are total weights of edges connecting nodes within A and B, respectively. We notice again this is an unbiased measure, which reflects how tightly on average nodes within the group are connected to each other. We also observe a natural relation between the definition of association and dissociation of a partition.

$$
\begin{aligned}
Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\
&= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} \\
&= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \right) \\
&= 2 - Nassoc(A,B).
\end{aligned}
$$

From the properties of a good cut discussed earlier, minimizing the dissociation between the clusters and maximizing the association within the clusters can be satisfied simultaneously. We adopt the normalized cut as the partitioning criterion for our work. Minimizing the normalized cut has been shown to be NP-Complete even for special cases of graphs on grids [14]. Shi and Malik [14] however showed in their work that, when the normalized cut problem is embedded in the real value domain, an approximate discrete solution can be computed efficiently. The proof of the NP-Completeness of the normalized cut problem to Papadimitriou (see [14]), can be found in Appendix A.

4.2 Computing the Optimal Partition

Given a partition of nodes of a graph, V, into two sets A and B, let $\mathbf{x}$ be an $N = |V|$ dimensional indicator vector,

$$
x_i = \begin{cases} 1, & \text{if node } i \text{ is in } A. \\ -1, & \text{otherwise.} \end{cases} \tag{4.4}
$$

Let $d_i = \sum_j w(i,j)$ be the total connection from node i to all other nodes. With the definitions $\mathbf{x}$ and D, we can rewrite $Ncut(A,B)$ as:

$$
\begin{aligned}
Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(B,A)}{assoc(B,V)} \\
&= \frac{\sum_{(x_i>0, x_j<0)} -w_{ij}x_ix_j}{\sum_{x_i>0} d_i} + \frac{\sum_{(x_i<0, x_j>0)} -w_{ij}x_ix_j}{\sum_{x_i<0} d_i}.
\end{aligned}
$$

Let D be an $N \times N$ diagonal matrix with d on its diagonal, W be an $N \times N$ symmetrical matrix with $W(i,j) = w_{ij}$,

$$k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$$

and $\mathbf{1}$ be an $N \times 1$ vector of all ones. Using the fact that $\frac{\mathbf{1}+\mathbf{x}}{2}$ and $\frac{\mathbf{1}-\mathbf{x}}{2}$ are indicator vectors for $x_i > 0$ and $x_i < 0$ respectively, we can rewrite $4[Ncut(\mathbf{x})]$ as:

$$\begin{aligned} 4[Ncut(\mathbf{x})] &= \frac{(\mathbf{1}+\mathbf{x})^T(D-W)(\mathbf{1}+\mathbf{x})}{k\mathbf{1}^T D\mathbf{1}} + \frac{(\mathbf{1}-\mathbf{x})^T(D-W)(\mathbf{1}-\mathbf{x})}{(1-k)\mathbf{1}^T D\mathbf{1}} \\ &= \frac{(\mathbf{x}^T(D-W)\mathbf{x} + \mathbf{1}^T(D-W)\mathbf{1})}{k(1-k)\mathbf{1}^T D\mathbf{1}} + \frac{2(1-2k)\mathbf{1}^T(D-W)\mathbf{x}}{k(1-k)\mathbf{1}^T D\mathbf{1}}. \end{aligned}$$

Let $\alpha(\mathbf{x}) = \mathbf{x}^T(D-W)\mathbf{x}$, $\beta(\mathbf{x}) = \mathbf{1}^T(D-W)\mathbf{x}$, $\gamma = \mathbf{1}^T(D-W)\mathbf{1}$, and $M = \mathbf{1}^T D\mathbf{1}$, we can further expand the above equation as:

$$\begin{aligned} 4[Ncut(\mathbf{x})] &= \frac{(\alpha(\mathbf{x})+\gamma) + 2(1-2k)\beta(\mathbf{x})}{k(1-k)M} \\ &= \frac{(\alpha(\mathbf{x})+\gamma) + 2(1-2k)\beta(\mathbf{x})}{k(1-k)M} - \frac{2(\alpha(\mathbf{x})+\gamma)}{M} + \frac{2\alpha(\mathbf{x})}{M} + \frac{2\gamma}{M}. \end{aligned}$$

Dropping the last term, which in this case equals 0, we get

$$\begin{aligned} 4[Ncut(\mathbf{x})] &= \frac{(1-2k+2k^2)(\alpha(\mathbf{x})+\gamma) + 2(1-2k)\beta(\mathbf{x})}{k(1-k)M} - \frac{2(\alpha(\mathbf{x})+\gamma)}{M} + \frac{2\alpha(\mathbf{x})}{M} \\ &= \frac{\frac{(1-2k+2k^2)}{(1-k)^2}(\alpha(\mathbf{x})+\gamma) + \frac{2(1-2k)}{(1-k)^2}\beta(\mathbf{x})}{\frac{k}{1-k}M} + \frac{2\alpha(\mathbf{x})}{M}. \end{aligned}$$

Letting $b = \frac{k}{1-k}$, and since $\gamma = 0$, it follows that:

$$
\begin{aligned}
4[Ncut(\mathbf{x})] &= \frac{(1+b^2)(\alpha(\mathbf{x})+\gamma)+2(1-b^2)\beta(\mathbf{x})}{bM}+\frac{2b\alpha(\mathbf{x})}{bM} \\
&= \frac{(1+b^2)(\alpha(\mathbf{x})+\gamma)}{bM}+\frac{2(1-b^2)\beta(\mathbf{x})}{bM}+\frac{2b\alpha(\mathbf{x})}{bM}-\frac{2b\gamma}{bM} \\
&= \frac{(1+b^2)(\mathbf{x}^T(D-W)\mathbf{x}+\mathbf{1}^T(D-W)\mathbf{1})}{b\mathbf{1}^TD\mathbf{1}}+\frac{2(1-b^2)\mathbf{1}^T(D-W)\mathbf{x}}{b\mathbf{1}^TD\mathbf{1}} \\
&\quad +\frac{2b\mathbf{x}^T(D-W)\mathbf{x}}{b\mathbf{1}^TD\mathbf{1}}-\frac{2b\mathbf{1}^T(D-W)\mathbf{1}}{b\mathbf{1}^TD\mathbf{1}} \\
&= \frac{(\mathbf{1}+\mathbf{x})^T(D-W)(\mathbf{1}+\mathbf{x})}{b\mathbf{1}^TD\mathbf{1}} \\
&\quad +\frac{b^2(\mathbf{1}-\mathbf{x})^T(D-W)(\mathbf{1}-\mathbf{x})}{b\mathbf{1}^TD\mathbf{1}} \\
&\quad -\frac{2b(\mathbf{1}-\mathbf{x})^T(D-W)(\mathbf{1}+\mathbf{x})}{b\mathbf{1}^TD\mathbf{1}} \\
&= \frac{[(\mathbf{1}+\mathbf{x})-b(\mathbf{1}-\mathbf{x})]^T(D-W)[(\mathbf{1}+\mathbf{x})-b(\mathbf{1}-\mathbf{x})]}{b\mathbf{1}^T\mathbf{1}}.
\end{aligned}
$$

Setting $\mathbf{y}=(\mathbf{1}+\mathbf{x})-b(\mathbf{1}-\mathbf{x})$, we see that

$$\mathbf{y}^TD\mathbf{1}=\sum_{x_i>0}d_i-b\sum_{x_i<0}d_i=0 \tag{4.5}$$

since $b=\frac{k}{1-k}=\frac{\sum_{x_i>0}d_i}{\sum_{x_i<0}d_i}$, and

$$
\begin{aligned}
\mathbf{y}^TD\mathbf{1} &= \sum_{x_i>0}d_i+b^2\sum_{x_i<0}d_i \\
&= b\sum_{x_i<0}d_i+b^2\sum_{x_i<0}d_i \\
&= b(\sum_{x_i<0}d_i+b\sum_{x_i<0}d_i) \\
&= b\mathbf{1}^TD\mathbf{1}.
\end{aligned}
$$

Putting everything together we have,

$$min_{\mathbf{x}}Ncut(\mathbf{x})=min_{\mathbf{y}}\frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^TD\mathbf{y}} \tag{4.6}$$

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

Consider the following generalized eigenvalue problem:

$$(D - W)\mathbf{y} = \lambda D\mathbf{y}. \tag{4.7}$$

We realize that equation 4.6 can be minimized by solving 4.7 if $\mathbf{y}$ is relaxed to take on real values.

From equation 4.6, there are two constraints on $\mathbf{y}$ which come from the condition on the corresponding indicator vector $\mathbf{x}$. They are :

1 $\mathbf{y}^T D\mathbf{1} = 0$ and

2 $\mathbf{y}(i) \in \{1, -b\}$.

We transform equation 4.7 to a standard eigensystem. We rewrite 4.7 as:

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z} \tag{4.8}$$

where $\mathbf{z} = D^{\frac{1}{2}}\mathbf{y}$. We see that $\mathbf{z}_0 = D^{\frac{1}{2}}\mathbf{1}$ is an eigenvector of 4.8 with eigenvalue of 0. Furthermore, $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ is symmetric positive semidefinite as we have shown in 3.1.3. Hence, $\mathbf{z}_0$ is, in fact, the smallest eigenvector of 4.8 and all eigenvectors of 4.8 are perpendicular to each other. In particular, $\mathbf{z}_1$, the second smallest eigenvector, is perpendicular to $\mathbf{z}_0$. Translating this statement back into the general eigensystem 4.7, we have:

1. $\mathbf{y}_0 = \mathbf{1}$ is the smallest eigenvector with eigenvalue of 0, and
2. $0 = \mathbf{z}_1^T\mathbf{z}_0 = \mathbf{y}_1^T D\mathbf{1}$ where $\mathbf{y}_1$ is the second smallest eigenvector of 4.7.

Now we recall a simple fact about the *Rayleigh quotient* See [14] and [7]

Theorem 4.2.1. *Let A be a real symmetric matrix. Under the constraint that $\mathbf{x}$ is orthogonal to the j-1 smallest eigenvectors $\mathbf{x_1}, \ldots, \mathbf{x_{j-1}}$, the quotient $\frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$ is minimized by the next smallest eigenvector $\mathbf{x}_j$ and its minimum value is the corresponding eigenvalue λ_j.*

As a result, we obtain:

$$\mathbf{z}_1 = arg.min_{\mathbf{z}^T\mathbf{z}_0=0}\frac{\mathbf{z}^T D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{z}}{\mathbf{z}^T\mathbf{z}}$$

and consequently,

$$\mathbf{y}_1 = arg.min_{\mathbf{y}^T D\mathbf{1}=0} \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}.$$

Thus, the second smallest eigenvector of the generalized eigensystem 4.7 is the real valued solution to the normalized cut problem.

We realize that the second smallest eigenvector of the generalized eigensystem is not necessarily the solution to the original problem. This is due to the fact that the second constraint on $\mathbf{y}$ (that is $\mathbf{y}(i)$ taking on two discrete values) is not automatically satisfied. We therefore relax this constraint.

From 4.2.1, we can generalize the argument that we can subdivide existing graphs by making use of the eigenvector corresponding to the next smallest eigenvalue since it is the real valued solution that optimally subpartitions existing parts.

In practice, approximation error accumulates with every taken eigenvector when the real valued solution is cast in discrete form which consequently results in the unreliability of solutions based on higher eigenvectors. It is therefore best to restart solving the partitioning problem on each subgraph individually. (See [14]).

While the second smallest eigenvector $\mathbf{y}$ of 4.7 only approximates the optimal normalized cut solution, it is interesting to note that it exactly minimizes the following problem:

$$\inf_{\mathbf{y}^T D\mathbf{1}=0} \frac{\sum_i \sum_j (\mathbf{y}(i)-\mathbf{y}(j))^2 w_{ij}}{\sum_i \mathbf{y}(i)^2 d(i)}$$

in real-valued domain, where $d(i) = D(i,i)$. Roughly speaking, this forces the indicator vector $\mathbf{y}$ to take similar values for nodes i and j that are tightly coupled (large w_{ij}). In summary, using the normalized cut criterion for graph partitioning serves as a simple and efficient tool and we have shown how this criterion can be computed efficiently by solving a generalized eigenvalue problem.

4.3 The grouping algorithm

The grouping algorithm consists of the following steps (See [14]):

1. Given an image or image sequence, set up a weighted graph $G = (V, E)$ and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.

2. Solve $(D - W)\mathbf{z} = \lambda D\mathbf{z}$ for eigenvectors with the smallest eigenvalues.

- Use the eigenvector with the second smallest eigenvalue to bi-partition the graph.

3. Decide if the current partition should be subdivided and recursively re-partition the segmented parts if necessary.

Example 4.3.1. To segment a given image, the following steps are taken [14]

- Construct a weighted graph $G = (V, E)$ by taking each pixel as a node and connecting each pair of pixels by an edge. The weight on that edge should reflect the likelihood that the two pixels belong to one object. Using just the brightness value $I = I(i)$ of the pixels and their spatial location, we can define the graph edge weight connecting the two nodes i and j as:

$$w_{ij} = \exp^{\frac{-\|\mathbf{I}(i)-\mathbf{I}(j)\|_2^2}{\sigma_I^2}} \times \begin{cases} \exp^{\frac{-\|\mathbf{X}(i)-\mathbf{X}(j)\|_2^2}{\sigma_X^2}}, & \text{if } \|\mathbf{X}(i) - \mathbf{X}(j)\|_2^2 < r. \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

  For brightness images, $\mathbf{I}(i)$ represents normalized intensity level of node i and $\mathbf{X}(i)$ represents spatial location of node i. σ_I and σ_X are parameters set to 10-20 percent of the range of their related values and r is a parameter that controls the sparsity of the resulting graph by setting edge weights between distant pixels to 0.

- Solve for the eigenvectors with the smallest eigenvalues of the system

$$(D - W)\mathbf{y} = \lambda D\mathbf{y}. \tag{4.10}$$

  As we saw above, the generalized eigensystem in 4.10 can be transformed into a standard eigenvalue problem of

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{x} = \lambda\mathbf{x}. \tag{4.11}$$

  Solving a standard eigenvalue problem for all eigenvectors takes $O(n^3)$ operations, where n is the number of nodes in the graph. This becomes impractical for image

segmentation applications where n is the number of pixels in an image.

Fortunately, our graph partitioning has the following properties:

1. The graphs are often only locally connected and the resulting eigensystems are very sparse,
2. only the top few eigenvectors are needed for graph partitioning and
3. the precision requirement for the eigenvectors is low, often only the right sign bit (which is the sign of the components of the eigenvectors) is required.

These special properties of our problem can be fully exploited by an eigensolver called the Lanczos method. The running time of a Lanczos algorithm is $O(mn) + O(mM(n))$ [7], where m is the maximum number of matrix-vector computations required and $M(n)$ is the cost of a matrix-vector computation of $A\mathbf{x}$, where

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

Note that the sparsity structure of A is identical to that of the weight matrix W. Since W is sparse, so is A and the matrix-vector computation is only $O(n)$. To see why this is the case, we will look at the cost of the inner product of one row of A with a vector $\mathbf{x}$.

Let $y_i = A_i.\mathbf{x} = \sum_j A_{ij}\mathbf{x}_j$. For a fixed i, A_{ij} is only nonzero if node j is in a spatial neighborhood of i. Hence, there are only a fixed number of operations required for each $A_i.\mathbf{x}$ and the total cost of computing $A\mathbf{x}$ is $O(n)$. The constant factor is determined by the size of the spatial neighborhood of a node. It turns out that we can substantially cut down additional connections from each node to its neighbors by randomly selecting the connections within the neighborhood for the weighted graph. Empirically, one can remove up to 90 percent of the total connections with each of the neighborhoods when the neighborhoods are large without affecting the eigenvector solution to the system [14].
Putting everything together, each of the matrixvector computations cost $O(n)$ operations with a small constant factor. The number m depends on many factors, see [7].

- After the computation of the eigenvectors, the second smallest eigenvector is used to partition the graph into two pieces. Ideally the computed eigenvector should take on

two discrete values. An indicator vector is generated from the signs of the values obtained and used to partition the graph. The ideal situation is not always the case as the eigenvectors can take on continuous values and therefore the need to find appropriate splitting points that will be used to partition the graph.

Several approaches can be adopted in finding a suitable splitting point. Common among them are the use of 0 or the median of the values as a splitting point. Another approach is to search for a suitable splitting point such that the resulting partition has the best $Ncut(A, B)$ value. Shi and Malik [14] adopted this approach in their work by checking l evenly spaced possible splitting points, and computing the best $Ncut$ among them.

- After the first step of bi-partitioning a graph using the second smallest eigenvector, the process is repeated recursively on the partitioned parts. The recursion stops when the $Ncut$ value exceeds a certain threshold value.

As mentioned previously, although the ideal case is to have the eigenvector take on discrete values, the eigenvector can sometimes take on continuous values.

Shi and Malik assert in their work that from the viewpoint of segmentation, such an eigenvector is attempting to subdivide an image region where there isn't a sure way of splitting since there will be many different splitting points with similar $Ncut$ values making the partition unstable (see [14]). Smoothly varying vector values are therefore ignored.

To achieve this, a stability criterion which measures the degree of smoothness in the eigenvector values is enforced. The histogram of the eigenvector values is computed and then the ratio between the minimum and maximum values in the bins(see appendix) is also computed. The values in the histogram bin remain relatively unchanged when the eigenvector values are continuously varying giving a high ratio. Simply thresholding the ratio between the maximum and minimum values in the histogram bin is used to exclude unstable eigenvectors.

4.4 Simultaneous K-Way Cut with Multiple Eigenvectors

The 2-way cut is computationally expensive and this is one of the major drawbacks of this method. This method only makes use of the second smallest eigenvector although it is possible to extract useful partitioning information from the next few eigenvectors.

To improve efficiency the top eigenvectors can be used simultaneously to obtain a k-way

partition. This approach uses the top n eigenvectors as an n-dimensional indicator vector for each pixel.In the first step, a simple clustering algorithm, such as the k-means algorithm, is used to obtain an oversegmentation of the image into k' groups. In the second step, one can proceed in the following two ways (see [14]):

- Greedy pruning: iteratively merge two segments at a time until only k segments are left. At each merge step, those two segments are merged that minimize the k-way $Ncut$ criterion defined as:

$$Ncut_k = \sum_{i=1}^{k} \frac{cut(A_i, V - A_i)}{assoc(A_i, V)}$$

  where A_i is the ith subset of whole set V. This computation can be efficiently carried out by iteratively updating the compacted weight matrix W^c, with $W^c(i,j) = assoc(A_i, A_j)$.

- Global recursive cut. From the initial k' segments, we can build a condensed graph $G^c = (V^c, E^c)$, where each segment A_i corresponds to a node V_i^c of the graph. The weight on each graph edge $W^c(i,j)$ is defined to be $assoc(A_i, A_j)$, the total edge weights from elements in A_i to elements in A_j. From this condensed graph, we then recursively bipartition the graph according the $Ncut$ criterion. This can be carried out either with the generalized eigenvalue system, or with exhaustive search in the discrete domain. Exhaustive search is possible in this case since k' is small, typically less than 100. [14]

Chapter 5

Image Segmentation

There are several types of images, namely, light intensity(visual) images, range images(depth images), nuclear magnetic resonance images(commonly known as magnetic resonance images(MRI)), thermal images among others with light intensity(visual) images being the most common type of image we encounter in our daily experience.

Visual images represent the variation of light intensity on the screen. In general, any image can be represented as a two-dimensional function $f(x, y)$ where (x, y) denotes the spatial coordinate and $f(x, y)$ denotes the feature value at (x, y). Depending on the type of image under consideration, the feature value $f(x, y)$ could be light intensity, depth, intensity of radio waves or temperature. [11]

A digital image is a two-dimensional discrete function $f(x, y)$ which has been digitized both in spatial coordinates and magnitude of feature value. We shall view a digital image as a two dimensional matrix whose row and column indices identify a point, known as a pixel in the image and the corresponding matrix element value identifies the feature intensity value.

Image segmentation is described as one of the most important problems in computer vision. It serves as one of the preparatory steps in getting an image ready for image analysis. Generally, image segmentation is the partitioning of an image into groups called segments such that the pixels belonging to the same partition have similar characteristics and differ from pixels from other segments. Although image segmentation is a simple problem, there is no single best known technique for solving this. Formally it can be defined as follows.

Definition 5.0.1. Let F be a set of all pixels and P be a uniformity or homogeneity predicate defined on groups of connected pixels, then segmentation is the partitioning of the set F into a set of connected subsets or regions $(S_1, S_2, S_3, ..., S_n)$ such that $F = \cup_{i=1}^{n} S_i$ with $S_i \cap S_j = \emptyset$

for all $i \neq j$. The uniformity predicate $P(S_i)$ is true for all regions S_i and $P(S_i \cup S_j)$ is false when S_i is adjacent to S_j.

This definition can be applied to all the types of images we have described.

Segmentation of images has two main objectives. The first objective is to decompose the image into parts for further analysis. In simple cases, the environment might be well controlled so that the segmentation process reliably extracts only the parts that need to be analysed. In complex cases, such as extracting a complete road network from a gray-scale aerial image, the segmentation problem can be very difficult and might require the application of some domain specific knowledge. For example segmentation could therefore be seen as a computer vision problem. A simple example of segmentation is thresholding a grayscale image with a fixed threshold t such that each pixel p is assigned to one of two classes, P_0 or P_1, depending on whether $I(p) < t$ or $I(p) \geq t$.

5.1 Image Segmentation Techniques

Hundreds of segmentation techniques are present in the literature, but there is no single method that can be considered as the best method for all classes of images, nor are all methods equally suited for a particular class of image[11]. Moreover algorithms developed for a particular class of images may not perform well on other classes of images. We introduce some of the common image segmentation techniques.

5.1.1 Thresholding

Thresholding is largely regarded as the simplest method of image segmentation. Using this method, binary images can be obtained from gray-scale images by thresholding operations which involves the choice of some pixels as the foreground pixels that make up the object of interest and the rest as background pixels. The threshold value used for segmentation can be obtained by automatically using the *histogram* of the gray-scale image. Several methods for determining the threshold have been proposed. Popular among them is the Otsu's method. [13]

Definition 5.1.2 (Histogram)**.** The histogram h of a gray-scale image I is defined by

$$h(m) = |\{(r, c) : I(r, c) = m\}|$$

where m spans the gray-level values.

5.1.3 Region-based methods

Region-based methods determine the regions directly. Region-growing is one of the popular region-based methods. These methods rely mainly on the assumption that the neighboring pixels within one region have similar values. Instead of partitioning an image, a *region grower* begins at one position in the image (often the top-left corner) and attempts to grow each region by comparing each pixel to its neighbouring pixels until the pixels being compared are too dissimilar to the region of interest[13]. Haralick proposed a region growing technique, which assumes that a region is a set of connected pixels with the same population mean and variance.

5.1.4 Histogram-based methods

Histogram-based are considered as very efficient compared to the other methods. This is due to the fact that they typically require only one pass through the pixels. A histogram is computed from all the pixels in the image as part of the method using the peaks and valleys in the histogram to locate clusters in the image[13]. Intensity and colour can be used as measure.

Recursively applying this technique to the clusters in the image will divide the clusters into smaller clusters.

5.1.5 Edge Detection

Edge detection is the name given to the set of mathematical methods which are used to identify points in the image at which pixel intensity values change abruptly or, more formally has discontinuities. Intuitively, an edge is a set of connected pixels that lie on the boundary between two regions. A point in an image is defined to be an *edge point* if its two-dimensional

first-order derivative is greater than some specified threshold [8]. First-order derivatives of a digital image are based on various approximations of the 2D gradient. The gradient of an image $f(x, y)$ at location (x, y) is defined as the *vector*

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Two important quantities in *edge detection* are the magnitude of the gradient vector given by

$$\|\nabla f\| = \sqrt{{G_x}^2 + {G_y}^2}$$

which gives the maximum rate of increase of $f(x, y)$ per unit distance in the direction of ∇f and the *direction* of the gradient vector given by

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

which represents the direction angle of ∇f at (x, y).

Some of the common edge operators are the Robert's operator, Sobel's operator and Prewitt's operator[8]. The following figure shows the results of edge detection by the Sobel operator.

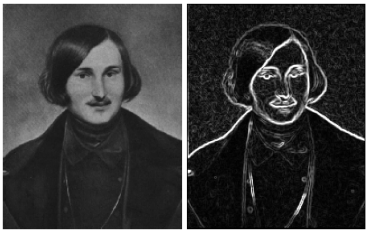


Figure 5.1: Edge detection of a man's image with the Sobel operator

5.1.6 Partial differential equation-based methods

Using a partial differential equation method, one can also segment an image by solving the PDE using a numerical scheme. Curve propagation is one of the popular techniques in this category with numerous applications to object extraction, stereo reconstruction, object tracking among others. The main idea is to evolve an initial curve towards the potential of a cost function [2].

By the classical energy based snakes, Let $\mathcal{C}(q) : [0,1] \to \mathbb{R}^2$ be a parametrized planar curve and let $I : [0,a] \times [0,b] \to \mathbb{R}^+$ be a given image in which we want to detect the object's boundaries. The classical snakes approach associates the curve $\mathcal{C}$ with an energy given by

$$E(\mathcal{C}) = \alpha \int_0^1 |\mathcal{C}'(q)|^2 \mathrm{d}q + \beta \int_0^1 |\mathcal{C}''(q)|^2 \mathrm{d}q - \lambda \int_0^1 |\nabla I(\mathcal{C}'(q))| \mathrm{d}q$$

where α, β, and λ are real positive constants. The first two terms control the smoothness of the contours to be detected (internal energy), while the third term is responsible for attracting the contour towards the object in the image (external energy). Solving the problem of snakes amounts to finding, for a given set of constants α, β, and λ the curve $\mathcal{C}$ that minimizes E.[2]

5.2 Segmentation by Grouping

Image segmentation can be related to perceptual grouping and organization in vision. Good continuation, similarity and proximity are among the key factors that influence visual grouping. However, perceptual grouping still has many computational issues that remain unresolved. [14].

In this thesis, the graph theoretic approach to this problem is adopted, focusing specifically on the case of image segmentation.

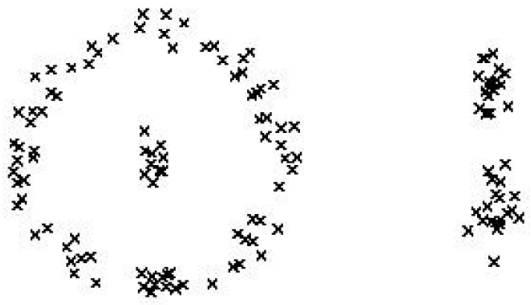


Figure 5.2: Points in a plane - what is the "correct" grouping

When we consider Figure 5.2, we realise that there exists more than one partitioning of the image into subsets. How then do we choose the "right" partition. It is necessary first of all to understand that there may be more than one correct partition. There are several possible interpretations in the context of prior world knowledge. It is however difficult to specify this prior world knowledge. Some of it is in the form of local properties, such as coherence

of brightness, color, texture, or motion, but equally important are global properties about symmetries of objects or object models.

Partitioning is inherently hierarchical. It is therefore more appropriate to think of returning a tree structure corresponding to a hierarchical partition instead of a single "flat" partition[14]. The key point here is that image segmentation is to be done from the big picture downward, marking out the major areas and then filling in the details.

There has been much previous work in the image segmentation and data clustering community using variations of the minimal spanning tree or limited neighbourhood set approaches. The segmentation criteria employed by these methods are based purely on the local properties of the graph which makes it fall short of the main goal of extracting global impressions from a scene. Global properties about symmetries of objects are as important as local properties in obtaining a good cluster.

5.3 Graph theoretic formulation

Grouping can be formulated as a graph partitioning and optimization problem. The graph theoretic formulation of image segmentation is as follows:

1. The set of points in an arbitrary feature space are represented as a weighted undirected graph $G = (V, E)$, where the nodes of the graph are the points in the feature space.
2. An edge is formed between every pair of nodes yielding a dense or complete graph.
3. The weight on each edge, $w_{ij} = w(i, j)$ is a function of the similarity between nodes i and j.
4. Partition the set of vertices into k disjoint sets $V_1, V_2, \ldots, V_k$ where by some measure the similarity among the vertices in a set V_i is high and, across different sets V_i, V_j is low.

To partition the graph in a meaningful manner, we also need to:

- pick an appropriate criterion (which can be computed from the graph) to optimize which would result in a good segmentation and
- find an efficient way to achieve the optimization.

5.3.1 Weight function

The weight function that we use to represent similarity between nodes must have generality, must be easy to compute, should be tweakable with parameters, and should take into account pixel gray level difference as well as radial distance between two pixels. As we saw in section 4.3 , one such weight function (see [14]) is:

$$w_{ij} = \exp^{\frac{-\|\mathbf{I}(i)-\mathbf{I}(j)\|_2^2}{\sigma_I^2}} \times \begin{cases} \exp^{\frac{-\|\mathbf{X}(i)-\mathbf{X}(j)\|_2^2}{\sigma_X^2}}, & \text{if } \|\mathbf{X}(i)-\mathbf{X}(j)\|_2^2 < R. \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

This weight measure reflects the likelihood of two pixels belonging to the same object. As gray level difference and Euclidean distance decrease (in other words, as pixels become more similar), $w(i,j)$ increases. Figure 5.3 illustrates this schematically with a graph superimposed on the corresponding image. Similar pixels have thicker edges between them.

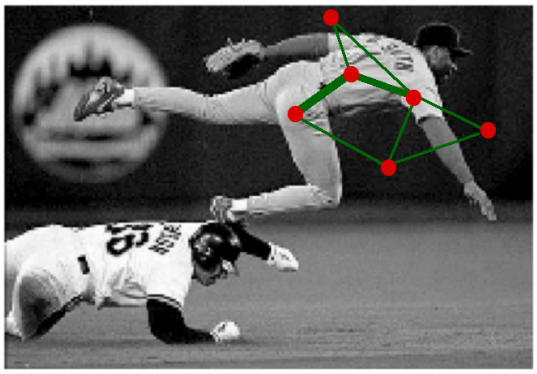


Figure 5.3: Schematic diagram showing graph weight edges for a gray scale image. Higher weights are shown as thicker edges. From: [14]

Figure 5.4 shows how the weight matrix, W, which represents the graph of the brightness image is computed from the image. W is an adjacency matrix for the graph and is an $n \times n$ matrix such that $W(i,j) = w_{ij}$. Here n is the total number of pixels in the image (number of rows $\times$ number of columns).

Figures 5.4 (c) and (d) show the weights to all the other nodes from two particular pixels, i_1 and i_2 in the form of brightness images.

The weights are shown in the form of an image with higher weights being denoted by brighter points in the corresponding images.

Thus, pixels similar to the selected pixel are brighter and the ones which are dissimilar are darker in the two figures (c) and (d).

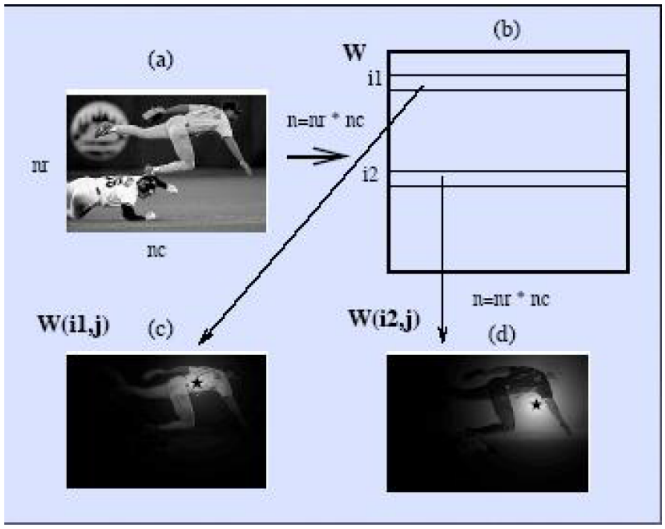


Figure 5.4: Edge weights shown in terms of brightness values for two pixels. From [14]

5.4 Matrix formulation

The criteria given in the preceding section can be cast into matrix form which in turn can be used for converting the optimization problem into a generalized eigenvalue problem.

Let $\mathbf{x}$ be an $n = |V|$ dimensional indicator vector such that

$$x_i = \begin{cases} 1, & \text{if } i \text{ belongs to set } A. \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

In Figure 5.5, the entries in x shaded yellow signify nodes that are in set A. Multiplying $\mathbf{x}^T W \mathbf{x}$ will give the sum of the weights within the red rectangle which is nothing but twice the sum of the weights connecting nodes in A or $2 \times assoc(A, A)$. Similarly, in Figure 5.6, the sum of the weights within the blue rectangle represents $2 \times assoc(A, V)$. This sum cannot be found by multiplying the indicator vector x with the weight matrix. Instead, we use the diagonal matrix D such that the diagonal entries of D contain the sums of the corresponding row in W.
So that $D(i, i) = \sum_j W(i, j)$. Now, the sum of the weights within the blue rectangle can be written $\mathbf{x}^T D \mathbf{x}$.

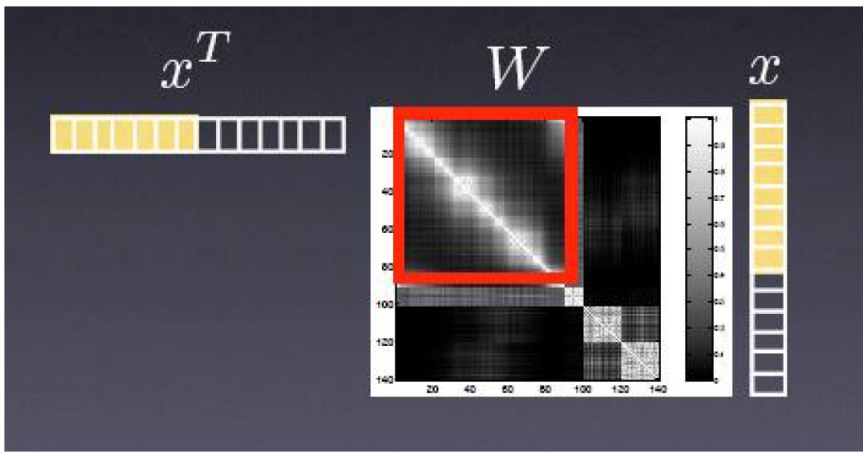


Figure 5.5: The points shaded yellow within the indicator vector x denote nodes belonging to set A (they all correspond to 1's). Sum of weights within the red rectangle represents 2*assoc(A,A). From: [14]

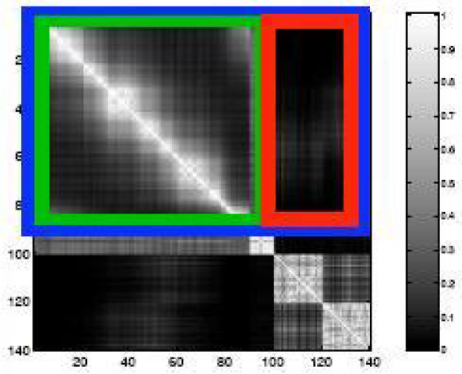


Figure 5.6: Sum of weights within (i) blue rectangle represents 2*assoc(A,V) (ii) red rectangle represents 2*cut(A,V-A). From: [14]

Notice that :

$$\begin{aligned} 2 * assoc(A, A) &= \mathbf{x}^T W \mathbf{x} \\ 2 * assoc(A, V) &= \mathbf{x}^T D \mathbf{x} \\ 2 * cut(A, V - A) &= 2 * assoc(A, V) - 2 * assoc(A, A) \\ &= \mathbf{x}^T (D - W) \mathbf{x}. \end{aligned}$$

Thus, the sum of the weights within the red rectangle in Figure 5.6 represents $2 * cut(A, V - A)$.

Chapter 6

Implementation and Results

In the preceeding chapters of this thesis, we have developed the mathematical framework to solve the image segmentation problem. We have seen how to model our image segmentation problem as a graph partitioning problem, and apply spectral partitioning to partition the graph into the desired segments.

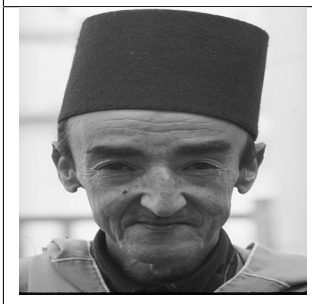
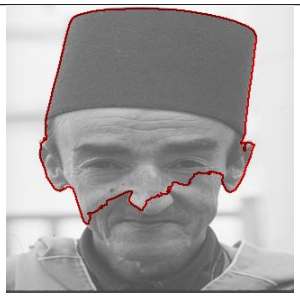
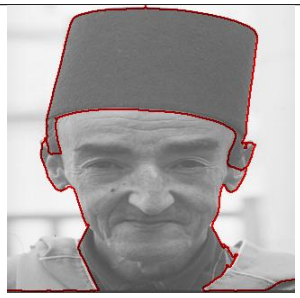
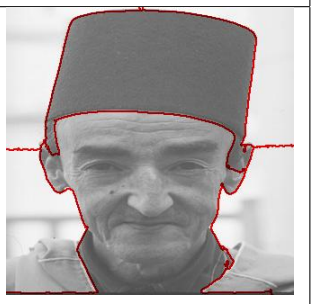
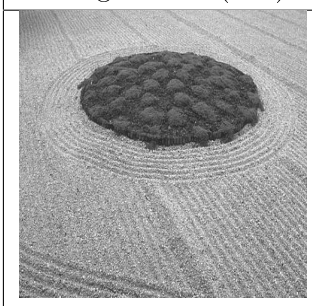
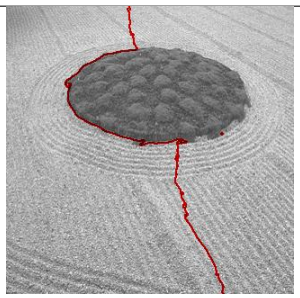
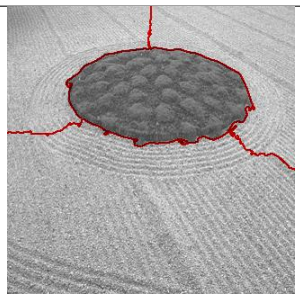
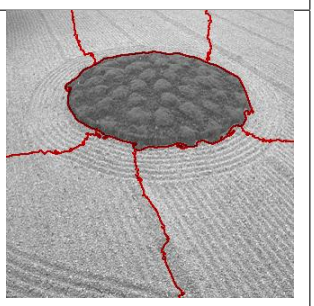
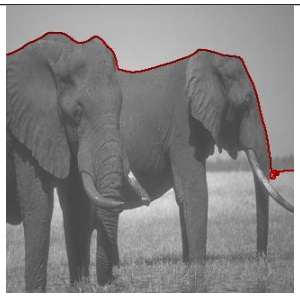
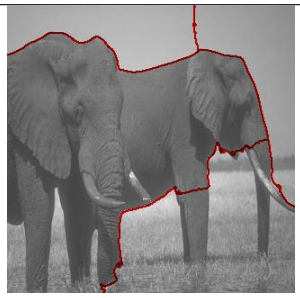
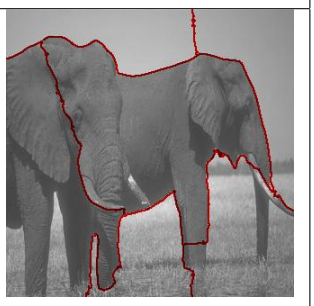
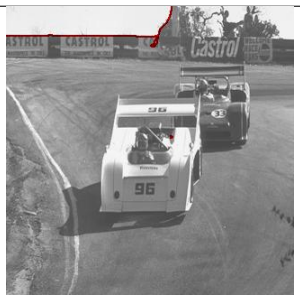
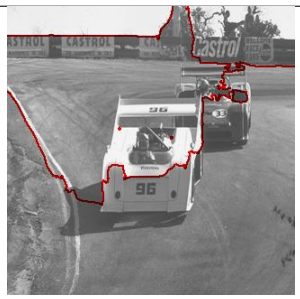
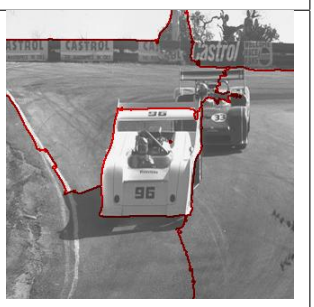
In this Chapter, we implement the spectral partitioning algorithm for image segmentation using matlab. This implementation is based on the Ncut implementation by Timothee Cour, Jianbo Shi and Stella Yu, see [4].

6.1 Analysis of Existing Implementation.

The following table summarises the results of the existing implementation. We highlight pointers in the results we can explore to yield better results.Key among them are the segmentation results produced on some images which do not reflect natural segmentation as well as the running time of the implementation. We propose ways by which these key problems can be addressed and plug the situations into the existing implementation to obtain desired results.

6.1.1 Observations

Test images were obtained from the Berkeley Segmentation Dataset and Benchmark. These images were segmented using the Ncut matlab code from [4] and the following results were obtained. All images were 320 by 320 pixels.

Test Results			
Original Image	2 Segments	4 Segments	6 Segments
			
Average Time (Sec):	24.5054	14.2726	15.9372
			
Average Time (Sec):	12.3173	15.2734	13.6664
			
Average Time (Sec)	16.5067	18.4200	19.9385
			
Average Time (Sec)	17.5835	18.0703	17.5350

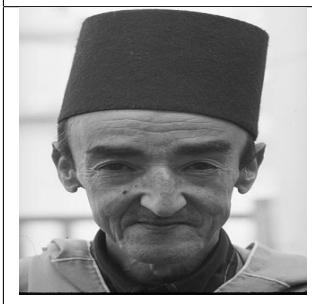
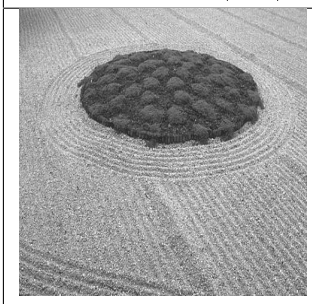
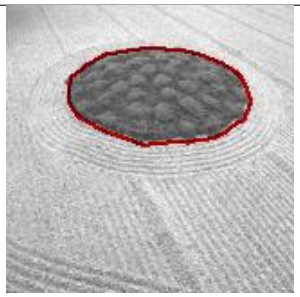
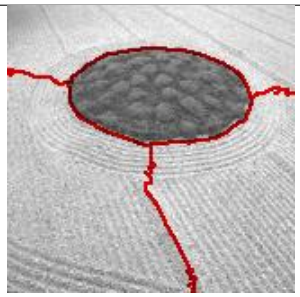
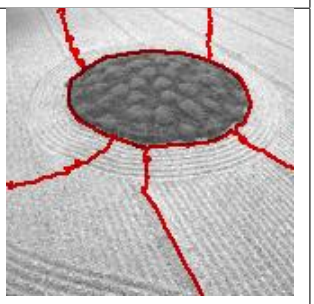
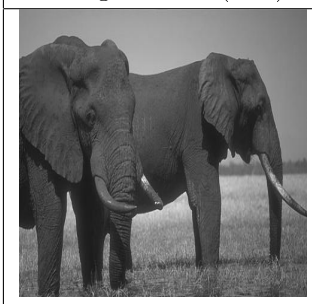
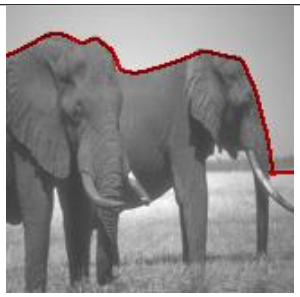
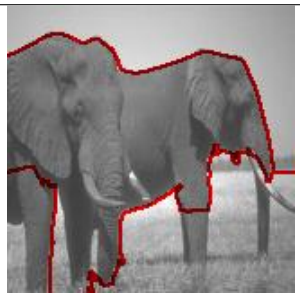
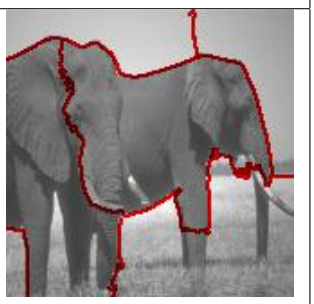
6.1.2 Discussion

We realized from the results of the experiments that there are two main issues with the implementation. These are :

- The time it takes to segment the input image into the desired partitions
- The quality of segmentation produced for some of the test images

To address these problems, in the case of the quality of segments produced, we propose the smoothening of the input image which will cause the edges to stand out. This will increase the quality of the segmentation, producing results that are near *expected* segmentation.

The time taken for the segmentation of the image is largely based on the size of the *weighted* adjacency matrix. For a 320 x 320 image, we will be working with a *weighted* adjacency matrix of size 102400 x 102400 which increases the computational complexity. We therefore scaled down the size of the image by a factor of 0.5 and noticed that the results were encouraging. We summarize our results in the table below:

Results			
Original Image	2 Segments	4 Segments	6 Segments
			
Average Time (Sec):	5.0045	3.7670	4.2767
			
Average Time (Sec):	4.4704	3.3940	4.2340
			
Average Time (Sec)	3.8011	4.7506	4.7253
			
Average Time (Sec)	4.6279	4.1068	4.6454

6.2 Conclusion

In this thesis, we have seen that the image segmentation problem can be modeled as a graph partitioning problem which makes it possible to apply spectral clustering techniques to find good partitions for our image. The matrix representation of the problem is a natural way of representing the problem in the computer system. The eigenvector corresponding to the second eigenvalue was first used by Miroslav Fiedler to partition a graph two parts [6]. This approach is an efficient way to partition graphs, however, the choice of parameters in assigning edge weights is very difficult and the automatic determination of the values of these parameters will be a good research topic to consider in the future.

References

[1] Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. Macmillan Press Ltd., New York.

[2] Caselles V., R. K. and Sapiro, G. (1997). *Geodesic Active Contours*. International Journal of Computer Vision, 22:61–79.

[3] Chung, F. R. K. (1997). *Spectral Graph Theory*. Ameriacan Mathematical Society, Providence, Rhode Island.

[4] Cour T., S. Y. and Shi, J. (2004). *MATLAB Normalized Cuts Segmentation Code*. `http://www.cis.upenn.edu/~jshi/software/`. Accessed: 2015-06-10.

[5] Deo, N. (1974). *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Inc., Englewood Cliffs, N. J.

[6] Fiedler, M. (1973). *Algebraic Connectivity of Graphs*. Czechoslovak Mathematics Journal, 23:298–305.

[7] Golub, G. H. and van Loan, C. F. (1989). *Matrix Computations*. John Hopkins Press, Englewood Cliffs, N. J.

[8] Gonzalez, R. C. and Woods, R. E. (2002). *Digital Image Processing*. Prentice Hall, Inc, Upper Saddle River, New Jersey 07458, 2nd edition.

[9] Luxburg, U. V. (2007). *A Tutorial on Spectral Clustering*. `http://arxiv.org/pdf/0711.0189.pdf`. Accessed: 2015-02-10.

[10] Mohar, B. (1997). *Some applications of Laplace eigenvalues of graphs*. G. Hahn and G. Sabidussi(Eds.), Graph Symmetry: Algebraic Methods and Applications, 497:225–275.

[11] Pal, S. K. and Pal, N. R. (1993). *A Review on Image Segmentation Techniques*. Pattern Recognition, 26:1277–1294.

[12] Pothen A., H. S. and Liou, K. (1990). *Partitioning Sparse Matrices with Eigenvectors of Graphs*. SIAM J. Matrix Analytical Applications, 11:430–452.

[13] Shapiro, L. G. and Stockman, G. C. (2001). *Computer Vision*. Prentice Hall, Inc, Upper Saddle River, NJ, USA.

[14] Shi, J. and Malik, J. (2000). *Normalized Cuts and Image Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22:888–905.

[15] Wu, Z. and Leahy, R. (1993). *An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15:1101–1113.

Appendix A

Some useful information

A.1 Normalized Cut is NP-Complete

Proposition A.1.1 (Shi and Malik[14]). *Normalized Cut (Ncut) for a graph on regular grids is NP-complete.*

Proof. A.1.1 We shall reduce Ncut on regular grids from PARTITION:

- Given integers $x_1, \ldots, x_n$ adding to $2k$, is there a subset adding to k?

We construct a weighted graph on a regular grid that has the property that it will have a small enough normalized cut if and only if we can find a subset from $x_1, x_2, \ldots, x_n$ adding to k. Fig.A.1(a) shows the graph and Fig.A.1(b) shows the form that a partition that minimizes the normalized cut must take.

In comparison to the integers $x_1, x_2, \ldots, x_n$, M is much larger, $M > 2k^2$, and a is much smaller, $0 < a < 1 = n$. We ask the question

- Is there a partition with $Ncut$ value less than $\frac{4an}{c-1/c'}$, where c is half the sum of edge weights in the graph, $c = 2M(n+1) + k + 3an$.

We shall see that a good $Ncut$ partition of the graph must separate the left and right columns. In particular, if and only if there is a subset $S_1 = \{x_1, \ldots, x_m\}$ adding to k, by taking the corresponding edges in the middle column to be in one side of the partition, as illustrated

in Fig.A.1(b), we achieve an Ncut value less than $\frac{4an}{c-1/c}$. For all other partitions, the $Ncut$ value will be bounded below by $\frac{4an}{c-1/c}$

First, let us show that the cut illustrated in Fig.A.1(b), where each side has a subset of middle column edges $x_1, x_2, \ldots, x_n$ that add up to k, does have $Ncut$ value less than $\frac{4an}{c-1/c}$. Let the $ncut*$ be the $Ncut$ value for this cut. By using the formula for Ncut 4.2, we can see that

$$ncut* = \frac{4an}{2c + 2an(2k_1 - 1)} + \frac{4an}{2c - 2an(2k_1 - 1)} \tag{A.1}$$

where c is half the total edge weights in the graph, $c = 2M(n+1) + k + 3an$, and $k_1 n$ and $(1-k_1)n$ are the number of edges from the middle column on the two sides of the graph partition, $0 < k_1 < 1$. The term $an(2k_1 - 1)$ can be interpreted as the amount of imbalance between the denominators in the two terms in the $Ncut$ formula and lies between -1 and -1 (since $0 < an < 1$). Simplifying, we see that

$$ncut* = \frac{4anc}{c^2 - (an(2k_1 - 1))^2} < \frac{4anc}{c^2 - 1} = \frac{4an}{c - 1/c} \tag{A.2}$$

as was to be shown.

To complete the proof we must show that all other partitions result in a $Ncut$ greater than or equal to$\frac{4an}{c-1/c}$. Informally speaking, what will happen is that either the numerators of the terms in the $Ncut$ formula - the cut become too large, or the denominators become significantly imbalanced, again increasing the $Ncut$ value. We need to consider three cases:

1. A cut that deviates from the cut in 1(b) slightly by reshuffling some of the x_i edges so that the sums of the x_i in each subset of the graph partition are no longer equal. For such cuts, the resulting $Ncut$ values are, at best, $Ncut_1 = \frac{2an}{c+x} + \frac{2an}{c-x} = \frac{4anc}{c^2-x^2}$

2. A cut that goes through any of the edges with weight M. Even with the denominators on both sides completely balanced, the Ncut value $Ncut_2 = \frac{2M}{c}$ is going to be larger than $\frac{4an}{c-1/c}$. This is ensured by our choice in the construction that $M > 2k^2$. We have to show that $\frac{2M}{c} \geq \frac{4an}{c-1/c}$. This is direct since $an < 1$ by construction, $\frac{c^2}{c^2-1} \leq \frac{81}{80}$(using $k \geq 1, M \geq 2, c \geq 9$).

3. A cut that partitions out some of the nodes in the middle as one group. We see that any cut that goes through one of the $x_i's$ can improve its $Ncut$ value by going through the edges with weight a instead. So, we will focus on the case where the cut only goes

through the weight a edges. Suppose that m edges of xis are grouped into one set, with total weight adding to x, where $1 < x < 2k$. The corresponding *ncut* value,

$$\begin{aligned} ncut_3(m) &= \frac{4am}{4am+2x} + \frac{4am}{8M(n+1)+4k+12an-4am-2x} \\ &= \frac{2am}{c-d_m} + \frac{2am}{c+d_m}, \end{aligned} \tag{A.3}$$

where

$$\begin{aligned} d_m &= 2M(n+1)+k+3an-2am-x \\ &> 2M(n+1)-k+3an-2am \\ &= x_l. \end{aligned}$$

The lower bound on $ncut_3(m) = \frac{4amc}{c^2-{d_m}^2}$ is then $ncut_l(m) = \frac{4amc}{c^2-{x_l}^2}$. Further expansion of the $ncut_l(m)$ yields

$$\begin{aligned} ncut_l(m) &= \frac{4amc}{c^2-{x_l}^2} \\ &= \frac{4amc}{c^2-b-2am^2} \\ &= \frac{4ac}{\frac{c^2-B^2}{m}-4a^2m+4aB} \end{aligned} \tag{A.4}$$

where $B = 2M(n+1)-k+3an$

One can check to see that $ncut_l(m)$ is a non-decreasing function and has its minimum at $\frac{4ac}{(c^2-B^2)+4aB-4a^2}$ when $m = 1$.

In order to prove that $ncut_l(m) > \frac{4an}{c-1/c'}$, we need to establish the inequality

$$\frac{4ac}{(c^2-B^2)+4aB-4a^2} \geq \frac{4anc}{c^2-1}$$

using the fact that $c = B+2k$. To continue, note that, since $an < 1$, this will be true if $(4ck-4k^2)n+4(c-2k)-4a+1 \leq c^2$ since $n < k$. Since $4k^3+8k_4a-1 > 0$, we only need to show that $4ck^2+4c < c^2$ or that $c > 4(k_1^2)$. This is so because $c = 2M(n+1)+k+3an$ and $M > 2k^2$. □

Definition A.1.2. In computing the histogram, the range of values of the numeric dataset is split into equal-sized classes known as bin.

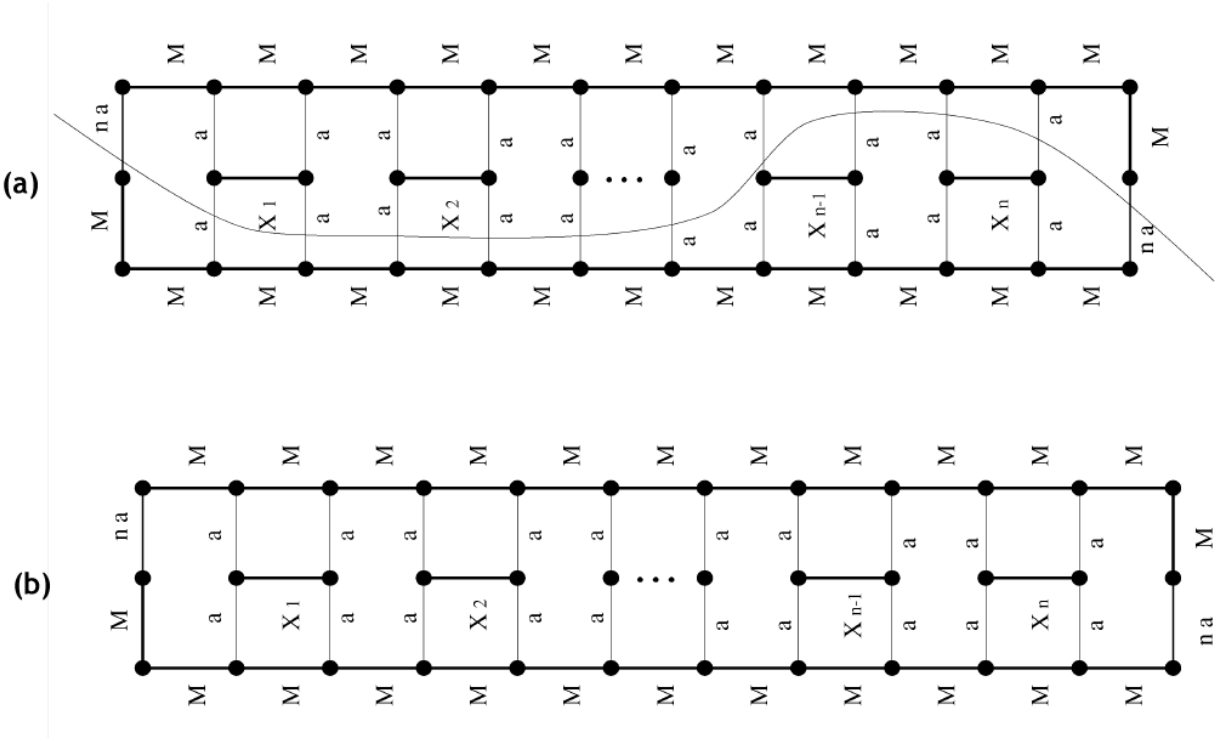


Figure A.1: (a) shows a weighted graph on a regular grid. The missing edges on the grids have weights of 0. In comparison to the integers $x_1, x_2, \ldots, x_n$, M is a large number ($M > 2k^2$), and a is very small number ($0 < a < 1/n$). (b) shows a cut that has a $Ncut$ value less than $\frac{4an}{c-1/c}$ This cut, which only goes through edges with weight equal to a or na, has the property that the x_i's on each side of the partition sum up to k.