

CMSC 420: Laplacian Matrices and Graph Clustering

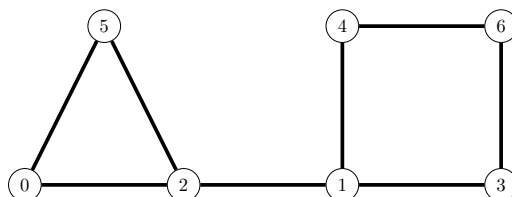
Justin Wyss-Gallifent

March 27, 2024

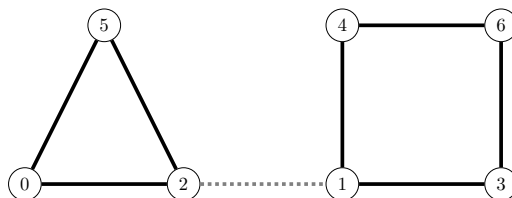
1	Introduction	2
2	CS Commentary	2
3	Maths!	3
	3.1 The Laplacian Matrix	3
	3.2 Matrix and Vector Multiplication	4
	3.3 Eigenvectors and Eigenvalues	4
	3.4 The Fiedler Vector and Value	5
4	Graph Partitioning	6
5	Notes	6

1 Introduction

The goal of this set of notes is to demonstrate a simple way to divide a graph into two “strongly connected” subgraphs when possible. For example consider the graph:



Intuitively this can be viewed as a triangle (on the left, strongly connected) and a square (on the right, strongly connected) with the triangle and square weakly connected to one another, as such:



Given a graph, how can we store this (a data structure!) in a way which allows us to obtain these clusters computationally?

It turns out that this can be done fairly easily if we store the graph as a matrix called the Laplacian Matrix. So in that sense think of the Laplacian Matrix as the data structure.

2 CS Commentary

The approach in these notes is very computation-based with little to no theory. It is done this way to be accessible to computer science students who may have limited (or no) exposure to linear algebra and simply want to know what’s going on at a high-level for application.

If you’re interested in learning more details than appear here you can ask me (Justin) or check the graph theory module of my MATH401 notes, available online.

3 Maths!

3.1 The Laplacian Matrix

Adjacency matrices are commonly used to store simple graphs. For a graph with n vertices we construct an $n \times n$ matrix A where:

$$A[i, j] = \begin{cases} 1 & \text{if vertices } i \text{ and } j \text{ are connected by an edge.} \\ 0 & \text{if vertices } i \text{ and } j \text{ are not connected by an edge.} \end{cases}$$

Example 3.1. The graph above has adjacency matrix:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Note 3.1.1. Observe that we are 0-indexing the rows and columns here, as is typical in CS.

There are two other matrices that arise frequently.

Definition 3.1.1. The degree matrix D has $A[i, i]$ equal to the degree of the vertex i and 0 elsewhere.

Definition 3.1.2. The Laplacian matrix is defined by $L = D - A$.

Example 3.2. Thus for the graph given in the introduction we have:

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$
$$L = D - A = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 2 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 2 \end{bmatrix}$$

Observe that the Laplacian matrix is sufficient enough to describe the graph completely. The adjacency matrix is, too, but the Laplacian matrix turns out to be more useful.

3.2 Matrix and Vector Multiplication

Given an $n \times n$ matrix A and a vector with n entries (an $n \times 1$ matrix) \vec{v} we can perform the multiplication $A\vec{v}$ to form a new vector with n entries.

We do this by multiplying each of the i^{th} entries in \vec{v} by the i^{th} column in A and adding.

Example 3.3. For example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 1 \\ 5 & 7 & 6 \end{bmatrix} \begin{bmatrix} 10 \\ 15 \\ 30 \end{bmatrix} = 10 \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} + 15 \begin{bmatrix} 2 \\ 4 \\ 7 \end{bmatrix} + 30 \begin{bmatrix} 3 \\ 1 \\ 6 \end{bmatrix} = \begin{bmatrix} 130 \\ 90 \\ 335 \end{bmatrix}$$

3.3 Eigenvectors and Eigenvalues

Given an $n \times n$ matrix A there are some very special vectors associated to A .

Definition 3.3.1. A vector \vec{v} is an eigenvector for A if $A\vec{v} = \lambda\vec{v}$ for some constant λ . The constant λ is called an eigenvalue.

Example 3.4. Let's define:

$$A = \begin{bmatrix} -5 & 2 \\ -7 & 4 \end{bmatrix} \quad \text{and} \quad \vec{v} = \begin{bmatrix} 2 \\ 7 \end{bmatrix}$$

Observe that:

$$A\vec{v} = \begin{bmatrix} -5 & 2 \\ -7 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 7 \end{bmatrix} = \begin{bmatrix} 4 \\ 14 \end{bmatrix} = 2 \begin{bmatrix} 2 \\ 7 \end{bmatrix} = 2\vec{v}$$

Thus we can say that this \vec{v} is an eigenvector with eigenvalue $\lambda = 2$.

Finding eigenvalues and eigenvectors is not overly difficult in theory but the associated calculations can be lengthy and approximations are often necessary.

Because of this nobody finds them by hand except for demonstration of the procedure. Luckily both eigenvectors and eigenvalues can be found easily with software, for example in Python the `numpy` package can do it.

Before proceeding observe that any multiple of an eigenvector is also an eigenvector with the exact same eigenvalue.

|

Example 3.5. Going off the above, observe that:

$$\begin{bmatrix} -5 & 2 \\ -7 & 4 \end{bmatrix} \begin{bmatrix} 6 \\ 21 \end{bmatrix} = \begin{bmatrix} 12 \\ 42 \end{bmatrix} = 2 \begin{bmatrix} 6 \\ 21 \end{bmatrix}$$

3.4 The Fiedler Vector and Value

For a connected graph G with n vertices the Laplacian matrix L has n eigenvalues (with some repeats - we won't go into the meaning of this) satisfying:

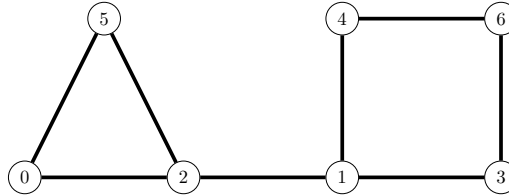
$$0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

Moreover in all cases that we'll look at we will in fact have:

$$0 = \lambda_1 < \lambda_2 \leq \dots \leq \lambda_n$$

Definition 3.4.1. The smallest positive eigenvalue is called the Fiedler value, named after Miroslav Fiedler (1926 - 2015). An associated eigenvector is called a Fiedler vector.

Example 3.6. The graph from the opening of the notes:



We saw has Laplacian matrix:

$$L = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 2 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 2 \end{bmatrix}$$

The eigenvalues for this matrix are, with inequalities to reference above:

$$0 < 0.3588 < 2.0000 \leq 2.2763 \leq 3.0000 \leq 3.5892 \leq 4.7757$$

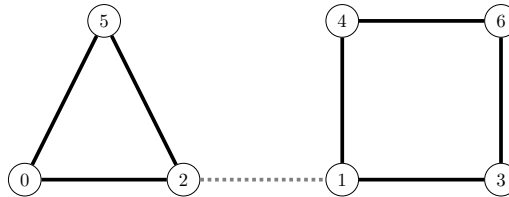
The Fiedler value is then 0.3588. An associated Fiedler vector is:

$$\bar{v} = \begin{bmatrix} 0.48 \\ -0.15 \\ 0.31 \\ -0.35 \\ -0.35 \\ 0.48 \\ -0.42 \end{bmatrix}$$

4 Graph Partitioning

It turns out to be the case that when a graph has a fairly natural partition into two strongly connected subgraphs of equal size then we can identify the nodes in each subgraph by looking at the Fiedler vector and separating the indices by whether the entries are positive or negative. If there happen to be entries which are 0 (happens rarely) they can go either way. We'll group them with the positive entries.

Example 4.1. In the graph we've been examining in our examples the indices in the Fiedler vector which have positive entries are indices 0, 2, 5 and the indices in the Fiedler vector which have negative entries are indices 1, 3, 4, 6. If we look at the graph as two subgraphs, one with vertices 0, 2, 5 and the other with vertices 1, 3, 4, 6 we see the following:



The Fiedler vector has done what we claimed!

5 Notes

A few notes that may be useful to consider:

1. An eigenvector entry of 0 could go either way in terms of which cluster to put it in.
2. Since any multiple of an eigenvector is also an eigenvector we might wonder what software will do. In practice most software returns a unit eigenvector, meaning it has length 1. However since there are two unit eigenvectors (because we can negate) different software may give different results. I've even seen Matlab give different results!

Example 5.1. Consider the matrix we saw earlier:

$$A = \begin{bmatrix} -5 & 2 \\ -7 & 4 \end{bmatrix}$$

We saw that $\bar{v} = \begin{bmatrix} 2 \\ 7 \end{bmatrix}$ is an eigenvector. To get a unit eigenvector we'd divide by the length $\sqrt{4 + 49} = \sqrt{53}$ and get:

$$\begin{bmatrix} 2/\sqrt{53} \\ 7/\sqrt{53} \end{bmatrix} \approx \begin{bmatrix} 0.27472112789 \\ 0.96152394764 \end{bmatrix}$$

Python's `numpy` package gives the negative version of this:

```
% python3
>>> import numpy as np
>>> A = np.array([[ -5, 2], [ -7, 4]])
>>> eval, evec = np.linalg.eig(A)
>>> eval
array([-3.,  2.])
>>> evec
array([[ -0.70710678,  -0.27472113],
       [ -0.70710678,  -0.96152395]])
>>>
```

We see the second eigenvalue of 2 corresponding to the second column of the matrix:

$$\begin{bmatrix} -0.27472113 \\ -0.96152395 \end{bmatrix}$$