An Overview of Graph Spectral Clustering and Partial Differential Equations

Max Daniels[3] Catherine Huang[4] Chloe Makdad[2]

Shubham Makharia[1]

[1]Brown University [2]Butler University, [3]Northeastern University, [4]University of California, Berkeley

August 19, 2020

Abstract

Clustering and dimensionality reduction are two useful methods for visualizing and interpreting a dataset. This can be a challenging task for high dimensional data because accurate clustering requires resolving which of the many features are important, and which are not. One approach for clustering high dimensional data is Graph Spectral Clustering, in which data clusters are derived from spectral properties of a representative matrix of the *similarity graph*, which captures similarity between datapoints. In this work, we illustrate the connections between Graph Spectral Clustering and a discretized diffusion process on the similarity graph. We then demonstrate an analogous clustering method using a continuous diffusion process in the original data space and we additionally explore clustering by way of a related physical process: wave propagation on the similarity graph. Finally, we show numerical results of the algorithms we discuss applied to synthetic datasets.

Contents

This project was conducted during the virtual Summer@ICERM 2020 program organized by the Institute for Computational and Experimental Research in Mathematics. We thank our project advisors, Akil Narayan and Minah Oh, our graduate student advisors, Liu Yang and Alex Mihai, and the organizing faculty at ICERM for hosting a wonderful program despite many unique challenges this year.

1 Introduction

High dimensional datasets, having a relatively large number of features per data instance, are subjects of analysis in many valuable areas: computational biology, medical imaging, and health informatics, as examples. These datasets are often controlled by relatively few intrinsic factors of variation. For example, images of human faces are highly compressible and can be described concisely by factors like 'hair color,' 'eye shape,' or 'skin tone.' In other datasets, identifying special features of variation may be less valuable than partitioning the dataset into groups of points which share meaningful characteristics. These groups, or 'clusters,' can be analyzed after the fact by their average or most-common features. In either case, the unnecessary degrees of freedom provided by high dimensionality incur noise, representation ambiguity, and added computational cost, making it crucial to find efficient techniques for clustering and dimensionality reduction which can extract only the important information from a high dimensional dataset.

We study graph spectral clustering, a clustering procedure which is geometrically motivated for high dimensional data having few features of variation. Graph spectral clustering is closely related to dimension reduction through Laplacian eigenmaps [1], and our exposition applies to both. Our ultimate goal is to demonstrate the connections of these algorithms to diffusion processes on a special discrete domain induced by a dataset.

In Section 2, we introduce graph spectral clustering and discuss its classical interpretations. In Section 3, we provide important context about differential equations, which we leverage in Section 4 to explain the connections between graph spectral clustering and diffusion processes. We show experimental results and future investigations in Sections 5. Additional proofs and supplemental materials may be found in the appendix. Additional results and code to reproduce our experiments can be found at `https://ghost-clusters.github.io/icerm-spectral-clustering/`.

2 Graph Spectral Clustering

To cluster a dataset, graph spectral clustering begins by reducing the data to a similarity graph. Nodes of the graph represent datapoints and edges have nonnegative scalar weights which represent pairwise similarities between data points. While the similarity function is domain dependent, construction of the similarity graph essentially eliminates the dimensionality of the original data. From the similarity graph, we derive an associated matrix called the graph laplacian, whose eigenvectors provide a lower dimensional data representation which can be more effectively clustered than the original dataset. In this sense, it is the graph laplacian whose spectral properties control the graph spectral clustering algorithm. We will define in this section the similarity graph, the graph laplacian, and the basic graph spectral clustering procedure.

2.1 Notation and Context

Consider a set of data points $x = \{x_1, x_2, \dots x_n\}$ where each $x_i \in \mathbb{R}^d$ is a d-dimensional feature vector with features $(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(d)})$. For certain data, the inner product or covariance $\langle x_i, x_j \rangle$ between pairs of points is a meaningful indicator of their semantic similarity[1]. Graph spectral clustering generalizes pairwise similarity through a domain dependent similarity $s(x_i, x_j)$ satisfying $s(x_i, x_j) \geq 0$ for all $i, j \leq n$. For example, in Figure 4, we use the Gaussian similarity function $s(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2/(2\sigma^2)\right)$, where σ is a scaling parameter.

From the similarity function, we construct a weighted, undirected similarity graph $G = (V, E)$ in which each vertex $v_i \in V$ represents a data point x_i. In this work, we focus on the *fully connected* similarity graph whose adjacency matrix $W \in \mathbb{R}^{n\times n}$ given by $w_{ij} = s(x_i, x_j)$. Some common alternative methods to compute W are shown in Fig. 1.

[1]Note the relationship between the Euclidean distance and Euclidean inner product: $2\langle x_i, x_j \rangle = \|x_i\|_2^2 + \|x_j\|_2^2 - \|x_i - x_j\|_2^2$. Correlation is meaningful in cases when the Euclidean distance between datapoints is meaningful, and vice versa.

Similarity Graphs	Edge Weights
ϵ-neighborhood graph	$w_{ij} = \begin{cases} 1 & s(x_i, x_j) < \epsilon \\ 0 & \text{else} \end{cases}$
k-nearest neighbors graph	$w_{ij} = \begin{cases} s(x_i, x_j) & x_i \in k\text{-}\texttt{NN}(x_j) \text{ or } x_j \in k\text{-}\texttt{NN}(x_i) \\ 0 & \text{else} \end{cases}$
Mutual k-nearest neighbors graph	$w_{ij} = \begin{cases} s(x_i, x_j) & x_i \in k\text{-}\texttt{NN}(x_j) \text{ and } x_j \in k\text{-}\texttt{NN}(x_i) \\ 0 & \text{else} \end{cases}$

Figure 1: Common examples of data similarity graphs and the associated similarity functions. Here, $k\text{-}\texttt{NN}(x_i)$ is the set of top-k most similar datapoints to x_i.

We define the degree of a vertex to be sum of weights of its edges:

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

The degrees of all nodes in G are represented by the *degree matrix* $D \in \mathbb{R}^{n \times n}$ given by $D = \text{diag}(\{d_1, d_2, \dots, d_n\})$. Finally, for two sets of vertices $A, B \subseteq V$, we define the weight between these vertex sets as follows.

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

For a vertex set $A \subseteq V$, let $\overline{A} = V \setminus A$ be the complement of A. We will use two different methods to measure the size of A. The first will measure the size of a subset strictly by the number of vertices and the second will consider the weights of the edges contained in A:

$$\begin{aligned} |A| &\coloneqq \text{ the number of vertices in A} \\ \text{vol}(A) &\coloneqq \sum_{i \in A} d_i. \end{aligned}$$

2.2 The Graph Laplacian

We are now equipped to introduce the the graph Laplacian matrix, the primary instrument in spectral clustering. Once again, a more detailed exploration into graph Laplacians can be found in [2], but we will state and discuss relevant properties and their implications.

We use two different definitions of the graph Laplacian:

$$\begin{aligned} &\textit{Unnormalized Graph Laplacian} && L = D - W \\ &\textit{Normalized Random Walk Graph Laplacian} && L_{rw} = I - D^{-1}W \end{aligned}$$

Note that $D^{-1}W$ has rows summing to 1 making it a transition matrix for the similarity graph. A summary of the relevant properties of L and L_{rw} is given by Proposition 2.1. The proof of Proposition 2.1 can be found in Propositions 1 and 3 in [2].

Proposition 2.1. *Properties of L and L_{rw} The matrices L and L_{rw} satisfy the following properties:*

1. *For every $f \in \mathbb{R}^n$, we have*
$$f'Lf = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

2. *L is symmetric.*

3. *L and L_{rw} are positive semi-definite and have n non-negative, real-valued eigenvalues λ_i where $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.*

4. *0 is an eigenvalue of L and L_{rw} and corresponds to the eigenvector $\mathbb{1}$, the constant one vector.*

5. *L_{rw} has eigenvalue λ if and only if λ and the vector u solve the generalized eigenproblem $Lu = \lambda Du$.*

2.3 Basic Graph Spectral Clustering Algorithms

We are now equipped to introduce the graph spectral clustering algorithm. The two graph Laplacians yield two variants, *unnormalized* and *normalized* graph spectral clustering. We outline the pseudocode for these procedures in Algorithms 1 and 2, respectively. As in Section 2.1, we assume a dataset $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ and a similarity function $s(x_i, x_j) \geq 0$ to compare datapoints.

```
Algorithm 1: Unnormalized Spectral Clustering
  Given: Dataset $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$. Similarity function $s(x_i, x_j) \geq 0$. A desired number of
   clusters $k$.
  Result: Clusters $A_1, \cdots, A_k$
    1. Construct a fully connected adjacency matrix for the similarity graph: $W_{ij} = s(x_i, x_j)$. Construct a
       degree matrix $D = \text{diag}(\{d_1 \ldots d_n\})$.
    2. Compute the unnormalized graph Laplacian $L = D - W$.
    3. Compute the first $k$ eigenvectors of $L$ ordered by ascending eigenvalues. Construct a matrix
       $H \in \mathbb{R}^{n \times k}$ whose columns are given by the first $k$ eigenvectors.
    4. Run $k$-means with input parameter $k$ on the rows of $H$. Assign datapoint $x_i$ the cluster value which
       $k$-means has assigned to the $i$th row of $H$.
```

Clustering with the unnormalized graph Laplacian often leads to uninformative clusters containing very few nodes. The alternative Algorithm 2, which uses the random walk Laplacian, avoids this problem through normalization of cluster sizes. The normalizing effects of the random walk Laplacian will be explained in more detail in Section 2.4.

```
Algorithm 2: Normalized Spectral Clustering [3]
  Given: Dataset $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$. Similarity function $s(x_i, x_j) \geq 0$. A desired number of
   clusters $k$.
  Result: Clusters $A_1, \cdots, A_k$
    1. Construct a fully connected adjacency matrix for the similarity graph: $W_{ij} = s(x_i, x_j)$. Construct a
       degree matrix $D = \text{diag}(\{d_1 \ldots d_n\})$.
    2. Compute the unnormalized graph Laplacian $L_{rw} = I - D^{-1}W$.
    3. Compute the first $k$ eigenvectors of $L_{rw}$ ordered by ascending eigenvalues. Construct a matrix
       $H \in \mathbb{R}^{n \times k}$ whose columns are given by the first $k$ eigenvectors.
    4. Run $k$-means with input parameter $k$ on the rows of $H$. Assign datapoint $x_i$ the cluster value which
       $k$-means has assigned to the $i$th row of $H$.
```

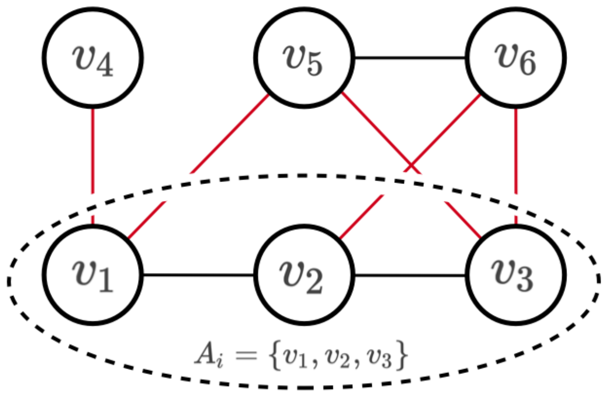


Figure 2: A set A_i of vertices 'cuts' the edges between A_i and $\overline{A_i}$, shown here in red. The value of the cut is the sum of the weights of the red edges.

Both of these algorithms cluster data using a dimensionality reducing embedding derived from the *least significant eigenvectors* of the corresponding graph Laplacian, which is an uncommon strategy in comparison to other techniques like principal component analysis. As we will discuss in the next section, the least significant eigenvectors capture intrinsic factors that minimize an associated graph cut problem, while larger eigenvalues correspond to eigenvectors that are discounted as noise terms.

2.4 Interpreting Graph Spectral Clustering through Normalized Cuts

Graph spectral clustering can be thought of as an approximation of a graph cut problem in which the goal is to group data points with high similarity by removing the connections between data points with low similarity. In other words, we seek to partition the graph so that the total weight of all edges between different clusters is minimized.

Following [2], we define terminology used in this section. First, we define a graph cut. Consider a disjoint partition of the set of vertices $V = \bigcup_{i=1}^{k} A_i$. For any partition $A_1, \ldots, A_k$, we measure the edges eliminated in the cut with

$$\text{cut}(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i). \tag{2.1}$$

The $\frac{1}{2}$ normalizing term results from double counting edges in the undirected graph. In the MinCut problem, we seek to find a partition that minimizes $\text{cut}(A_1, \ldots, A_k)$. In practice, MinCut fails to produce desirable clusters, often isolating singletons from the graph rather than producing clusters with comparable sizes. The normalized cut problem, or NCut, addresses this issue with a new cut based objective which is normalized by the number of nodes in each cluster of a given partition. It is defined as

$$\text{NCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A_i})}{\text{vol}(A_i)}$$

NCut rewards cluster assignments that have a high degree of inter-point similarity within each cluster while points between clusters are dissimilar.

In the rest of this section, we work to show a correspondence between normalized spectral clustering and the NCut problem. We construct indicators $h_j^{(i)}$ to denote if a vertex i belongs to cluster j where

$$h_j^{(i)} = \begin{cases} \frac{1}{\sqrt{\text{vol}(A_j)}} & i \in A_j \\ 0 & i \in \overline{A_j} \end{cases}. \tag{2.2}$$

Now, let $h_j = \left[h_j^{(i)} \quad \cdots \quad h_j^{(n)}\right]^T$ and let $H = [h_1 \; h_2 \; \cdots \; h_k]$ where each column of H corresponds to a different cluster j for $j = 1, \ldots, k$.

These indicators allow us to write the normalized cut cost for an individual cluster in terms of the graph Laplacian. We derive this relationship in detail in Proposition (7.1) of the Appendix.

$$h_j' L h_j = \frac{1}{2} \sum_{i,k \in A} w_{ik} (h_j^{(i)} - h_j^{(k)})^2 = \frac{\text{cut}(A_j, \overline{A_j})}{\text{vol}(A_j)}$$

$$\text{NCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A_i})}{\text{vol}(A_i)} = \text{tr}(H^T L H).$$

Moreover, letting $\text{vol}(A_j) = \sum_{i \in A_j} d_i$, the matrix H satisfies the following constraint in terms of D.

$$h_i^T D h_j = \sum_{k=1}^{n} \mathbb{1}_{\{k \in A_i\}} \mathbb{1}_{\{k \in A_j\}} \frac{d(k)}{\sqrt{\text{vol}(A_i) \cdot \text{vol}(A_j)}} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

The solution to the NCut problem is therefore given by the following constrained optimization problem:

$$\begin{aligned} &\min_H \; \text{tr}(H^T L H) \\ &\text{such that } H^T D H = I_k. \end{aligned}$$

Due to the combinatorial defintion of H, this problem is computationally intractable for large graphs. However, relaxing the problem by allowing entries of H to be continuous yields a simple generalized eigenvector problem. Let K be an unconstrained matrix which will take the place of $D^{\frac{1}{2}} H$. The minimization problem becomes

$$\begin{aligned} &\min_K \text{tr}(K^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} K) \\ &\text{such that } K^T K = I_k \end{aligned}$$

where we have assumed that the degree $\{d(i)\}_{i=1}^n$ of each vertex is nonzero so that D is invertible. By the Rayleigh-Ritz Theorem, the solution is given by a matrix K whose columns correspond to the k least significant eigenvectors of $D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. The unconstrained matrix H' that solves the relaxed version of the original minimization problem is $H' = D^{-\frac{1}{2}} K$. By Proposition 2.1, the columns of H' are the k least significant eigenvectors of L_{rw}.

In the same way that normalized clustering using L_{rw} is a relaxation of NCut, unnormalized clustering using L is a relaxation of the MinCut problem [2]. While network flow algorithms like Push-Relabel can solve MinCut in polynomial time, NCut is NP-Complete [3]. Normalized spectral clustering strikes a balance between computational feasibility and using normalization to improve MinCut and unnormalized spectral clustering. However, it is important to note that this relaxation does not guarantee desirable clusters, a problem which is exacerbated in certain graph structures as seen in [4].

3 Analytic Solutions to PDEs of Interest

After using a chosen similarity function to compute a similarity graph, all clustering information is derived from the graph laplacian matrix. To understand the information carried in this matrix, we will begin by exploring the continuous Laplacian operator, of which the graph laplacian is a discrete approximation. The Laplacian operator is a second order differential operator defined as the divergence of the gradient. In Cartesian coordinates, it has the following form.

$$\Delta f(x^{(1)}, x^{(2)}, \ldots, x^{(n)}) = \nabla \cdot \nabla f = \left(\frac{\partial^2}{\partial (x^{(1)})^2} + \frac{\partial^2}{\partial (x^{(2)})^2} + \cdots + \frac{\partial^2}{\partial (x^{(n)})^2} \right) f$$

The Laplacian operator inherits physical interpretations from the wide variety of physical phenomena governed by partial differential equations (PDEs) involving the Laplacian. In this section, we will study the analytic solutions of two examples, the heat and wave equations.

3.1 Solutions of a Simple Boundary Value Problem

Solutions to the heat and wave equations become physically meaningful when paired with boundary and initial conditions. The combination of a particular partial differential equation with these parameters is known as a *Boundary Value Problem.* When the boundary values are zero, we have *Dirichlet Boundary Conditions.* Here is a simple example of a Dirichlet Boundary Value Problem:

$$\begin{aligned}
&\text{Find } X(x) : [0,1] \to \mathbb{R} \\
&\text{such that } X''(x) = -\lambda X(x) \\
&\text{subject to } X(0) = X(1) = 0,\ X(x) \not\equiv 0
\end{aligned} \tag{3.1}$$

Ignoring the boundary conditions, we can write two candidate solutions in terms of their parameter λ.

$$\begin{aligned}
X_1(x) &= e^{-\sqrt{-\lambda}x} & X_1''(x) &= (-\sqrt{-\lambda})^2 X_1(x) = -\lambda X_1 \\
X_2(x) &= e^{\sqrt{-\lambda}x} & X_2''(x) &= (\sqrt{-\lambda})^2 X_2(x) = -\lambda X_2
\end{aligned}$$

In the special case when $\lambda = 0$, there is an additional solution to consider.

$$X_3(x) = x \qquad\qquad X_3''(x) = 0$$

These candidate solutions are chosen to be linearly independent, such that when $\lambda \neq 0$ we have $AX_1(x) + BX_2(x) = 0$ only if $A = B = 0$. The same holds for $AX_1(x) + BX_2(x) + CX_3(x) = 0$ when $\lambda = 0$. Other function which are linearly dependent are themselves candidate solutions, as any linear combination of candidate solutions is also a candidate solution. Depending on the sign of λ, the candidate solutions will then be either exponential, linear, or periodic. We will evaluate these cases to determine the subset of solutions satisfying the boundary conditions:

1. $\lambda < 0$: then $\sqrt{-\lambda} \in \mathbb{R}^+$, and this implies $X_1(x) > 0$ and $X_2(x) > 0$ for all $x \in \mathbb{R}$. Furthermore, $AX_1(x)$ and $-BX_2(x)$ intersect for at most one point unless $A = B = 0$, so the only solution meeting the boundary conditions is uniformly zero.

2. $\lambda = 0$: then $X_1(x) = X_2(x) = 1$ and $AX_1(x) + CX_3(x) = A + Cx$. To satisfy the boundary conditions, $A = C = 0$ is required. Again, solutions meeting the boundary conditions are uniformly zero or nonexistent.

3. $\lambda > 0$: in this case, solutions exist, but only for a countable set of λ. For convenience, let $\omega = \sqrt{\lambda}/\pi$. Supposing the conditions are met,

$$\begin{aligned}
X(x) &= Ae^{i\omega\pi x} + Be^{-i\omega\pi x} \\
X(0) = 0 &\implies A + B = 0 \implies B = -A \\
X(1) = 0 &\implies Ae^{i\omega\pi} - Ae^{-i\omega\pi} = 0 \\
&\implies e^{i\omega\pi} = e^{-i\omega\pi} \implies \omega \in \mathbb{Z}
\end{aligned}$$

   The last equality holds because $\exp(i\omega\pi)$ and $\exp(-i\omega\pi)$ are complex conjugates and can only be equal if they are real. To be real, the complex phase $\omega\pi$ must be an integer multiple of π.

Hence when $\lambda > 0$, the solution $X(x)$ either takes the form $e^{in\pi x}$, $n \in \mathbb{Z}$ or a linear combination of these functions:

$$X(x) = \sum_{n=-\infty}^{\infty} A_n e^{in\pi x}$$

Interestingly, the solutions to this differential equation are Fourier basis functions, whose linear combinations are dense in the vector space of square integrable functions. Not only are linear combinations of this form solutions to the boundary value problem, but any solution is a (possibly infinite) linear combination of these special solutions. Moreover, under the inner product $\langle f, g \rangle = \int f(x)g(x)\,dx$ these solutions are orthonormal:

$$\langle e^{im\pi x}, e^{in\pi x} \rangle = \int_0^1 e^{i(n-m)x}\,dx = \begin{cases} 1 & m = n \\ 0 & m \neq n \end{cases}$$

In this case, the second order infinite dimensional linear operator admits an orthonormal basis of eigenvectors much like a finite dimensional linear operator. This behavior is shared by a wide variety of boundary value problems for second order PDEs, and is characterized by the *Sturm Liouville Theory.* We refer the interested reader to [5] for more details of this theory.

3.2 Solutions to the Heat and Wave Equations

3.2.1 Heat Equation

The heat equation is a partial differential equation that describes how heat diffuses in a solid medium over time. In the following, we use $\Delta_x u(x,t)$ to denote the partial Laplacian operator applied only to the x input coordinates and $f(x)$ to denote initial conditions.

Heat Equation Find $u(x,t) : [0,1] \times \mathbb{R}^+ \to \mathbb{R}$

such that $\frac{\partial}{\partial t} u(x,t) = c\Delta_x u(x,t)$

subject to $u(x,0) = f(x)$ and $u(0,t) = u(1,t) = 0$

The constant c is known as the heat conductivity, and for simplicity we may assume $c = 1$. We will solve for $u(x,t)$ analytically using the method of separation of variables. Assuming $u(x,t) = X(x)T(t)$, the heat problem has the following form.

$$X(x)T'(t) = X''(x)T(t) \tag{3.2}$$

$$X(0)T(t) = X(1)T(t) = 0 \tag{3.3}$$

$$X(x)T(0) = f(x) \tag{3.4}$$

We can rearrange (3.2) to isolate the time and space dependent functions. Because they vary independently, both sides of the following equality must be a constant. We will call it $-\lambda \in \mathbb{R}$ to match the convention in Equation (3.1).

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} := -\lambda \text{ for some } \lambda \in \mathbb{R}$$

We have essentially reduced this PDE into solving two simpler ODEs.

$$T'(t) = -\lambda T(t)$$
$$X''(x) = -\lambda X(x).$$

Looking at the boundary condition $X(0)T(t) = 0$, we can conclude that either $X(0) = 0$ or $T(t) = 0$. We discard the case $T(t) = 0$ because this implies that $u(x,t) = 0$ for all x and t, the trivial case, so we look at when $X(0) = 0$ and $X(1) = 0$. Like the basic boundary value problem, a candidate solution $X(x)$ is the complex exponential $e^{ik\pi x}$ and we can build the Fourier expansion of $f(x)$ using these functions.

$$f(x) = \sum_{n=-\infty}^{\infty} A_n X^{(n)}(x) = \sum_{n=-\infty}^{\infty} A_n e^{in\pi x}$$

Each of the A_n is a Fourier coefficient and is therefore given by

$$A_n = \int_0^1 f(x) e^{in\pi x} dx.$$

From our analysis in (3.1), setting $\lambda = n^2\pi^2$ for any $n \in \mathbb{Z}$ yields nonzero solutions in $X(x)$. Hence the nonzero solutions take the form $e^{n^2\pi^2 t} e^{in\pi x}$ or a (possibly infinite) linear combination of these functions.

$$u(x,t) = \sum_{n=-\infty}^{\infty} X^{(n)}(x) T^{(n)}(t) = \sum_{n=-\infty}^{\infty} A_n e^{n^2\pi^2 t} e^{in\pi x} \tag{3.5}$$

We prove the uniqueness of this solution when the boundary and initial conditions are met in Section 7.2 of the appendix. Intuitively, this equation indicates that heat diffusion acts by decaying each frequency component of the initial conditions. Higher frequencies, for which $n\pi$ is large, decay exponentially faster than lower frequencies. These high frequencies induce regions of heat which are significantly hotter or colder than their surroundings and so they diffuse away quickly.

3.2.2 Wave Equation

The one-dimensional wave equation is a boundary value problem which has the following setup:

$$\begin{aligned} &\textit{Wave Equation} \qquad && \text{Find } u(x,t) : [0,1] \times \mathbb{R}^+ \to \mathbb{R} \\ & && \text{such that } \frac{\partial^2}{\partial t^2} u(x,t) = c\Delta_x u(x,t) \\ & && \text{subject to } u(x,0) = f(x),\ u(0,t) = u(1,t) = 0 \end{aligned}$$

The wave equation also has a parameter c, known as the wave propagation speed. We will assume in this case that $c = 1$ as well. Finding the solution to the wave equation follows a similar procedure as the heat equation. First, we assume $u(x,t) = X(x)T(t)$ is separable into spatial and time dependent components, and then we apply the wave equation.

$$\begin{aligned} \frac{\partial^2}{\partial t^2} X(x)T(t) = \Delta_x u(x,t) &\implies X(x)T''(t) = X''(x)T(t) \\ \frac{X(x)}{X''(x)} = \frac{T(t)}{T''(t)} &= -\lambda \\ &\implies \begin{cases} T''(t) = -\lambda T(t) \\ X''(x) = -\lambda X(x) \end{cases} \end{aligned}$$

In this case, both $X(x)$ and $T(t)$ match the form of (3.1) with boundary data given for $X(x)$ by the initial conditions $f(x)$ and for $T(t)$ by Dirichlet conditions. Hence the solution $u(x,t)$ takes the following form.

$$f(x) = \sum_{n=-\infty}^{\infty} X^{(n)}(x) = \sum_{n=-\infty}^{\infty} A_n e^{in\pi x} \text{ where } A_n = \int_0^1 f(x) e^{in\pi x} dx$$

$$u(x,t) = \sum_{n=-\infty}^{\infty} X^{(n)}(x)T^{(n)}(t) = \sum_{n=-\infty}^{\infty} A_n e^{n\pi t} e^{in\pi x}$$

In contrast to the heat equation, frequency components of $u(x,t)$ do not decay in time. Instead, they oscillate in time and remain eternally excited. In Section 4.3.2, we harness this property to find eigenvectors of the discrete Laplacian.

4 Graph Spectral Clustering as a Diffusion Process

Recall the procedure for spectral clustering: first, construct a Laplacian matrix from the similarity graph of a data set. Then, run k-means on rows of the least significant eigenvectors to determine cluster membership for each point. We may view the similarity graph as a metric on the discrete domain of graph vertices V. Real functions on the graph are maps $f : V \to \mathbb{R}$ and on the space of such functions the negative graph Laplacian $-L$ acts takes the place of the Laplacian operator for functions over a continuous domain. It is from this view that we will deduce the physical interpretation of graph spectral clustering.

4.1 Discretization of a PDE into a Graph

To simulate the heat equation on a graph, we can treat the graph as a discretization of space where the distance between two points is given by their edge weight in the similarity graph. In the case of a simple line graph, for which the similarity function between nodes is binary, the identification of the graph laplacian with a Laplacian operator is motivated by the second difference approximation of the second derivative:

$$f''(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - 2f(x) + f(x-\epsilon)}{\epsilon^2}$$

Suppose we are given data $\{x_i\}_{i=1}^n$. Denote by $f(x_i)$ the heat at node x_i. Representing f by a vector of values for each vertex, matrix multiplication by the line graph Laplacian computes the second difference approximation:

$$L = D - W = \begin{bmatrix} 1 & -1 & 0 & 0 & \\ -1 & 2 & -1 & 0 & \\ 0 & -1 & 2 & -1 & \\ & & & & \ddots \end{bmatrix} \qquad -[\epsilon^{-2}Lf]_i = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{\epsilon^2},\ 1 < i < n \quad (4.1)$$

In this case, the line graph is a discretization of the unit interval. For more complicated graphs, where the similarity function can takes arbitrary nonnegative values, we will assume that the rate of heat transfer between two nodes is proportional to the similarity between these nodes, which is given by their edge weight. This gives us

$$\frac{\partial f_i}{\partial t} = \kappa \sum_{j:(i,j)\in E} (f_j - f_i) w_{ij}$$

where $\kappa < 1$ is a proportionality constant. Building the Laplacian $L \in \mathbb{R}^{n\times n}$ edge by edge, we have

$$L = D - W = \sum_{(i,j)\in E}^{n} (e_i - e_j)(e_i - e_j)^T w_{ij}$$

where each e_i is a canonical basis vector. From this expansion, it follows that $\partial f_i / \partial t$ is given by $[-\kappa L f]_i$. In matrix notation,

$$\frac{\partial f}{\partial t} = -\kappa L f$$

By recreating heat diffusion on the domain induced by a similarity graph, we have shown how the graph Laplacian plays the role of the Laplacian operator on this discrete domain.

Because the second derivative computes curvature, it reflects the amount of 'dissimilarity' of the value of f at a point from the value of f around the point. In the same way, the graph Laplacian computes the dissimilarity between each vertex and its weighted neighbors according to the data similarity graph. As discussed in Section 3.2, the continuous heat equation acts on a set of initial conditions by diffusing regions of high heat towards regions of low heat. In terms of a Fourier basis expansion, the heat operator acts on each frequency component by scaling, much like the action of a matrix on its eigenvectors:

$$u(x,t) = \sum_{n=\infty}^{\infty} A_n e^{-in^2\pi^2 t} e^{in\pi x}$$

The regions of f with high local dissimilarity are induced by the components of f with high frequency, and these regions diffuse the fastest. The eigenvectors of the graph Laplacian play the same role on the domain induced by the graph as components which decay the fastest when present in initial conditions:

$$\frac{\partial f}{\partial t} = -\kappa L f \implies f_{t+1} = f_t - \kappa L f \implies f_t = (I - \kappa L)^t f$$

$$\implies f_t = \sum_{i=1}^{n} (1 - \kappa\lambda_i)^t \langle f, v_i \rangle v_i$$

where $\lambda_i \in [0,1]$ and v_i are eigenvalues and eigenvectors respectively. Given this connection, one can even define a "Fourier basis" for functions on a graph using the graph Laplacian eigenvectors.

4.2 Applying Continuous Solutions of PDEs to Data

In the previous section, we defined data points as heat sources to be the initial conditions of the heat equation. The Laplace operator is a linear differential operator, and its solution is given by convolving the initial conditions with the Laplace operator's associated Green's function, G.

Given a dataset $\mathbf{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, we want to find a convex domain $\Omega \supset \mathbf{X}$ so that we can define boundary conditions for the heat equation. We find Ω so that $\mathbf{X}$ is strictly in its interior. Now, we define the initial conditions to be the data framed as the sum of n Dirac functions, i.e. $\delta(\mathbf{X}) \coloneqq \Sigma_{x_i \in \mathbf{X}} \delta(x - x_i)$. We now solve the heat equation by assuming zero Dirichlet boundary conditions and setting the initial condition $u(\Omega, 0) = \delta(\mathbf{X})$.

Definition 4.1. *The Green's function G for a linear differential operator $\mathcal{D}_x$ is the function that solves the equation $\mathcal{D}_x G(x,t) = \delta(x-t)$.*

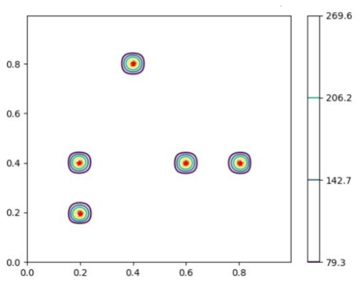
We can apply the superposition principle to our data, which is a sum of Dirac functions, to find the solution to the heat equation as a sum of Green's function. The Green's function for the heat equation is given by

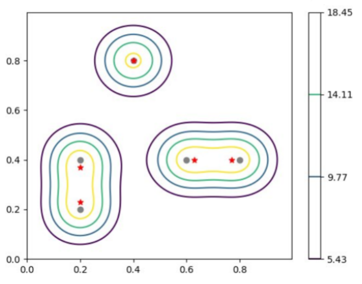
$$G(x,t) = \left(\frac{1}{4\pi t}\right)^{d/2} \exp\left(-\frac{\|x\|_2^2}{4t}\right). \tag{4.2}$$

This gives us the following solution to the heat equation:

$$u(x,t) = \left(\frac{1}{4\pi t}\right)^{d/2} \int_\Omega \delta(\mathbf{X}) \exp\left(-\frac{\|x-y\|^2}{4t}\right) dy^{(1)} \ldots dy^{(d)}. \tag{4.3}$$

We can now view solutions to the heat equation as convolution with a gaussian function, which is exactly applying a gaussian kernel smoother to the dataset, where time is the parameter for kernel radius. As the time goes on, the length scale of the input space increases and we see more drastic smoothing of the data, which corresponds to finding cluster assignments to smaller values of k.


(a) $u(x, t = 0)$


(b) $u(x, t = 0.0041)$

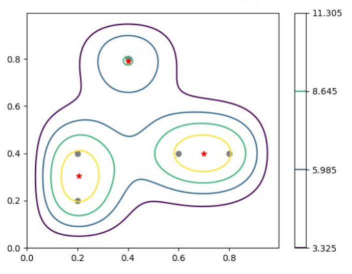

(c) $u(x, t = 0.0091)$

Figure 3: Contour maps of u. Red stars indicate local maxima.

4.3 Physical Interpretations of Spectral Clustering

4.3.1 Clustering with the Continuous Heat Equation

Earlier, we saw how the negative Laplacian matrix $-L$ is the discrete form of the continuous Laplace operator Δ, which can be leveraged to find numerical solutions to the heat equation on a graph. In this section we will explore how to find clusters by evolving the heat equation on a graph created from initial data points as a comparison point to k-means. We now solve the heat equation by assuming zero Dirichlet boundary conditions, and setting the initial condition $u(\Omega, 0) = \delta(\mathbf{X})$.

We can use the continuous time evolution of the heat equation to cluster the data. For a value of heat y and time T, we define the clusters for some dataset $\mathbf{X}$ by counting the separable components in the connected domain Ω for the solution set $u(x, T) \geq y$. Then we intersect each component individually with $\mathbf{X}$ to find cluster assignments.

As shown in Figure 3, the initial conditions give a trivial clustering of the dataset with n clusters. As time evolves the heat equation, the boundary conditions enforce that the heat of the entire region decays to zero. Thus after a sufficient amount of time, any ϵ-level set will be a solution to clustering the dataset with one cluster. This solution is continuous with time, so plotting various level sets as time changes will yield various solutions to clustering a data set with k clusters, with k decreasing from n to one.

Observe that the initial conditions are also the local (spatial) maxima of u. In the following results, we will examine the relations between local maxima among solution sets and cluster assignments of the data. The following graphics are time steps of evolving the heat equation with a dataset of five two-dimensional data points over a unit square $\Omega = [0, 1]^2$.

The purple level sets begin with n separable components. As time evolves, the components merge together to create new cluster assignments for a smaller value of k. At certain points in time, the local maxima of the heat equation match directly to the centroids of a k-means algorithm. We see this correspondence in Figure 2 and Figure 4, but not in Figure 3.

4.3.2 Clustering with the Discrete Wave Equation

The authors of [6] introduce a spectral clustering algorithm (3) which simulates wave propagation on a graph to extract information about its laplacian eigenvectors. This information can be used to cluster nodes of the graph. We give a brief overview of the algorithm with more detail and derivations available in Section 7.2 of the Appendix.

Algorithm 3: Finding Eigenvectors from Wave Diffusion [6]

```
Given: a similarity matrix S $\in R^{n\times n}$, $\leq 2^k$ clusters
Result: cluster number for each vertex.
    1. Initialize each vertex with a random initial value
    2. Set $\boldsymbol{u}(-1) := \boldsymbol{u}(0)$
    3. Evolve the system according to the discretized heat equation for $T_{max}$ steps
    4. For each node, apply the FFT to displacement over time at that node.
    5. For each node i, find the first k values where the magnitude of the frequency is a peak (should pick
       out the same frequencies for all nodes). If the real part is positive, set the ith entry for the jth
       eigenvector ($j = 1, \ldots, k$) to 1 and 0 otherwise.
    6. Using the binary representation of each row (node), we can find cluster assignments for each vertex.
```

We saw in 3.2.2 that in the continuous solution of the wave equation, the eigenvectors of the Laplace operator do not die out as t grows large. Wave propagation on a graph domain is also governed by the Fourier decomposition, and if we allow random initial conditions for each vertex to evolve until their frequency content stabilizes, we can apply the FFT to extract eigenvector signs.

We treat each vertex in the graph as a random variable and assign it a random number. We want to evolve this system according the wave equation because we know the eigenvectors of the wave equation stay eternally excited. We do this by discretizing the wave equation and the evolution can be written as a matrix system

$$\boldsymbol{z}(t) = \begin{pmatrix} \boldsymbol{w}(t) \\ \boldsymbol{w}(t-1) \end{pmatrix} = \begin{pmatrix} 2I - \mathcal{L} & -I \\ I & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{w}(t-1) \\ \boldsymbol{w}(t-2) \end{pmatrix} =: M\boldsymbol{z}(t-1) \text{ where } \boldsymbol{w}(t) = \begin{pmatrix} w_1(t) \\ \vdots \\ w_n(t) \end{pmatrix}$$

The eigenvectors of this new matrix M form a complete subspace aside from the vector $(\mathbf{1}\ \ -\mathbf{1})^T$, so we choose let $\boldsymbol{w}(-1) = \boldsymbol{w}(0)$. The other eigenvectors of M have eigenvector and eigenvalue pairs that can be written in terms of the eigenvector and eigenvalues of $\mathcal{L}$. In particular, the eigenvectors are

$$\boldsymbol{m}^{(j)}_{+/-} = \begin{pmatrix} \alpha^{(j)}_{+/-}\boldsymbol{v}^{(j)} \\ \boldsymbol{v}^{(j)} \end{pmatrix} \text{ and the corresponding eigenvalue } \alpha^{(j)}_{+/-} \text{ where } \alpha^{(j)}_{+/-} = \frac{2 - \lambda_j \pm \sqrt{\lambda_j^2 - 4\lambda_j}}{2}.$$

Each $\alpha^{(j)}_{+} = e^{i\omega_j}$ and $\alpha^{(j)}_{-} = e^{-i\omega_j}$ for some ω_j because $|\alpha^{(j)}_{+/-}| = 1$ and the square root term is imaginary. Notice that the larger smaller eigenvalues of $\mathcal{L}$ correspond with small ω_j. After some calculation, we find that

$$\boldsymbol{w}(t) = 2\sum_{j=1}^{n} A_j \boldsymbol{v}^{(j)}\cos(2\omega_j t) - B_j \boldsymbol{v}^{(j)}\sin(2\omega_j t)$$

For node i, we have a signal that evolves with time as $[w_i(0), \ldots, w_i(T_{max})]$. The frequency peaks of FFT applied to $[w_i(0), \ldots, w_i(T_{max})]$ corresponds to the i^{th} entry of the j^{th} eigenvector. They will be of the form $(A_j + iB_j)v_i^{(j)}$. $A_j + iB_j$ remains the same for all nodes in the j^{th} eigenvector, so the sign of real component allows us to find the sign of $v_i^{(j)}$. Since graph spectral clustering can be thought of as a relaxation of indicator vectors where the i^{th} column is roughly an indicator vector for the i^{th} cluster, this algorithm gives us a method to cluster the vertices.

5 Experimental Results

We demonstrate various applications of Graph Spectral Clustering on both synthetic and real-world datasets. First, we show in Figure 4 a comparison of Graph Spectral Clustering to the KMeans++ algorithm [7] for two synthetic datasets. The first dataset, shown in the first row, is a mixture of eight Gaussians, which both algorithms can cluster perfectly. In the second row, we show clustering performance on the synthetic Two Moons dataset. Due to the manifold structure of the data, KMeans++ provides a suboptimal clustering, while Graph Spectral Clustering can identify and separate the two moons.

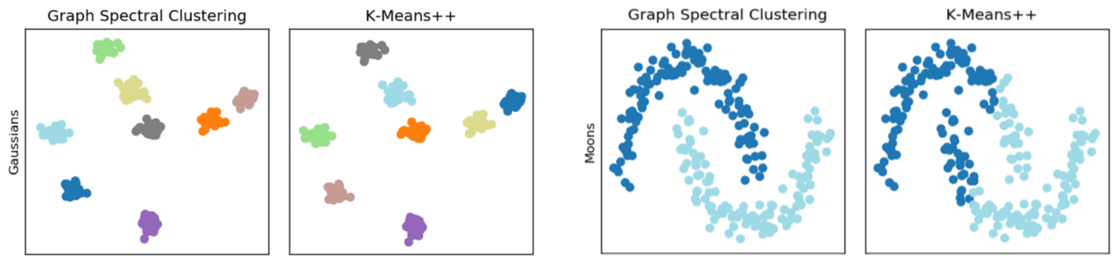


Figure 4: Comparison of KMeans++ to Graph Spectral Clustering on two synthetically generated datasets.

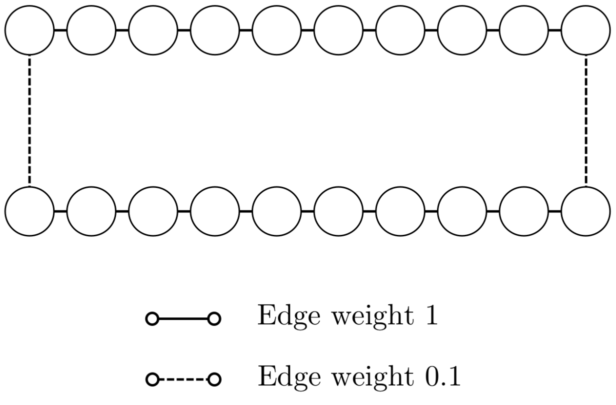
5.1 Clustering Using the Wave Equation

To further explore the clustering properties of the graph Laplacian eigenvectors, we replicate the results of the wave equation based clustering algorithm presented in [6]. Like the heat equation, solutions to the wave equation are goverened by a Laplacian operator. Unlike the heat equation, these solutions need not exhibit time decay. By propagating a wave on a graph, one can compute Fourier coefficents of the long time solutions to the wave equation, which can in turn be used to deduce clustering information given by the Laplacian eigenvectors. As shown in Figure 5, this wave based clustering algorithm can recover simple clusters like those of a line graph.

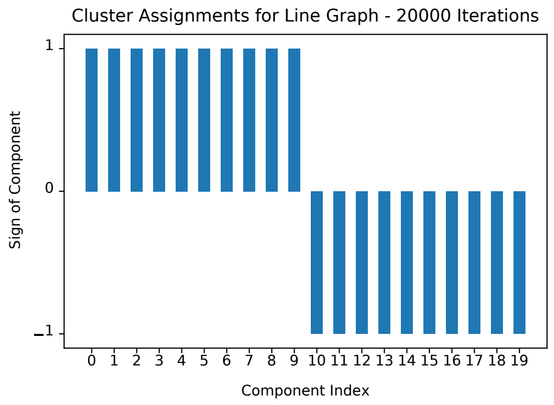
6 Conclusion

In the Two Moons dataset shown in Figure 4, certain points at the ends of each moon lie inside the arc of the other moon, close in Euclidean distance to points from the opposite cluster. This dataset is a two dimensional example of a key difficulty in clustering high dimensional data: when the points lie approximately on a manifold, pairwise Euclidean distances can become uninformative at all but the smallest scales. Algorithms like k-Means, which assume that nearby points in Euclidean distance ought to be clustered together, struggle to cluster manifold data. In contrast, graph spectral clustering uses the paradigm that *points should be clustered together if they can quickly diffuse heat to each other*. Drawing on connections to minimum graph cuts, this approach is favorable when the data has high inter-cluster similarity and low cross-cluster similarity.

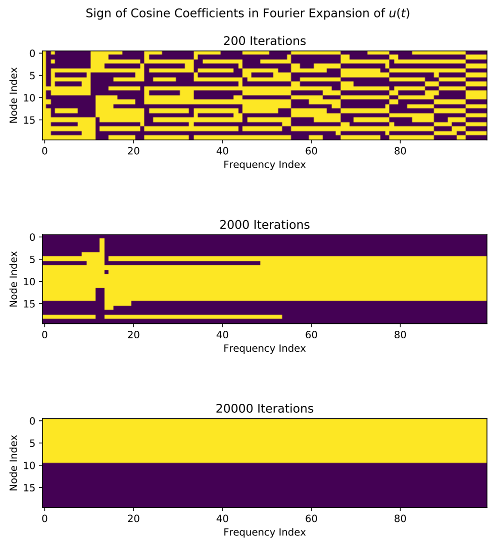
Despite the usefulness of this clustering algorithm, it is computationally expensive, as even computing the similarity graph laplacian is $O(n^2)$ for n datapoints. Consequently, computationally cheaper algorithms like t-SNE, which has been shown to approximate graph spectral clustering under certain hyperparameter choices [8, 9], are preferred for large datasets. Finding new and cheaper methods for data clustering, potentially through diffusion or other physically motivated clustering paradigms, is an interesting area for future research.



(a) The data graph. Dotted edges have weight 0.1 while solid edges have weight 1. The graph has two clusters by design.



(b) Starting from random initial conditions, wave propagation can be used to recover the signs of the components of the second laplacian eigenvector, shown here. This information can be used to cluster the data.



(c) After more time has passed, the signs of Fourier expansion coefficients become better resolved and can be used for clustering.

Figure 5: Using the wave equation to cluster data on a line graph.

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation, 15(6):1373–1396, 2003.

[2] Ulrike von Luxburg. A tutorial on spectral clustering. Statistics and Computing, 17(4):395–416, Dec 2007. arXiv: 0711.0189 [cs.DS].

[3] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888–905, 2000.

[4] Stephen Guattery and Gary L. Miller. On the performance of spectral graph partitioning methods. In Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '95, page 233–242, USA, 1995. Society for Industrial and Applied Mathematics.

[5] Gerald Teschl. Ordinary Differential Equations and Dynamical Systems. American Mathematical Soc., Aug 2012. Google-Books-ID: FZ0CAQAAQBAJ.

[6] Tuhin Sahai, Alberto Speranzon, and Andrzej Banaszuk. Hearing the clusters in a graph: A distributed algorithm. arXiv:0911.4729 [physics], Apr 2011. arXiv: 0911.4729.

[7] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab, June 2006.

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. Journal of Machine Learning Research, 9:2579–2605, 2008.

[9] George C. Linderman and Stefan Steinerberger. Clustering with t-sne, provably. SIAM Journal on Mathematics of Data Science, 1(2):313–332, 2019.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[11] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In Applications of Data Mining in Computer Security. Kluwer, 2002.

[12] Ravi P. Agarwal and Donal O'Regan. An Introduction to Ordinary Differential Equations. Springer New York, 2008.

[13] Michele Benzi, Daniele Bertaccini, Fabio Durastante, and Igor Simunec. Nonlocal network dynamics via fractional graph laplacians. arXiv:1912.07288 [physics], May 2020. arXiv: 1912.07288.

[14] Yongxin Chen, Tryphon T. Georgiou, and Allen Tannenbaum. Optimal transport for gaussian mixture models. arXiv:1710.07876 [cs, math], Jan 2018. arXiv: 1710.07876.

[15] Joel Friedman and Jean-Pierre Tillich. Wave equations for graphs and the edge-based laplacian. Pacific Journal of Mathematics, 216(2):229–266, Oct 2004.

[16] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the poisson equation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12):1991–2005, 2006.

[17] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888–905, Aug 2000. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[18] Anna Lischke, Guofei Pang, Mamikon Gulian, Fangying Song, Christian Glusa, Xiaoning Zheng, Zhiping Mao, Wei Cai, Mark M. Meerschaert, Mark Ainsworth, and et al. What is the fractional laplacian? arXiv:1801.09767 [math], Jan 2018. arXiv: 1801.09767.

[19] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: analysis and an algorithm. In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, NIPS'01, page 849–856. MIT Press, Jan 2001.

[20] A. P. Riascos and José L. Mateos. Fractional dynamics on networks: Emergence of anomalous diffusion and lévy flights. Physical Review E, 90(3):032809, Sep 2014. arXiv: 1506.06167.

[21] Marco Saerens, Francois Fouss, Luh Yen, and Pierre Dupont. The Principal Components Analysis of a Graph, and Its Relationships to Spectral Clustering, volume 3201, page 371–383. Springer Berlin Heidelberg, 2004. Series Title: Lecture Notes in Computer Science.

[22] Daniel Ting, Ling Huang, and Michael I. Jordan. An analysis of the convergence of graph laplacians. In Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, page 1079–1086. Omnipress, Jun 2010.

[23] Xiao Wen, Wai Sun Don, Zhen Gao, and Yulong Xing. Entropy stable and well-balanced discontinuous galerkin methods for the nonlinear shallow water equations. Journal of Scientific Computing, 83(3):66, Jun 2020.

[24] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. page 8.

[25] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[26] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

7 Appendix

7.1 Additional Proofs

Proposition 7.1 (NCut and the Graph Laplacian)**.** *The NCut objective can be written in terms of the graph Laplacian:*

$$h_j^T L h_j = \frac{cut(A_j, \overline{A_j})}{vol(A_j)}$$

where $h_j \in \mathbb{R}^n$ is an indicator vector for cluster membership, given by the following.

$$h_j^{(i)} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}} & i \in A_j \\ 0 & i \notin A_j \end{cases}$$

Proof.

$$\begin{aligned} h_j^T L h_j &= h_j^T D h_j - h_j^T W h_j \\ &= \frac{1}{2} \sum_{i,k=1}^{n} w_{ik} (h_j^{(i)} - h_j^{(k)})^2 \\ &= \sum_{i \in A_j, k \in \overline{A}_j} w_{ik} (h_j^{(i)} - h_j^{(k)})^2 \\ &= \sum_{i \in A, k \in \overline{A}_j} w_{ik} \left(\frac{1}{\text{vol}(A_j)} \right) \\ &= \frac{\text{cut}(A_j, \overline{A}_j)}{\text{vol}(A_j)} \end{aligned}$$

■

Proposition 7.2 (Uniqueness of the Solution to the One Dimensional Heat Equation)**.** *For given set of initial conditions $f(x)$, the one dimensional heat equation with Dirichlet boundary conditions has a unique solution $u(x,t)$ which satisfies $u(x,0) = f(x)$.*

Recall our formulation of the heat equation with Dirichlet boundary conditions, diffusion constant $c > 0$, and initial conditions $f(x)$. As in the text, we will assume for the proof that units are chosen in which $c = 1$.

$$\begin{aligned} \textit{Heat Equation} \qquad & \text{Find } u(x,t) : [0,1] \times \mathbb{R}^+ \to \mathbb{R} \\ & \text{such that } \frac{\partial}{\partial t} u(x,t) = c \Delta_x u(x,t) \\ & \text{subject to } u(x,0) = f(x) \text{ and } u(0,t) = u(1,t) = 0 \end{aligned}$$

Proof. Suppose $u_1(x,t)$ and $u_2(x,t)$ are two solutions to the heat equation. We will show that $v(x,t) = u_1(x,t) - u_2(x,t)$ has $v(x,t) = 0$ for all $x \in [0,1], t \in \mathbb{R}^+$. This would imply that $u_1(x,t) = u_2(x,t)$ at all points in the domain.

Let v_x and v_{xx} denote first and second partial derivatives respectively. Under the assumptions, v satisfies the heat equation with initial conditions $g(x) = 0$.

$$\begin{aligned} v_t &= u_{1t} + u_{2t} = u_{1xx} + u_{2xx} = v_{xx} \\ v(x,0) &= u_1(x,0) - u_2(x,0) = 0 = g(x) \\ v(0,t) &= v(1,t) = 0 \end{aligned}$$

Let $V(t) = \int_0^1 v^2(x,t)\,dx$. By monotonicity, $V(t) \geq 0$ for all $t > 0$. Differentiating in t,

$$\begin{aligned}
V_t(t) &= \int_0^1 2v(x,t)v_t(x,t)dx \\
&= \int_0^1 2v(x,t)v_{xx}(x,t)dx \\
&= 2c\left(vv_x|_0^1 - \int_0^1 v_x^2\,dx \right) \\
&= -c\int_0^1 v_x^2\,dx \leq 0
\end{aligned}$$

We have shown $V_t(t) \leq 0$ for all $t > 0$. By the mean value theorem, $V(t) \leq V(0) = 0$ for all $t > 0$, but by its definition, $V(t) \geq 0$. Hence $V(t) = 0$ for all $t > 0$, and so continuity (implied by differentiability) of v guarantees that $v(t) = 0$ for all $t > 0$. ■

7.2 Wave Equation Clustering

In this section we illustrate the algorithm proposed in [6] for clustering nodes in a graph through wave propagation on the graph. The key idea is to evolve wave propagation on the graph, starting from random initial conditions, and to then use the time evolved system to extract information about the graph laplacian matrix. For the ith node, the algorithm extracts the signs of the ith components of each of the first k least significant eigenvectors, which determine cluster membership of the node. These signs are extracted through a Fourier Transform in time of the displacement at each node.

Let L be the graph Laplacian matrix. We can discretize the wave equation into

$$w^{(i)}(t) = 2w^{(i)}(t-1) - w^{(i)}(t-2) - c^2 L_i w(t-1).$$

For this system to be numerically stable, $0 < c < \sqrt{2}$ is required – see [6] for details. For the sake of simplicity, we let $c = 1$. Let

$$\boldsymbol{w}(t) = \begin{pmatrix} w^{(1)}(t) \\ \vdots \\ w^{(n)}(t) \end{pmatrix} \text{ and } \boldsymbol{z}(t) = \begin{pmatrix} \boldsymbol{w}(t) \\ \boldsymbol{w}(t-1) \end{pmatrix}.$$

Then, we can rewrite the wave equation evolution as a matrix system

$$\boldsymbol{z}(t) = \begin{pmatrix} \boldsymbol{w}(t) \\ \boldsymbol{w}(t-1) \end{pmatrix} = \begin{pmatrix} 2I - L & -I \\ I & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{w}(t-1) \\ \boldsymbol{w}(t-2) \end{pmatrix},$$

so given some initial state $\boldsymbol{z}(0)$,

$$\boldsymbol{z}(t) = \begin{pmatrix} 2I - L & -I \\ I & 0 \end{pmatrix}^t \boldsymbol{z}(0) := \boldsymbol{M}^t \boldsymbol{z}(0).$$

Let $\boldsymbol{v}_j$ denote the eigenvector corresponding to the jth smallest eigenvalue, λ_j, of L. It turns out that the pair of eigenvectors of $\boldsymbol{M}$ corresponding to $\boldsymbol{v}_j$ is $\boldsymbol{m}_{j\pm} = \begin{pmatrix} \alpha_{j\pm}\boldsymbol{v}_j \\ \boldsymbol{v}_j \end{pmatrix}$, with eigenvalues $\alpha_{j\pm}$ where $\alpha_{j\pm} =$

$\dfrac{2-\lambda_j \pm \sqrt{\lambda_j^2 - 4\lambda_j}}{2}$. This expression for $\alpha_{j\pm}$ comes from applying the definition of eigenvalue-eigenvector pairs:

$$M\begin{pmatrix}\alpha_j \boldsymbol{v}_j \\ \boldsymbol{v}_j\end{pmatrix} = \alpha_j \begin{pmatrix}\alpha_j \boldsymbol{v}_j \\ \boldsymbol{v}_j\end{pmatrix}$$

$$\begin{pmatrix}\left[(2I-L)\,\alpha_j - I\right]\boldsymbol{v}_j \\ \alpha_j \boldsymbol{v}_j\end{pmatrix} = \begin{pmatrix}\alpha_j^2 \boldsymbol{v}_j \\ \alpha_j \boldsymbol{v}_j\end{pmatrix} \tag{7.1}$$

Now looking at the first entry of (7.1), we get the expression

$$\left[(2I-L)\,\alpha_j - I\right]\boldsymbol{v}_j = \alpha_j^2 \boldsymbol{v}_j$$

Then because $\boldsymbol{L}\boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j$ and $\boldsymbol{I}\boldsymbol{v}_j = \boldsymbol{v}_j$,

$$\left[(2-\lambda_j)\,\alpha_j - 1\right]\boldsymbol{v}_j = \alpha_j^2 \boldsymbol{v}_j.$$

Applying the quadratic formula on $(2-\lambda_j)\,\alpha_j - 1 = \alpha_j^2$ and solving for $\alpha_{j\pm}$, we get

$$\alpha_{j\pm} = \frac{2-\lambda_j \pm \sqrt{\lambda_j^2 - 4\lambda_j}}{2}.$$

Let the eigenvectors of M be $\boldsymbol{m}_{j\pm} = \begin{pmatrix}\alpha_{j\pm}\boldsymbol{v}_j \\ \boldsymbol{v}_j\end{pmatrix}$. When we have stability, i.e. $0 < c < \sqrt{2}$, the quantity inside the square root is negative because the largest possible eigenvalue of a normalized laplacian is also 2, so we can express $\boldsymbol{m}_j = \boldsymbol{p}_j \pm i\boldsymbol{q}_j$ where $\boldsymbol{p}_j = \begin{pmatrix}\text{Real}(\alpha_{j+})\boldsymbol{v}_j \\ \boldsymbol{v}_j\end{pmatrix}$ and $\boldsymbol{q}_j = \begin{pmatrix}\text{Imag}(\alpha_{j+})\boldsymbol{v}_j \\ \boldsymbol{0}\end{pmatrix}$. Notice too that $|\alpha_{j\pm}| = 1$.

$$\begin{aligned}|\alpha_{j\pm}| = \alpha_{j-} + \alpha_{j-} &= \left(\frac{2-\lambda_j}{2}\right)^2 - \left(\frac{\sqrt{\lambda_j^2 - 4\lambda_j}}{2}\right)^2 \\ &= \frac{1}{4}\left(4 - 4\lambda_j + \lambda_j^2 - \lambda_j^2 + 4\lambda_j\right) \\ &= 1\end{aligned}$$

This allows us to write $\alpha_{j+} = e^{i\omega_j t}$ and $\alpha_{j-} = e^{-i\omega_j t}$. When we want to cluster, we are seeking the eigenvectors that correspond to the small eigenvalues of the normalized random walk laplacian. Notice that the smaller λ_j is, the closer the real part of α_j is to 1, so the angle ω_j is small. Let $A_j = \boldsymbol{z}(0)^T\boldsymbol{p}_j$ and $B_j = \boldsymbol{z}(0)^T\boldsymbol{q}_j$. Since we've chosen $\boldsymbol{z}(0)$ to be orthogonal to $(\boldsymbol{1}\ \boldsymbol{-1})^T$ and the eigenvectors span the rest of the space [6], we are able to express $\boldsymbol{z}(0)$ as a linear combination of eigenvectors.

$$\begin{aligned}\boldsymbol{z}(0) &= \sum_{j=1}^{n}(\boldsymbol{z}(0)^T\boldsymbol{m}_{j+})\boldsymbol{m}_{j+} + \sum_{j=1}^{n}(\boldsymbol{z}(0)^T\boldsymbol{m}_{j-})\boldsymbol{m}_{j-} \\ &= \sum_{j=1}^{n}(A_j + iB_j)(\boldsymbol{p} + i\boldsymbol{q}) + (A_j - iB_j)(\boldsymbol{p} - i\boldsymbol{q})\end{aligned}$$

So,

$$\begin{aligned}
\boldsymbol{z}(t) &= \boldsymbol{M}^t \boldsymbol{z}(0) \\
&= \boldsymbol{M}^t \left[\sum_{j=1}^{n} (A_j + iB_j)(\boldsymbol{p} + i\boldsymbol{q}) + (A_j - iB_j)(\boldsymbol{p} - i\boldsymbol{q}) \right] \\
&= \sum_{j=1}^{n} \left[(A_j + iB_j)e^{i\omega_j t}(\boldsymbol{p}_j + i\boldsymbol{q}_j) + (A_j - iB_j)e^{-i\omega_j t}(\boldsymbol{p}_j - i\boldsymbol{q}_j) \right] \\
&= \sum_{j=1}^{n} \boldsymbol{p}_j \left[(A_j + iB_j)e^{\omega_j t} + (A_j - iB_j)e^{-\omega_j t} \right] + i\boldsymbol{q}_j \left[(A_j + iB_j)e^{\omega_j t} - (A_j - iB_j)e^{-\omega_j t} \right] \\
&= 2\sum_{j=1}^{n} \boldsymbol{p}_j \left[A_j \cos(\omega_j t) - B_j \sin(\omega_j t) \right] - \boldsymbol{q}_j \left[A_j \sin(\omega_j t) + B_j \cos(\omega_j t) \right] \\
&= 2\sum_{j=1}^{n} \cos(\omega_j t) \left[A_j \boldsymbol{p}_j - B_j \boldsymbol{q_j} \right] - \sin(\omega_j t) \left[B_j \boldsymbol{p}_j + A_j \boldsymbol{q}_j \right]
\end{aligned}$$

$$\boldsymbol{z}(t) = 2\sum_{j=1}^{N} A_j \left[\boldsymbol{p}_j \cos(\omega_j t) - \boldsymbol{q}_j \boldsymbol{\sin}(\omega_j t) \right] - B_j \left[\boldsymbol{p}_j \sin(\omega_j t) + \boldsymbol{q}_j \cos(\omega_j t) \right] \tag{7.2}$$

Rewriting $\boldsymbol{z}(t)$ in terms of $\boldsymbol{w}(t)$ and $\boldsymbol{w}(t-1)$, and recognizing that $\text{Real}(\alpha_{j+}) = \cos(\omega_j t)$ and $\text{Imag}(\alpha_{j+}) = \sin(\omega_j t)$, we have

$$\boldsymbol{p}_j = \begin{pmatrix} \cos(\omega_j t)\boldsymbol{v}_j \\ \boldsymbol{v}_j \end{pmatrix} \text{ and } \boldsymbol{q}_j = \begin{pmatrix} \sin(\omega_j t)\boldsymbol{v}_j \\ \boldsymbol{0} \end{pmatrix}$$

Plugging into equation 7.2,

$$\begin{aligned}
\begin{pmatrix} \boldsymbol{w}(t) \\ \boldsymbol{w}(t-1) \end{pmatrix} = 2\sum_{j=1}^{N} A_j & \left[\begin{pmatrix} \cos(\omega_j t)\boldsymbol{v}_j \\ \boldsymbol{v}_j \end{pmatrix} \cos(\omega_j t) - \begin{pmatrix} \sin(\omega_j t)\boldsymbol{v}_j \\ \boldsymbol{0} \end{pmatrix} \sin(\omega_j t) \right] \\
-B_j & \left[\begin{pmatrix} \cos(\omega_j t)\boldsymbol{v}_j \\ \boldsymbol{v}_j \end{pmatrix} \sin(\omega_j t) + \begin{pmatrix} \sin(\omega_j t)\boldsymbol{v}_j \\ \boldsymbol{0} \end{pmatrix} \cos(\omega_j t) \right]
\end{aligned}$$

Looking at the top half of each vector, we have

$$\begin{aligned}
\boldsymbol{w}(t) &= 2\sum_{j=1}^{n} \boldsymbol{v}_j \cos(\omega_j t) \left[A_j \cos(\omega_j t) - B_j \sin(\omega_j t) \right] - \boldsymbol{v}_j \sin(\omega_j t) \left[A_j \sin(\omega_j t) + B_j \cos(\omega_j t) \right] \\
&= 2\sum_{j=1}^{n} A_j \boldsymbol{v}_j \left(\cos^2(\omega_j t) - \sin^2(\omega_j t) \right) - 2B_j \boldsymbol{v}_j (\sin(\omega_j t)\cos(\omega_j t)) \\
&= 2\sum_{j=1}^{n} A_j \boldsymbol{v}_j \cos(2\omega_j t) - B_j \boldsymbol{v}_j \sin(2\omega_j t) \qquad (7.3)
\end{aligned}$$

For node i, we have a signal that evolves with time as $[w^{(i)}(0), \ldots, w^{(i)}(T_{max})]$. The frequency peaks of FFT applied to $[w^{(i)}(0), \ldots, w^{(i)}(T_{max})]$ corresponds to the i^{th} entry of the j^{th} eigenvector. They will be of the form $(A_j + iB_j)v_i^{(j)}$. Notice that $A_j + iB_j$ remains the same for all nodes in the j^{th} eigenvector, so the sign of real component allows us to find the sign of $v_i^{(j)}$. Since graph spectral clustering can be thought of as a relaxation of indicator vectors where the i^{th} column is roughly an indicator vector for the i^{th} cluster, this algorithm gives us a method to cluster the vertices.