COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2022.

Lecture 19

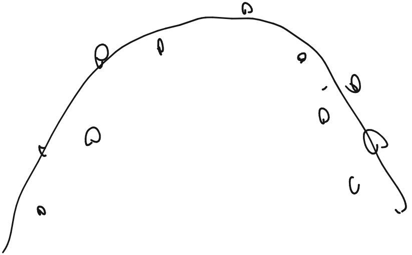
Logistics

- Problem Set 3 is due Monday at 11:59pm.
- No quiz due.

Summary

Last Class: Applications of Low-Rank Approximation

- Matrix completion
- Entity Embeddings. ⇒ high dimensional vectors → low approx.
- Non-linear dimensionality reduction via low-rank approximation of near-neighbor graphs



Summary

Last Class: Applications of Low-Rank Approximation

- Matrix completion
- Entity Embeddings.
- Non-linear dimensionality reduction via low-rank approximation of near-neighbor graphs

This Class: Spectral Graph Theory and Spectral Clustering

- Start on graph clustering for community detection and non-linear clustering.
- Spectral clustering: finding good cuts via Laplacian eigenvectors.
- Start on stochastic block model: A simple clustered graph model where we can prove the effectiveness of spectral clustering.

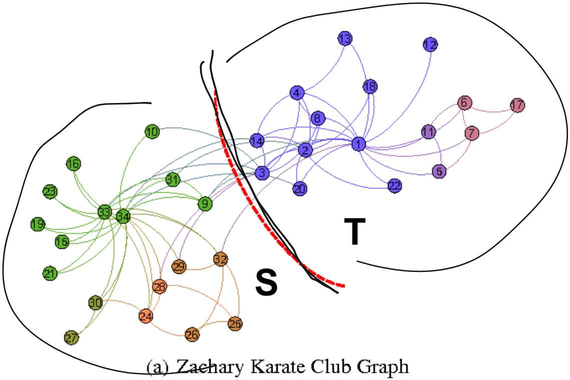
Spectral Clustering

A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

Spectral Clustering

A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

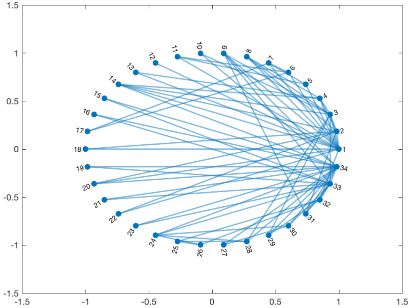
Community detection in naturally occurring networks.



(a) Zachary Karate Club Graph

A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

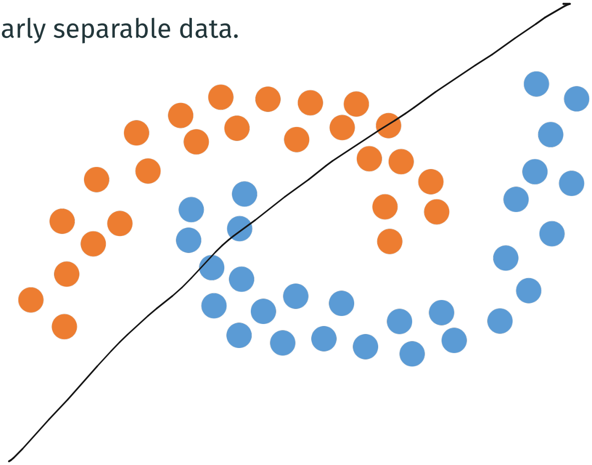
Community detection in naturally occurring networks.



Spectral Clustering

A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

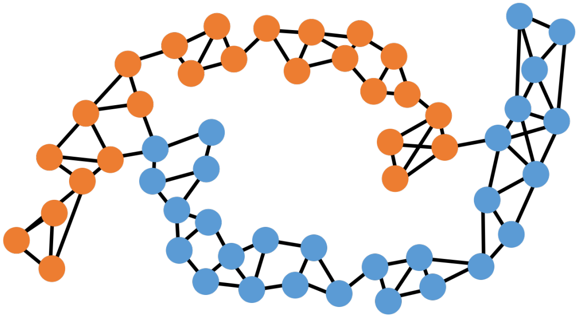
Non-linearly separable data.



Spectral Clustering

A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

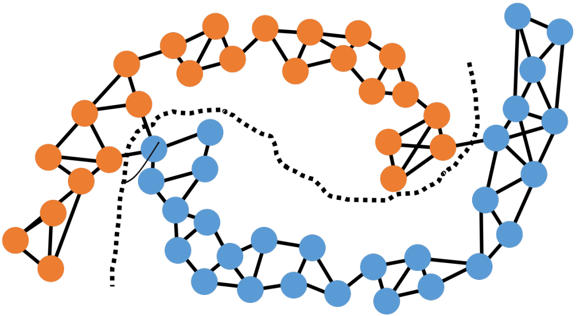
Non-linearly separable data.



Spectral Clustering

A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

Non-linearly separable data.

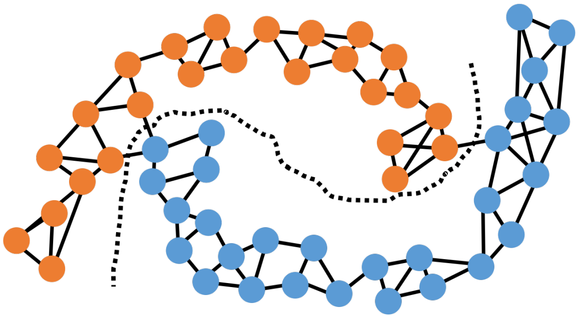


- kernel SVMs.

Spectral Clustering

A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

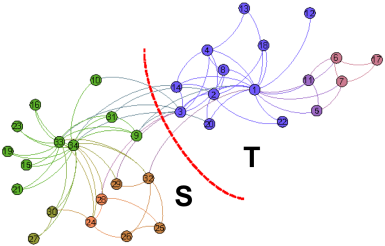
Non-linearly separable data.



Next Few Classes: Find this cut using eigendecomposition. First – motivate why this type of approach makes sense.

Cut Minimization

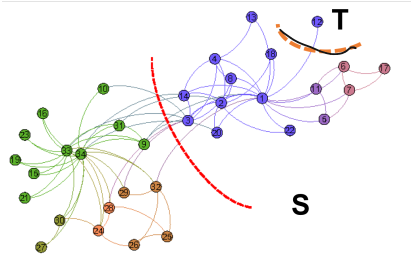
Simple Idea: Partition clusters along minimum cut in graph.



(a) Zachary Karate Club Graph

Cut Minimization

Simple Idea: Partition clusters along minimum cut in graph.



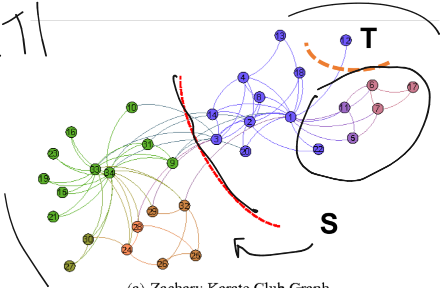
(a) Zachary Karate Club Graph

Small cuts are often not informative.

Cut Minimization

Simple Idea: Partition clusters along minimum cut in graph.

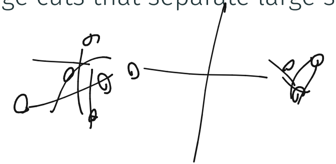
$|S| = |T|$



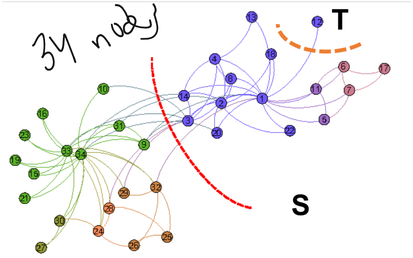
(a) Zachary Karate Club Graph

Small cuts are often not informative.

Solution: Encourage cuts that separate large sections of the graph.



Simple Idea: Partition clusters along minimum cut in graph.



(a) Zachary Karate Club Graph

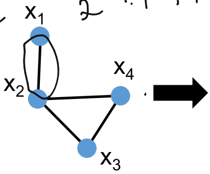
Small cuts are often not informative.

Solution: Encourage cuts that separate large sections of the graph.

- Let $\vec{v} \in \mathbb{R}^n$ be a cut indicator: $\vec{v}(i) = 1$ if $i \in S$. $\vec{v}(i) = -1$ if $i \in T$. Want $\vec{v}$ to have roughly equal numbers of 1s and -1s. I.e., $\vec{v}^T \vec{1} \approx 0$.

The Laplacian View

For a graph with adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the graph Laplacian.



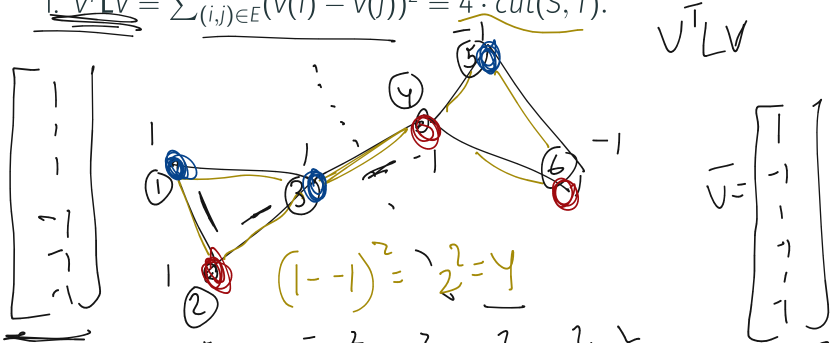
$$\overset{\mathbf{D}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}} - \overset{\mathbf{A}}{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} = \overset{\mathbf{L}}{\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 3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

For any vector $\vec{v}$, its 'smoothness' over the graph is given by:

$$\sum_{(i,j)\in E} (\vec{v}(i) - \vec{v}(j))^2 = \vec{v}^T \mathbf{L} \vec{v}.$$

For a cut indicator vector $\vec{v} \in \{-1,1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

1. $\vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j)\in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S,T)$.



$\bar{v}^T L \bar{v} = 2^2 + 2^2 + 2^2 + 2^2 + 2^2 = 20$

The Laplacian View

E edges

For a cut indicator vector $\vec{v} \in \{-1, 1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

1. $\vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S, T)$.
2. $\vec{v}^T \vec{1} = |T| - |S|$.

The Laplacian View

For a cut indicator vector $\vec{v} \in \{-1, 1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

1. $\vec{v}^T \mathbf{L} \vec{V} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S, T)$.
2. $\vec{v}^T \vec{1} = |V| - |S|$.

Want to minimize both $\vec{v}^T \mathbf{L} \vec{v}$ (cut size) and $\vec{v}^T \vec{1}$ (imbalance).

The Laplacian View

For a cut indicator vector $\vec{v} \in \{-1, 1\}^n$ with $\vec{v}(i) = -1$ for $i \in S$ and $\vec{v}(i) = 1$ for $i \in T$:

1. $\vec{v}^T \mathbf{L} \vec{V} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot cut(S, T)$.
2. $\vec{v}^T \vec{1} = |V| - |S|$.

Want to minimize both $\vec{v}^T \mathbf{L} \vec{v}$ (cut size) and $|\vec{v}^T \vec{1}|$ (imbalance).

Next Step: See how this dual minimization problem is naturally solved (sort of) by eigendecomposition.

Smallest Laplacian Eigenvector

The smallest eigenvector of the Laplacian is:

$$\vec{v}_n = \frac{1}{\sqrt{n}} \cdot \vec{1} = \underset{v \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}$$

with eigenvalue $\lambda_n(\mathbf{L}) = \vec{v}_n^T \mathbf{L} \vec{v}_n = 0$. Why?

$$Lv = \lambda v$$

$$v^T L v = \lambda v^T v = \lambda$$

$$\|v_n\|_2 = 1$$

$$v_n^T L v_n = \sum_{(i,j) \in E} [v_n(i) - v_n(j)]^2 = 0$$

n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

$$v^T L v \geq 0$$

Second Smallest Laplacian Eigenvector

By Courant-Fischer, the second smallest eigenvector is given by:

$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1,\ \vec{v}_n^T \vec{v}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}.$$

n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$. S, T: vertex sets on different sides of cut.

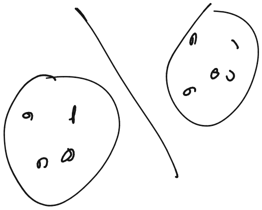
Second Smallest Laplacian Eigenvector

By Courant-Fischer, the second smallest eigenvector is given by:

$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1,\ \vec{v}_n^T \vec{v}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}.$$

If $\vec{v}_{n-1}$ were in $\left\{-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right\}^n$ it would have:

- $\vec{v}_{n-1}^T L \vec{v}_{n-1} = \frac{4}{\sqrt{n}} \cdot cut(S, T)$ as small as possible given that $\vec{v}_{n-1}^T \vec{v}_n = \frac{1}{\sqrt{n}} \vec{v}_{n-1}^T \vec{1} = \frac{|T|-|S|}{n} = 0.$



n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$. S, T: vertex sets on different sides of cut.

Second Smallest Laplacian Eigenvector

By Courant-Fischer, the second smallest eigenvector is given by:

$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1,\ \vec{v}_n^T \vec{v}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}.$$

If $\vec{v}_{n-1}$ were in $\left\{-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right\}^n$ it would have:

- $\vec{v}_{n-1}^T L \vec{v}_{n-1} = \frac{4}{\sqrt{n}} \cdot cut(S,T)$ as small as possible given that $\vec{v}_{n-1}^T \vec{v}_n = \frac{1}{\sqrt{n}} \vec{v}_{n-1}^T \vec{1} = \frac{|T|-|S|}{n} = 0$.
- I.e., $\vec{v}_{n-1}$ would indicate the smallest perfectly balanced cut.

> n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$. S, T: vertex sets on different sides of cut.

Second Smallest Laplacian Eigenvector

By Courant-Fischer, the second smallest eigenvector is given by:

$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1,\ \vec{v}_n^T \vec{v}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}.$$

$v \in \{\frac{1}{\sqrt{n}}, -\frac{1}{\sqrt{n}}\}^n$

If $\vec{v}_{n-1}$ were in $\left\{-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right\}^n$ it would have:

- $\vec{v}_{n-1}^T L \vec{v}_{n-1} = \frac{4}{\sqrt{n}} \cdot cut(S, T)$ as small as possible given that $\vec{v}_{n-1}^T \vec{v}_n = \frac{1}{\sqrt{n}} \vec{v}_{n-1}^T \vec{1} = \frac{|T|-|S|}{n} = 0.$
- I.e., $\vec{v}_{n-1}$ would indicate the smallest perfectly balanced cut.
- The eigenvector $\vec{v}_{n-1} \in \mathbb{R}^n$ is not generally binary, but still satisfies a 'relaxed' version of this property.

n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$. S, T: vertex sets on different sides of cut.

Cutting With the Second Laplacian Eigenvector

Find a good partition of the graph by computing

$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}^T \vec{1}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{V}.$$

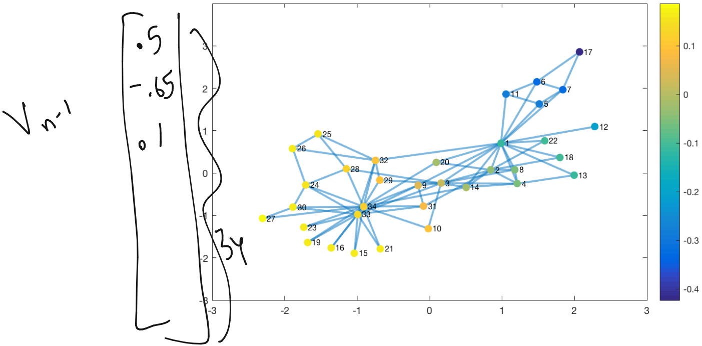
Set S to be all nodes with $\vec{v}_{n-1}(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.

Cutting With the Second Laplacian Eigenvector

Find a good partition of the graph by computing

$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}^T\vec{1}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}.$$

Set S to be all nodes with $\vec{v}_{n-1}(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.



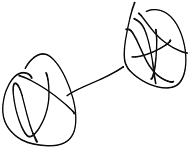
Find a good partition of the graph by computing

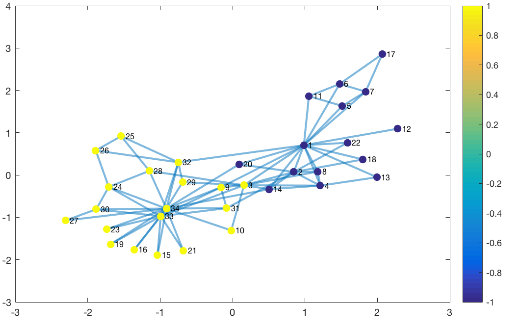
$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}^T\vec{1}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}.$$

Fiedler vector

v_{n-1}

Set S to be all nodes with $\vec{v}_{n-1}(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.





$v_{n-1}^T \mathbb{1} = 0$

Spectral Partitioning in Practice

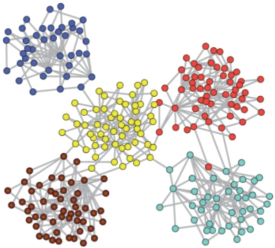
The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\bar{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

Spectral Partitioning in Practice

The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\bar{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

Important Consideration: What to do when we want to split the graph into more than two parts?



n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n\times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n\times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n\times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

Spectral Partitioning in Practice

The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\overline{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

Important Consideration: What to do when we want to split the graph into more than two parts?

Spectral Clustering:

n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

Spectral Partitioning in Practice

The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\overline{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

Important Consideration: What to do when we want to split the graph into more than two parts?

Spectral Clustering:

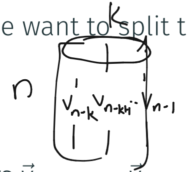
- Compute smallest k nonzero eigenvectors $\vec{v}_{n-1}, \ldots, \vec{v}_{n-k}$ of $\overline{\mathbf{L}}$.

n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

Spectral Partitioning in Practice

The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\overline{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

Important Consideration: What to do when we want to split the graph into more than two parts?



Spectral Clustering:

- Compute smallest k nonzero eigenvectors $\vec{v}_{n-1}, \ldots, \vec{v}_{n-k}$ of $\overline{\mathbf{L}}$.
- Represent each node by its corresponding row in $\mathbf{V} \in \mathbb{R}^{n \times k}$ whose columns are $\vec{v}_{n-1}, \ldots \vec{v}_{n-k}$.

> n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n \times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n \times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

Spectral Partitioning in Practice

The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\bar{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

Important Consideration: What to do when we want to split the graph into more than two parts?

$\begin{bmatrix}-1\\ -.5\\ .6\end{bmatrix}$

Spectral Clustering:

- Compute smallest k nonzero eigenvectors $\vec{v}_{n-1}, \ldots, \vec{v}_{n-k}$ of $\bar{\mathbf{L}}$.
- Represent each node by its corresponding row in $\mathbf{V} \in \mathbb{R}^{n\times k}$ whose columns are $\vec{v}_{n-1}, \ldots \vec{v}_{n-k}$.
- Cluster these rows using k-means clustering (or really any clustering method).

n: number of nodes in graph, $\mathbf{A} \in \mathbb{R}^{n\times n}$: adjacency matrix, $\mathbf{D} \in \mathbb{R}^{n\times n}$: diagonal degree matrix, $\mathbf{L} \in \mathbb{R}^{n\times n}$: Laplacian matrix $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

Laplacian Embedding

The smallest eigenvectors of $\mathbf{L} = \mathbf{D} - \mathbf{A}$ give the orthogonal 'functions' that are smoothest over the graph. I.e., minimize

$$\vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

Laplacian Embedding

The smallest eigenvectors of $\mathsf{L} = \mathsf{D} - \mathsf{A}$ give the orthogonal 'functions' that are smoothest over the graph. I.e., minimize

$$\vec{v}^T \mathsf{L} \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

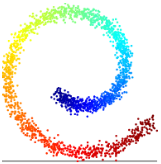
Embedding points with coordinates given by $[\vec{v}_{n-1}(j), \vec{v}_{n-2}(j), \ldots, \vec{v}_{n-k}(j)]$ ensures that coordinates connected by edges have minimum total squared Euclidean distance.

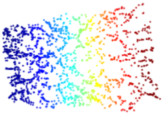
Laplacian Embedding

The smallest eigenvectors of $\mathbf{L} = \mathbf{D} - \mathbf{A}$ give the orthogonal 'functions' that are smoothest over the graph. I.e., minimize

$$\vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

Embedding points with coordinates given by $[\vec{v}_{n-1}(j), \vec{v}_{n-2}(j), \ldots, \vec{v}_{n-k}(j)]$ ensures that coordinates connected by edges have minimum total squared Euclidean distance.

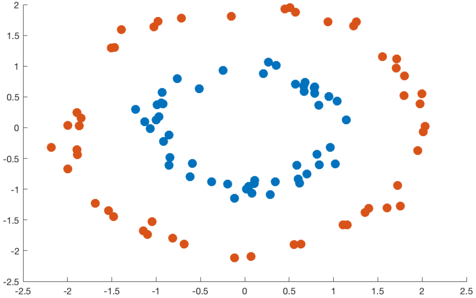




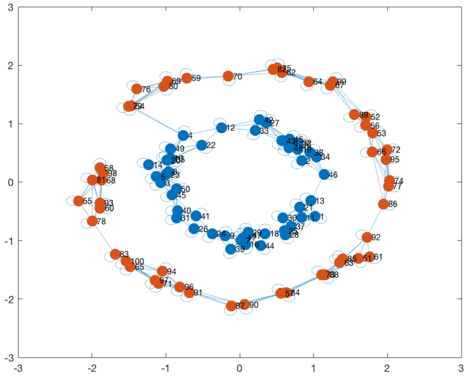
- Spectral Clustering
- Laplacian Eigenmaps
- Locally linear embedding
- Isomap
- Node2Vec, DeepWalk, etc. (variants on Laplacian)

Laplacian Embedding

Original Data: (not linearly separable)



k-Nearest Neighbors Graph:



Embedding with eigenvectors $\vec{v}_{n-1}, \vec{v}_{n-2}$: (linearly separable)