COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2021.

Lecture 19

LOGISTICS

- Week 11 Quiz will be due Monday 11/15.
- No class or office hours this Thursday due to Veteran's day.
- I will hold Office Hours in person after class on Tuesday instead. 2:30pm-3:30pm. CS 234

Last Class: Applications of Low-Rank Approximation

- Entity Embeddings.
- Non-linear dimensionality reduction via low-rank approximation of near-neighbor graphs
- Start on spectral graph theory.

SUMMARY

Last Class: Applications of Low-Rank Approximation

- Entity Embeddings.
- Non-linear dimensionality reduction via low-rank approximation of near-neighbor graphs
- Start on spectral graph theory.

This Class: Spectral Clustering and the Stochastic Block Model

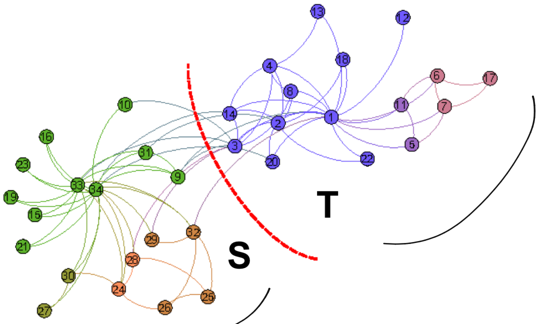
- Start on graph clustering for community detection and non-linear clustering.
- Spectral clustering: finding good cuts via Laplacian eigenvectors.
- Start on Stochastic block model: A simple clustered graph model where we can prove the effectiveness of spectral clustering.

SPECTRAL CLUSTERING

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

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

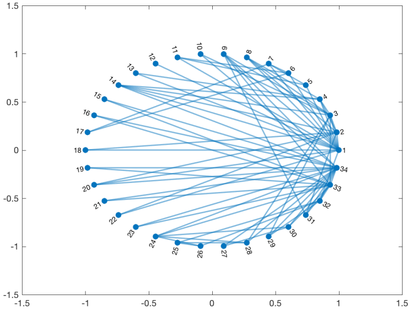
Community detection in naturally occurring networks.



(a) Zachary Karate Club Graph

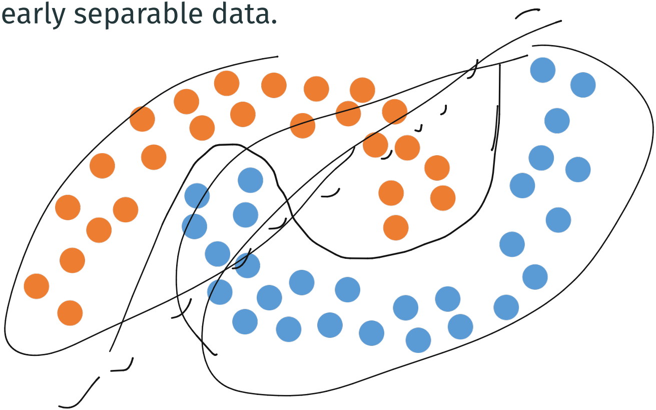
A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

Community detection in naturally occurring networks.



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

Non-linearly separable data.



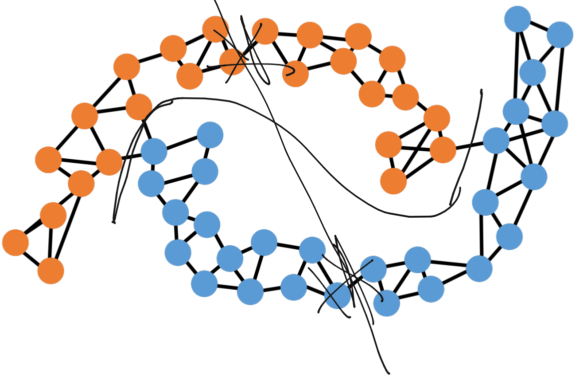
SPECTRAL CLUSTERING

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

k-means

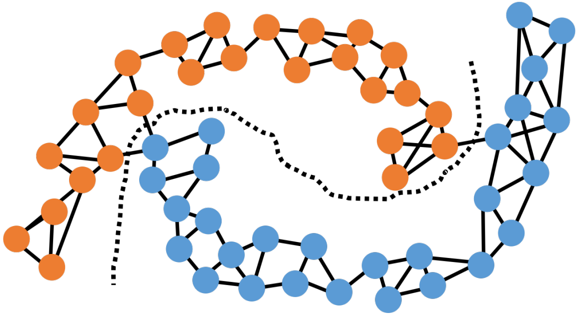
SVM

Non-linearly separable data.



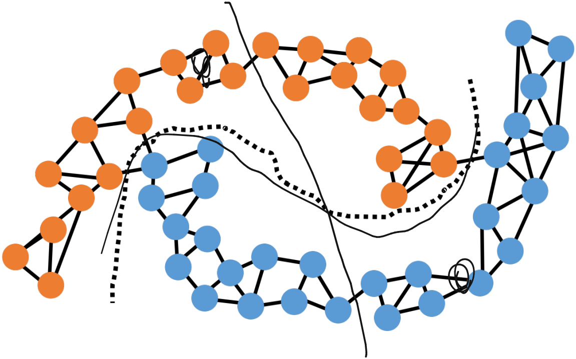
A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

Non-linearly separable data.



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

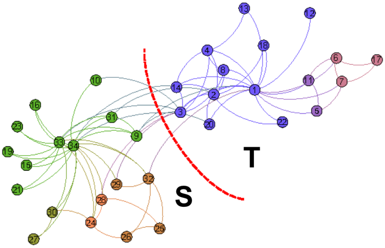
Non-linearly separable data.



Next Few Classes: Find this cut using eigendecomposition.

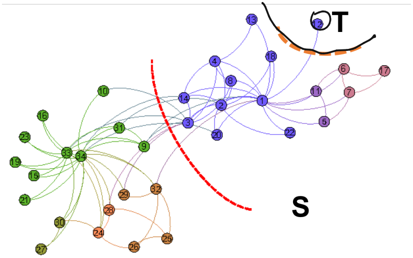
First – motivate why this type of approach makes sense.

Simple Idea: Partition clusters along minimum cut in graph.



(a) Zachary Karate Club Graph

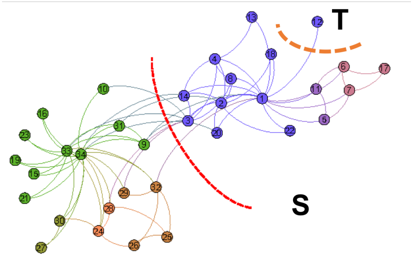
Simple Idea: Partition clusters along minimum cut in graph.



(a) Zachary Karate Club Graph

Small cuts are often not informative.

Simple Idea: Partition clusters along minimum cut in graph.



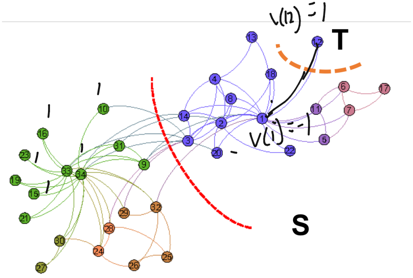
(a) Zachary Karate Club Graph

Small cuts are often not informative.

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

Simple Idea: Partition clusters along minimum cut in graph.

$v^T 1 = \sum_{i=1}^{n} v(i)$



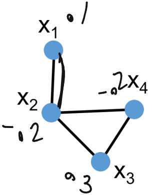
(a) Zachary Karate Club Graph

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., $\vec{v}^T\vec{1} \approx 0$.

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



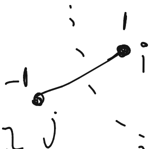
$$\overset{\mathbf{D}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}} - \overset{\mathbf{A}}{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} = \overset{\mathbf{L}}{\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 3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

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



$[v(i) - v(j)]^2 = (1 - (-1))^2 = 2^2 = 4$

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. $\vec{v}^T \vec{1} = |V| - |S|$.

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. $\vec{v}^T \vec{1} = |T| - |S|$.

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

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.

$v_n = \left[\frac{1}{\sqrt{n}}\ \frac{1}{\sqrt{n}}\ \frac{1}{\sqrt{n}}\ \cdots\ \frac{1}{\sqrt{n}}\right]$

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?

$Lv_n = 0$

$Lv = \lambda v$

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

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

$v_n^T L v_n = 0$

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

Why is 0 the smallest eigenvalue of 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 $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

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

$v_n \left[\frac{1}{\sqrt{n}} \; \frac{1}{\sqrt{n}} \; \cdots \; \frac{1}{\sqrt{n}} \right]$

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.

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:

assumption

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

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.
- 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

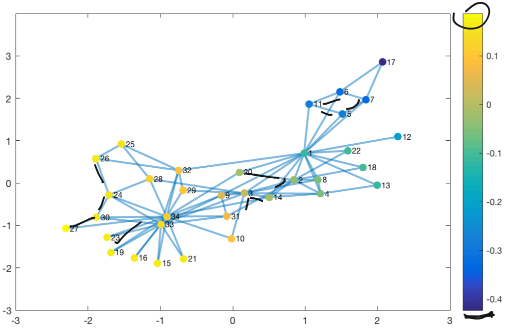
$$\vec{v}_{n-1} = \underset{v \in \mathbb{R}^d \text{with } \|\vec{v}\|=1,\ \vec{v}^T\vec{1}=0}{\arg\min} \vec{v}^T \mathbf{L} \vec{v}.$$

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

Find a good partition of the graph by computing

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

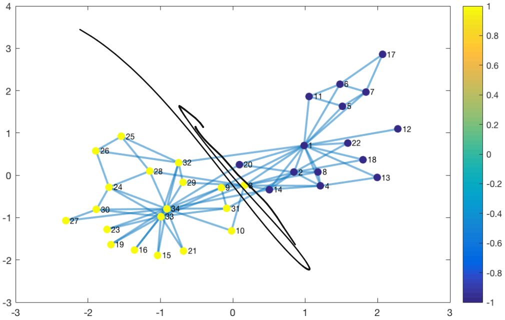
Set S to be all nodes with $\vec{v}_{n-1}(i) < 0$, T to be all with $\vec{v}_2(i) \geq 0$.



Find a good partition of the graph by computing

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

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

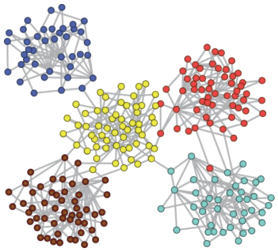


The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\bar{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

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 $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

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 $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

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

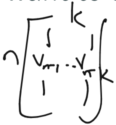
Important Consideration: What to do when we want to split the graph into more than two parts?

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 $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

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?



Spectral Clustering:

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

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 $\bar{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

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

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 rows 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 $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

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?

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 rows 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 $\mathbf{L} = \mathbf{A} - \mathbf{D}$.

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

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

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

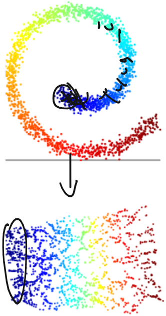
$$\vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

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.

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

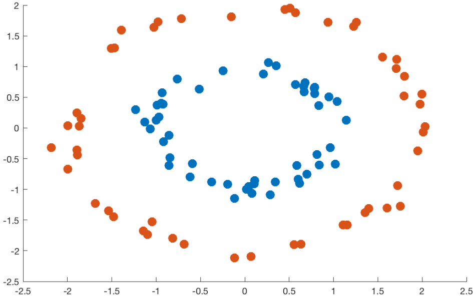
$$\vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j)\in E} [\vec{v}(i) - \vec{v}(j)]^2.$$

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.

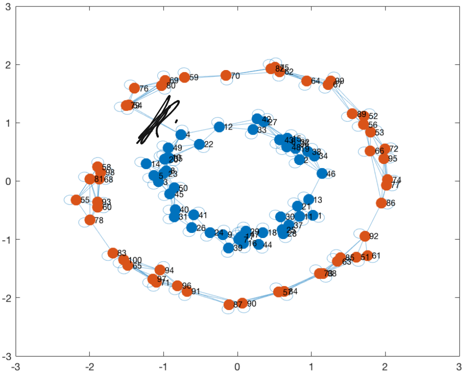


- Spectral Clustering
- Laplacian Eigenmaps
- Locally linear embedding
- Isomap
- Node2Vec, DeepWalk, etc. (variants on Laplacian)

Original Data: (not linearly separable)



k-Nearest Neighbors Graph:



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