COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco
University of Massachusetts Amherst. Fall 2019.
Lecture 15
Last Class:

• Entity embeddings (e.g., word embeddings).
• Dimensionality reduction for data not lying close to a low-dimensional subspace (non-linear dimensionality reduction).
• Approach via low-rank approximation of a graph based similarity matrix (adjacency matrix).
• Spectral graph theory, spectral clustering, graph Laplacian.

This Class: Finish up spectral clustering.

• Clustering non-linearly separable data via graph eigenvectors.
• Application to the stochastic block model and community detection.
Last Class:

- Entity embeddings (e.g., word embeddings).
- Dimensionality reduction for data not lying close to a low-dimensional subspace (non-linear dimensionality reduction).
- Approach via low-rank approximation of a graph based similarity matrix (adjacency matrix).
- Spectral graph theory, spectral clustering, graph Laplacian.

This Class: Finish up spectral clustering.

- Clustering non-linearly separable data via graph eigenvectors.
- Application to the stochastic block model and community detection.
Last Class:

- Entity embeddings (e.g., word embeddings).
- Dimensionality reduction for data not lying close to a low-dimensional subspace (non-linear dimensionality reduction).
- Approach via low-rank approximation of a graph based similarity matrix (adjacency matrix).
- Spectral graph theory, spectral clustering, graph Laplacian.

This Class: Finish up spectral clustering.

- Clustering non-linearly separable data via graph eigenvectors.
- Application to the stochastic block model and community detection.
**Goal:** Partition or cluster vertices in a graph based on ‘similarity’.
Goal: Partition or cluster vertices in a graph based on ‘similarity’.

Linearly separable data.
Goal: *Partition or cluster* vertices in a graph based on ‘similarity’.

Linearly separable data.
Goal: Partition or cluster vertices in a graph based on ‘similarity’.

Non-linearly separable data $k$-nearest neighbor graph.
Goal: *Partition or cluster* vertices in a graph based on ‘similarity’.

Non-linearly separable data $k$-nearest neighbor graph.
Goal: Partition or cluster vertices in a graph based on 'similarity'.

Non-linearly separable data $k$-nearest neighbor graph.
Goal: Partition or cluster vertices in a graph based on ‘similarity’.

Community detection in naturally occurring networks.

(a) Zachary Karate Club Graph
Main Idea: Partition clusters along a cut that:

1. Has few edges crossing it: $|(u, v) \in E : u \in S, v \in T| $ is small.
2. Separates large sections of the graph: $|S|, |T|$ are not too small.

(a) Zachary Karate Club Graph
THE LAPLACIAN VIEW

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

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. $\vec{v}^T L \vec{v} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot \text{cut}(S, T)$.
2. $\vec{v}^T \vec{1} = |V| - |S|$.
For a graph with adjacency matrix $A$ and degree matrix $D$, $L = D - A$ is the graph Laplacian.

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

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

Want to minimize both $\vec{\nu}^T L \vec{\nu}$ (cut size) and $\vec{\nu}^T \vec{1}$ (imbalance).
The smallest eigenvector of the Laplacian is:

$$\vec{v}_n = \frac{1}{\sqrt{n}} \cdot \vec{1} = \arg\min_{\vec{v} \in \mathbb{R}^n} \vec{v}^T L \vec{v}$$

with \(\vec{v}_n^T L \vec{v}_n = 0\).

\(n\): number of nodes in graph, \(A \in \mathbb{R}^{n \times n}\): adjacency matrix, \(D \in \mathbb{R}^{n \times n}\): diagonal degree matrix, \(L \in \mathbb{R}^{n \times n}\): Laplacian matrix \(L = A - D\).
The smallest eigenvector of the Laplacian is:

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

with $$\vec{v}_n^T L \vec{v}_n = 0$$. Why? Use that $$L = D - A$$.

---

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

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

with $\vec{v}_n^T L \vec{v}_n = 0$. Why? Use that $L = D - A$.

---

$n$: number of nodes in graph, $A \in \mathbb{R}^{n \times n}$: adjacency matrix, $D \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $L \in \mathbb{R}^{n \times n}$: Laplacian matrix $L = A - D$. 
By Courant-Fischer, the second smallest eigenvector is given by:

\[ \vec{v}_{n-1} = \arg\min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1} \vec{v}^T L \vec{v} \]

\[ \text{with } \vec{v}_n^T \vec{v} = 0 \]

---

**n**: number of nodes in graph, \( A \in \mathbb{R}^{n \times n} \): adjacency matrix, \( D \in \mathbb{R}^{n \times n} \): diagonal degree matrix, \( L \in \mathbb{R}^{n \times n} \): Laplacian matrix \( L = A - D \). \( S, T \): vertex sets on different sides of cut.
By Courant-Fischer, the second smallest eigenvector is given by:

$$\vec{v}_{n-1} = \arg \min_{\vec{v} \in \mathbb{R}^n} \vec{v}^T L \vec{v} \quad \text{with} \quad \vec{v} \parallel \vec{v} = 1, \vec{v}^T \vec{v} = 0$$

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

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

$n$: number of nodes in graph, $A \in \mathbb{R}^{n \times n}$: adjacency matrix, $D \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $L \in \mathbb{R}^{n \times n}$: Laplacian matrix $L = A - D$. $S, T$: vertex sets on different sides of cut.
By Courant-Fischer, the second smallest eigenvector is given by:

\[ \vec{v}_{n-1} = \arg\min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1} \vec{v}^T L \vec{v} \]

If \( \vec{v}_{n-1} \) were in \( \{-1, 1\}^n \) it would have:

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

\( n \): number of nodes in graph, \( A \in \mathbb{R}^{n \times n} \): adjacency matrix, \( D \in \mathbb{R}^{n \times n} \): diagonal degree matrix, \( L \in \mathbb{R}^{n \times n} \): Laplacian matrix \( L = A - D \). \( S, T \): vertex sets on different sides of cut.
By Courant-Fischer, the second smallest eigenvector is given by:

$$\vec{v}_{n-1} = \arg\min_{\vec{v} \in \mathbb{R}^n \text{ with } ||\vec{v}||=1} \vec{v}^T L \vec{v}$$

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

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

\[\begin{align*}
n: \text{number of nodes in graph}, \quad A &\in \mathbb{R}^{n \times n}: \text{adjacency matrix}, \quad D &\in \mathbb{R}^{n \times n}: \text{diagonal degree matrix}, \quad L &\in \mathbb{R}^{n \times n}: \text{Laplacian matrix } L = A - D. \quad S, T: \text{vertex sets on different sides of cut.}\end{align*}\]
Find a good partition of the graph by computing

$$\vec{v}_{n-1} = \arg\min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1, \vec{v}^T\vec{1}=0} \vec{v}^T L \vec{v}$$

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

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

\[ \vec{v}_{n-1} = \arg \min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\| = 1} \vec{v}^T \vec{L} \vec{v} \]

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

\( n \): number of nodes in graph, \( A \in \mathbb{R}^{n \times n} \): adjacency matrix, \( D \in \mathbb{R}^{n \times n} \): diagonal degree matrix, \( L \in \mathbb{R}^{n \times n} \): Laplacian matrix \( L = A - D \). \( S, T \): vertex sets on different sides of cut.
Find a good partition of the graph by computing

$$\vec{v}_{n-1} = \arg \min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1} \vec{v}^T L \vec{v}$$

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

$n$: number of nodes in graph, $A \in \mathbb{R}^{n \times n}$: adjacency matrix, $D \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $L \in \mathbb{R}^{n \times n}$: Laplacian matrix $L = A - D$. $S, T$: vertex sets on different sides of cut.
The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian \( \tilde{L} = D^{-1/2}LD^{-1/2} \).

\( n \): number of nodes in graph, \( A \in \mathbb{R}^{n \times n} \): adjacency matrix, \( D \in \mathbb{R}^{n \times n} \): diagonal degree matrix, \( L \in \mathbb{R}^{n \times n} \): Laplacian matrix \( L = A - D \).
The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\bar{L} = D^{-1/2}LD^{-1/2}$.

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

$n$: number of nodes in graph, $A \in \mathbb{R}^{n \times n}$: adjacency matrix, $D \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $L \in \mathbb{R}^{n \times n}$: Laplacian matrix $L = A - D$. 
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 \( \mathbf{L} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \).

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

**Spectral Clustering:**

- Compute smallest \( k \) nonzero eigenvectors \( \mathbf{v}_1; \ldots; \mathbf{v}_k \) of \( \mathbf{L} \).
- Represent each node by its corresponding row in \( \mathbf{V} \in \mathbb{R}^{n \times k} \) whose rows are \( \mathbf{v}_1; \ldots; \mathbf{v}_k \).
- 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 Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\bar{L} = D^{-1/2}LD^{-1/2}$.

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

**Spectral Clustering:**

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

---

$n$: number of nodes in graph, $A \in \mathbb{R}^{n \times n}$: adjacency matrix, $D \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $L \in \mathbb{R}^{n \times n}$: Laplacian matrix $L = A - D$. 
The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\overline{L} = D^{-1/2}LD^{-1/2}$.

**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 $\overline{L}$.
- Represent each node by its corresponding row in $V \in \mathbb{R}^{n \times k}$ whose rows are $\vec{v}_{n-1}, \ldots \vec{v}_{n-k}$.

$n$: number of nodes in graph, $A \in \mathbb{R}^{n \times n}$: adjacency matrix, $D \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $L \in \mathbb{R}^{n \times n}$: Laplacian matrix $L = A - D$. 
The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian $\overline{L} = D^{-1/2}LD^{-1/2}$.

**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 $\overline{L}$.
- Represent each node by its corresponding row in $V \in \mathbb{R}^{n \times k}$ 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, $A \in \mathbb{R}^{n \times n}$: adjacency matrix, $D \in \mathbb{R}^{n \times n}$: diagonal degree matrix, $L \in \mathbb{R}^{n \times n}$: Laplacian matrix $L = A - D$. 
The smallest eigenvectors of $L = D - A$ give the orthogonal ‘functions’ that are smoothest over the graph. I.e., minimize

$$\vec{v}^T L \vec{v} = \sum_{(i,j) \in E} [\vec{v}(i) - \vec{v}(j)]^2.$$
The smallest eigenvectors of $L = D - A$ give the orthogonal ‘functions’ that are smoothest over the graph. I.e., minimize

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

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 $L = D - A$ give the orthogonal ‘functions’ that are smoothest over the graph. I.e., minimize

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

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 $L = D - A$ give the orthogonal ‘functions’ that are smoothest over the graph. I.e., minimize

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

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
- Etc...
Original Data: (not linearly separable)
$\kappa$-Nearest Neighbors Graph:
Embedding with eigenvectors $\vec{v}_{n-1}, \vec{v}_{n-2}$: (linearly separable)
So Far: Have argued that spectral clustering partitions a graph effectively, along a small cut that separates the graph into large pieces.
**So Far:** Have argued that spectral clustering partitions a graph effectively, along a small cut that separates the graph into large pieces.

- Haven’t given any formal guarantee on the ‘quality’ of the partitioning.
So Far: Have argued that spectral clustering partitions a graph effectively, along a small cut that separates the graph into large pieces.

- Haven’t given any formal guarantee on the ‘quality’ of the partitioning.
- This is difficult to do for general input graphs.
So Far: Have argued that spectral clustering partitions a graph effectively, along a small cut that separates the graph into large pieces.

• Haven’t given any formal guarantee on the ‘quality’ of the partitioning.
• This is difficult to do for general input graphs.

Common Approach: Give a natural generative model for random inputs and analyze how the algorithm performs on inputs drawn from this model.
So Far: Have argued that spectral clustering partitions a graph effectively, along a small cut that separates the graph into large pieces.

- Haven’t given any formal guarantee on the ‘quality’ of the partitioning.
- This is difficult to do for general input graphs.

Common Approach: Give a natural generative model for random inputs and analyze how the algorithm performs on inputs drawn from this model.

- Very common in algorithm design for data analysis/machine learning (can be used to justify $\ell_2$ linear regression, $k$-means clustering, PCA, etc.)
Stochastic Block Model (Planted Partition Model): Let $G_n(p, q)$ be a distribution over graphs on $n$ nodes, split equally into two groups $B$ and $C$, each with $n/2$ nodes.
Stochastic Block Model (Planted Partition Model): Let $G_n(p, q)$ be a distribution over graphs on $n$ nodes, split equally into two groups $B$ and $C$, each with $n/2$ nodes.

- Any two nodes in the **same group** are connected with probability $p$ (including self-loops).
- Any two nodes in **different groups** are connected with prob. $q < p$.
- Connections are independent.
Stochastic Block Model (Planted Partition Model): Let $G_n(p, q)$ be a distribution over graphs on $n$ nodes, split equally into two groups $B$ and $C$, each with $n/2$ nodes.

- Any two nodes in the same group are connected with probability $p$ (including self-loops).
- Any two nodes in different groups are connected with prob. $q < p$.
- Connections are independent.
Let $G$ be a stochastic block model graph drawn from $G_n(p, q)$. 

$G_n(p, q)$: stochastic block model distribution. $B, C$: groups with $n/2$ nodes each. Connections are independent with probability $p$ between nodes in the same group, and probability $q$ between nodes not in the same group.
Let $G$ be a stochastic block model graph drawn from $G_n(p, q)$.

- Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix of $G$.
Let $G$ be a stochastic block model graph drawn from $G_n(p, q)$.

- Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix of $G$.
Let $G$ be a stochastic block model graph drawn from $G_n(p, q)$.

- Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix of $G$. What is $\mathbb{E}[A]$?

$G_n(p, q)$: stochastic block model distribution. $B, C$: groups with $n/2$ nodes each. Connections are independent with probability $p$ between nodes in the same group, and probability $q$ between nodes not in the same group.
Letting $G$ be a stochastic block model graph drawn from $G_n(p, q)$ and $A \in \mathbb{R}^{n \times n}$ be its adjacency matrix. What is $\mathbb{E}[A]$?

$G_n(p, q)$: stochastic block model distribution. $B, C$: groups with $n/2$ nodes each. Connections are independent with probability $p$ between nodes in the same group, and probability $q$ between nodes not in the same group.
Letting \( G \) be a stochastic block model graph drawn from \( G_n(p, q) \) and \( A \in \mathbb{R}^{n \times n} \) be its adjacency matrix. \( (E[A])_{i,j} = p \) for \( i, j \) in same group, \( (E[A])_{i,j} = q \) otherwise.

\( G_n(p, q) \): stochastic block model distribution. \( B, C \): groups with \( n/2 \) nodes each. Connections are independent with probability \( p \) between nodes in the same group, and probability \( q \) between nodes not in the same group.
Letting $G$ be a stochastic block model graph drawn from $G_n(p, q)$ and $A \in \mathbb{R}^{n \times n}$ be its adjacency matrix. $(\mathbb{E}[A])_{i,j} = p$ for $i, j$ in same group, $(\mathbb{E}[A])_{i,j} = q$ otherwise.

What are the eigenvectors and eigenvalues of $\mathbb{E}[A]$?

$G_n(p, q)$: stochastic block model distribution. $B, C$: groups with $n/2$ nodes each. Connections are independent with probability $p$ between nodes in the same group, and probability $q$ between nodes not in the same group.
Letting $G$ be a stochastic block model graph drawn from $G_n(p, q)$ and $A \in \mathbb{R}^{n \times n}$ be its adjacency matrix, what are the eigenvectors and eigenvalues of $\mathbb{E}[A]$?
Letting $G$ be a stochastic block model graph drawn from $G_n(p, q)$ and $A \in \mathbb{R}^{n \times n}$ be its adjacency matrix, what are the eigenvectors and eigenvalues of $\mathbb{E}[A]$?
If we compute $\vec{v}_2$ then we recover the communities $B$ and $C$!
If we compute $\vec{v}_2$ then we recover the communities $B$ and $C$!

- Can show that for $G \sim G_n(p, q)$, $A$ is close to $E[A]$ with high probability.
- Thus, the true second eigenvector of $A$ is close to $[1, 1, 1, \ldots, -1, -1, -1]$ and gives a good estimate of the communities.
Letting $G$ be a stochastic block model graph drawn from $G_n(p, q)$, $A \in \mathbb{R}^{n \times n}$ be its adjacency matrix and $L$ be its Laplacian, what are the eigenvectors and eigenvalues of $\mathbb{E}[L]$?
Letting $G$ be a stochastic block model graph drawn from $G_n(p, q)$, $A \in \mathbb{R}^{n \times n}$ be its adjacency matrix and $L$ be its Laplacian, what are the eigenvectors and eigenvalues of $\mathbb{E}[L]$?
Questions?