Last Class: Spectral Clustering

- Spectral clustering: finding good cuts via Laplacian eigenvectors.

This Class: Stochastic Block Model

- Stochastic block model: A simple clustered graph model where we can prove the effectiveness of spectral clustering.
- Prove that clustering with the Laplacian eigenvectors (spectral clustering) finds communities in the stochastic block model.
SUMMARY

Last Class: Spectral Clustering

• Spectral clustering: finding good cuts via Laplacian eigenvectors.

This Class: Stochastic Block Model

• Stochastic block model: A simple clustered graph model where we can prove the effectiveness of spectral clustering.
• Prove that clustering with the Laplacian eigenvectors (spectral clustering) finds communities in the stochastic block model.
For a graph with adjacency matrix $A$ and degree matrix $D$, $L = D - A$ is the graph Laplacian.

How smooth any vector $\vec{v}$ is over the graph can be measured by:

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

- We'll use eigenvectors of Laplacian to divide the nodes of the graph into roughly equal groups such that the number of cut edges is small.
Find a good partition of the graph by computing

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

Let $S$ be nodes with $\vec{v}_{n-1}(i) < 0$, $T$ be nodes with $\vec{v}_{n-1}(i) \geq 0$. 
Find a good partition of the graph by computing

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

Let $S$ be nodes with $\vec{v}_{n-1}(i) < 0$, $T$ be nodes with $\vec{v}_{n-1}(i) \geq 0$. 
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. $\vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot \text{cut}(S, T)$. 
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. \( \vec{v}^T \mathbf{L} \vec{v} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot \text{cut}(S, T) \).
2. \( \vec{v}^T \mathbf{1} = |T| - |S| \).
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. $\vec{v}^T L \vec{v} = \sum_{(i,j) \in E} (\vec{v}(i) - \vec{v}(j))^2 = 4 \cdot \text{cut}(S, T)$.
2. $\vec{v}^T \vec{1} = |T| - |S|$.

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

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

**Next Step:** See how this dual minimization problem is naturally solved by eigendecomposition.
The smallest eigenvector of the Laplacian is:

\[ \vec{v}_1 = \frac{1}{\sqrt{n}} \cdot \vec{1} = \arg \min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1} \vec{v}^T L \vec{v} \]

with eigenvalue \( \vec{v}_1^T L \vec{v}_1 = 0 \).

---

\( n \): 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:

$$\vec{v}_1 = \frac{1}{\sqrt{n}} \cdot \vec{1} = \arg \min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\| = 1} \vec{v}^T L \vec{v}$$

with eigenvalue $\vec{v}_1^T L \vec{v}_1 = 0$. Why?

$n$: 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:

$$\vec{v}_2 = \arg \min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1} \vec{v}^T L \vec{v}$$

If $\vec{v}_2$ were in $\{-\sqrt{n}, 1\}$ it would have:

- $\vec{v}_2^T L \vec{v}_2 = \frac{4}{n} \cdot \text{cut}(S, T)$ as small as possible subject to $\vec{v}_2^T \vec{v}_1 = 1$, $\vec{v}_1^T \vec{v} = 0$

- I.e., $\vec{v}_2$ would indicate the smallest perfectly balanced cut.

- The eigenvector $\vec{v}_2 \in \mathbb{R}^n$ is not generally binary, but still satisfies a 'relaxed' version of this property.
By Courant-Fischer, the second smallest eigenvector is given by:

$$\vec{v}_2 = \arg\min_{\vec{v} \in \mathbb{R}^n \text{ with } ||\vec{v}||=1, \vec{v}_1^T \vec{v}=0} \vec{v}^T L \vec{v}$$

If $\vec{v}_2$ were in $\left\{ -\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right\}^n$ it would have:

- $\vec{v}_2^T L \vec{v}_2 = \frac{4}{n} \cdot \text{cut}(S, T)$ as small as possible subject to

$$\vec{v}_2^T \vec{v}_1 = \frac{1}{\sqrt{n}} \vec{v}_2^T \vec{1} = \frac{|T| - |S|}{\sqrt{n}} = 0$$
By Courant-Fischer, the second smallest eigenvector is given by:

\[ \vec{v}_2 = \arg \min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\|=1, \, \vec{v}_1^T \vec{v}=0} \vec{v}^T L \vec{v} \]

If \( \vec{v}_2 \) were in \( \left\{-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right\}^n \) it would have:

- \( \vec{v}_2^T L \vec{v}_2 = \frac{4}{n} \cdot \text{cut}(S, T) \) as small as possible subject to

\[ \vec{v}_2^T \vec{v}_1 = \frac{1}{\sqrt{n}} \vec{v}_2^T \vec{1} = \frac{|T| - |S|}{\sqrt{n}} = 0 \]

- I.e., \( \vec{v}_2 \) would indicate the smallest perfectly balanced cut.
By Courant-Fischer, the second smallest eigenvector is given by:

\[ \vec{v}_2 = \arg \min_{\vec{v} \in \mathbb{R}^n \text{ with } \|\vec{v}\| = 1, \vec{v}_1^T \vec{v} = 0} \vec{v}^T L \vec{v} \]

If \( \vec{v}_2 \) were in \( \left\{ -\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right\}^n \) it would have:

- \( \vec{v}_2^T L \vec{v}_2 = \frac{4}{n} \cdot \text{cut}(S, T) \) as small as possible subject to
  \[ \vec{v}_2^T \vec{v}_1 = \frac{1}{\sqrt{n}} \vec{v}_2^T \vec{1} = \frac{|T| - |S|}{\sqrt{n}} = 0 \]

- I.e., \( \vec{v}_2 \) would indicate the smallest perfectly balanced cut.
- The eigenvector \( \vec{v}_2 \in \mathbb{R}^n \) is not generally binary, but still satisfies a ‘relaxed’ version of this property.
Find a good partition of the graph by computing

$$\vec{v}_2 = \arg\min_{\vec{v} \in \mathbb{R}^d \text{ with } \|\vec{v}\|=1, \vec{v}^T \vec{1} = 0} \vec{v}^T L \vec{v}$$

Set $S$ to be all nodes with $\vec{v}_2(i) < 0$, $T$ to be all with $\vec{v}_2(i) \geq 0$. 
Find a good partition of the graph by computing

$$\vec{v}_2 = \arg \min_{\vec{v} \in \mathbb{R}^d \text{ with } \|\vec{v}\|=1} \vec{v}^T L \vec{v}$$

Set $S$ to be all nodes with $\vec{v}_2(i) < 0$, $T$ to be all with $\vec{v}_2(i) \geq 0$. 
Find a good partition of the graph by computing

\[ \vec{v}_2 = \arg \min_{\vec{v} \in \mathbb{R}^d} \vec{v}^T L \vec{v} \]

with \( \|\vec{v}\|=1 \), \( \vec{v}_2^T \vec{1}=0 \)

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

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

Non-linearly separable data $k$-nearest neighbor graph.
A very common task is to **partition or cluster** vertices in a graph based on similarity/connectivity.

**Non-linearly separable data** \( k \)-nearest neighbor graph.
A very common task is to **partition or cluster** vertices in a graph based on similarity/connectivity.

**Non-linearly separable data** $k$-nearest neighbor graph.
A very common task is to partition or cluster vertices in a graph based on similarity/connectivity.

Non-linearly separable data \(k\)-nearest neighbor graph.

Can find this cut using eigendecomposition!
The Shi-Malik normalized cuts algorithm is one of the most commonly used variants of this approach, using the normalized Laplacian \( \overline{L} = D^{-1/2}LD^{-1/2} \).

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

$n$: 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 $t$ nonzero eigenvectors $\vec{v}_2, \ldots, \vec{v}_{t+1}$ of $L$.
- Represent each node by its corresponding row in $V \in \mathbb{R}^{n \times t}$ whose columns are $\vec{v}_2, \ldots, \vec{v}_{t+1}$.
- Cluster these rows using $k$-means clustering (or really any clustering method).
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 $t$ nonzero eigenvectors $\vec{v}_2, \ldots, \vec{v}_{t+1}$ 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 $\tilde{L} = D^{-1/2}LD^{-1/2}$.

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

**Spectral Clustering:**

- Compute smallest $t$ nonzero eigenvectors $\vec{v}_2, \ldots, \vec{v}_{t+1}$ of $\tilde{L}$.
- Represent each node by its corresponding row in $V \in \mathbb{R}^{n \times t}$ whose columns are $\vec{v}_2, \ldots, \vec{v}_{t+1}$.

$n$: 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 \( \bar{L} = D^{-1/2}LD^{-1/2} \).

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

**Spectral Clustering:**

- Compute smallest \( t \) nonzero eigenvectors \( \vec{v}_2, \ldots, \vec{v}_{t+1} \) of \( \bar{L} \).
- Represent each node by its corresponding row in \( V \in \mathbb{R}^{n \times t} \) whose columns are \( \vec{v}_2, \ldots, \vec{v}_{t+1} \).
- 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 \( L = A - D \).
So Far: Argued that spectral clustering partitions a graph, along a small cut that separates the graph into large pieces.
So Far: Argued that spectral clustering partitions a graph, along a small cut that separates the graph into large pieces.

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

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

- Haven’t given formal guarantees on ‘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: Argued that spectral clustering partitions a graph, along a small cut that separates the graph into large pieces.

• Haven’t given formal guarantees on ‘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 randomly 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 $G_n(p, q)$.

- Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix of $G$, ordered in terms of group ID.

$G_n(p, q)$: 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.

*Expected adjacency spectrum*

$$G_n(p, q):$$ 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 is $\text{rank}(\mathbb{E}[A])$?

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.
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 $E[A]$ in some appropriate sense (matrix concentration inequality).
- Second eigenvector of $A$ is close to $[1, 1, 1, \ldots, -1, -1, -1]$ and gives a good estimate of the communities.

When the rows/columns aren’t sorted by community ID, the second eigenvector is something like $[1, -1, 1, -1, \ldots, 1, 1, -1]$ and the entries give community IDs.