COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco
University of Massachusetts Amherst. Fall 2019.
Lecture 16
Last Class:

- Spectral clustering and embeddings
- Started application to stochastic block model.
Last Class:

- Spectral clustering and embeddings
- Started application to stochastic block model.

This Class:

- Finish up stochastic block model.
- Efficient algorithms for SVD/eigendecomposition.
- Iterative methods: power method, Krylov subspace methods.
**Goal:** Argue the effectiveness of spectral clustering in a natural, if oversimplified, generative model.
**Goal:** Argue the effectiveness of spectral clustering in a natural, if oversimplified, generative model.

**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.
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.

$G_n(p, q)$: 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 the same group, $(\mathbb{E}[A])_{i,j} = q$ otherwise.

What is the rank of $\mathbb{E}[A]$ and how can you see this quickly?

$G_n(p, q)$: 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 the rank of $\mathbb{E}[A]$ and how can you see this quickly? How many nonzero eigenvalues does $\mathbb{E}[A]$ have?

$G_n(p, q)$: 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.
EXPECTED ADJACENCY SPECTRUM

\[
\begin{align*}
B \quad & \quad (n/2 \text{ nodes}) \\
\quad & \quad \quad \quad E[A] \quad \quad \quad (n/2 \text{ nodes}) \\
p \quad & \quad q \\
q \quad & \quad p
\end{align*}
\]

\[
\begin{align*}
V &= \begin{pmatrix}
1 & 1 \\
1 & 1 \\
1 & 1 \\
1 & -1 \\
1 & -1 \\
1 & -1 \\
1 & -1
\end{pmatrix}, \\
\Lambda &= \begin{pmatrix}
\frac{n(p + q)}{2} & \frac{n(p - q)}{2} \\
\frac{n(p - q)}{2} & \frac{n(p + q)}{2}
\end{pmatrix}, \\
V^T &= \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
\]
\[ \vec{v}_1 = \vec{1} \text{ with eigenvalue } \lambda_1 = \frac{(p+q)n}{2}. \]

\[ \vec{v}_2 = \chi_{B,C} \text{ with eigenvalue } \lambda_2 = \frac{(p-q)n}{2}. \]

\[ \chi_{B,C}(i) = 1 \text{ if } i \in B \text{ and } \chi_{B,C}(i) = -1 \text{ for } i \in C. \]
\[ \vec{v}_1 = \vec{1} \text{ with eigenvalue } \lambda_1 = \frac{(p+q)n}{2}. \]

\[ \vec{v}_2 = \chi_{B,C} \text{ with eigenvalue } \lambda_2 = \frac{(p-q)n}{2}. \]

\[ \chi_{B,C}(i) = 1 \text{ if } i \in B \text{ and } \chi_{B,C}(i) = -1 \text{ for } i \in C. \]

If we compute \( \vec{v}_2 \) then we recover the communities \( B \) and \( C \)!
EXPECTED LAPLACIAN SPECTRUM

Letting $G$ be a stochastic block model graph drawn from $G_n(p, q)$, $\mathbf{A} \in \mathbb{R}^{n \times n}$ be its adjacency matrix and $\mathbf{L}$ be its Laplacian, what are the eigenvectors and eigenvalues of $\mathbb{E}[\mathbf{L}]$?

$$\mathbb{E}[\mathbf{L}] = \mathbb{E}[\mathbf{D} - \mathbf{A}] = \mathbb{E}\mathbf{D} - \mathbb{E}\mathbf{A}$$

$$= \left[ \begin{array}{ccc}
\frac{(p+q)n}{2} & \cdots & \frac{pq}{n} \\
\cdots & \ddots & \cdots \\
\frac{pq}{n} & \cdots & \frac{q+q}{2}
\end{array} \right] - \left[ \begin{array}{c}
p \\
q \\
p \\
q
\end{array} \right]$$

$$\mathbb{E}[\mathbf{L}] = (p+q)\mathbf{I} - \mathbb{E}[\mathbf{A}]$$

$$\mathbb{E}[\mathbf{L}] = \left( \frac{p+q}{2} \right) \mathbf{I} - \mathbb{E}[\mathbf{A}] \mathbf{V}$$

$$\mathbb{E}[\mathbf{L}] = (p+q)\mathbf{V} - \mathbb{E}[\mathbf{A}] \mathbf{V}$$

$$\mathbb{E}[\mathbf{L}] \mathbf{v}_i = \left( \frac{p+q}{2} \mathbf{v}_i - \mathbb{E}[\mathbf{A}] \mathbf{v}_i \right)$$
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 $E[L]$?

\[ E[LL^T]v_i = \frac{(p+q)\gamma}{2} v_i - \lambda_i v_i = \left[ \frac{(p+q)\gamma}{2} - \lambda_i \right] v_i \]

\[ \lambda_1 = \frac{(p+q)\gamma}{2} \quad \lambda_2 = \frac{(p-q)\gamma}{2} \quad \lambda_3, \lambda_4, \ldots = 0 \]

\[ \lambda_1 = \frac{(p+q)\gamma}{2} \quad \lambda_n = \frac{(p-q)\gamma}{2} \quad \lambda_{n-1}(E[L]) = (p+q)\gamma \]

\[ \text{rank}(E[L]) = n-1 \]

\[ \text{rank}(E[AA^T]) = 2 \]

\[ X_{BC} = \begin{bmatrix} 1 & 1 & -1 & -1 & -1 \end{bmatrix} \]
**Upshot:** The second small eigenvector of $\mathbb{E}[L]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.
**Upshot:** The second small eigenvector of $\mathbb{E}[L]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph $G$ (equivilantly $A$ and $L$) were exactly equal to its expectation, partitioning using this eigenvector would exactly recover the two communities $B$ and $C$. 
**Upshot:** The second small eigenvector of $\mathbb{E}[L]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph $G$ (equivalently $A$ and $L$) were exactly equal to its expectation, partitioning using this eigenvector would exactly recover the two communities $B$ and $C$.

How do we show that a matrix (e.g., $A$) is close to its expectation? Matrix concentration inequalities.
**Upshot:** The second small eigenvector of $\mathbb{E}[L]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph $G$ (equivalently $A$ and $L$) were exactly equal to its expectation, partitioning using this eigenvector would exactly recover the two communities $B$ and $C$.

How do we show that a matrix (e.g., $A$) is close to its expectation? Matrix concentration inequalities.

- Analogous to scalar concentration inequalities like Markovs, Chebyshevs, Bernsteins.
**Upshot:** The second small eigenvector of $\mathbb{E}[L]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph $G$ (equivilantly $A$ and $L$) were exactly equal to its expectation, partitioning using this eigenvector would exactly recover the two communities $B$ and $C$.

**How do we show that a matrix (e.g., $A$) is close to its expectation?** Matrix concentration inequalities.

- Analogous to scalar concentration inequalities like Markovs, Chebyshevs, Bernsteins.
- Random matrix theory is a very recent and cutting edge subfield of mathematics that is being actively applied in computer science, statistics, and ML.
Matrix Concentration Inequality: If $p \geq O\left(\frac{\log^4 n}{n}\right)$, then with high probability

$$\|A - \mathbb{E}[A]\|_2 \leq O(\sqrt{p n}).$$

where $\| \cdot \|_2$ is the matrix spectral norm (operator norm).

For any $X \in \mathbb{R}^{n \times d}$, $\|X\|_2 = \max_{z \in \mathbb{R}^d : \|z\|_2 = 1} \|Xz\|_2$. The top eigenvale of $X^TX$. 
Matrix Concentration Inequality: If \( p \geq O\left(\frac{\log^4 n}{n}\right) \), then with high probability

\[
\|A - \mathbb{E}[A]\|_2 \leq O(\sqrt{pn}).
\]

where \( \| \cdot \|_2 \) is the matrix spectral norm (operator norm).

For any \( X \in \mathbb{R}^{n \times d} \), \( \|X\|_2 = \max_{z \in \mathbb{R}^d : \|z\|_2 = 1} \|Xz\|_2 \).

Exercise: Show that \( \|X\|_2 \) is equal to the largest singular value of \( X \).

For symmetric \( X \) (like \( A - \mathbb{E}[A] \)) show that it is equal to the magnitude of the largest magnitude eigenvalue.
Matrix Concentration Inequality: If \( p \geq O\left(\frac{\log^4 n}{n}\right) \), then with high probability

\[
\|A - \mathbb{E}[A]\|_2 \leq O(\sqrt{pn}).
\]

where \( \| \cdot \|_2 \) is the matrix spectral norm (operator norm).

For any \( X \in \mathbb{R}^{n \times d} \), \( \|X\|_2 = \max_{z \in \mathbb{R}^d : \|z\|_2 = 1} \|Xz\|_2 \).

Exercise: Show that \( \|X\|_2 \) is equal to the largest singular value of \( X \).

For symmetric \( X \) (like \( A - \mathbb{E}[A] \)) show that it is equal to the magnitude of the largest magnitude eigenvalue.

For the stochastic block model application, we want to show that the second eigenvectors of \( A \) and \( \mathbb{E}[A] \) are close. How does this relate to their difference in spectral norm?
Davis-Kahan Eigenvector Perturbation Theorem: Suppose $A, \overline{A} \in \mathbb{R}^{d \times d}$ are symmetric with $\|A - \overline{A}\|_2 \leq \epsilon$ and eigenvectors $v_1, v_2, \ldots, v_d$ and $\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_d$. Letting $\theta(v_i, \overline{v}_i)$ denote the angle between $v_i$ and $\overline{v}_i$, for all $i$: \[
abla\theta(v_i, \overline{v}_i) \leq \frac{\epsilon}{\min_{j \neq i} |\lambda_i - \lambda_j|} \quad 1, 1+\xi\]

where $\lambda_1, \ldots, \lambda_d$ are the eigenvalues of $\overline{A}$.

The errors get large if there are eigenvalues with similar magnitudes.
For the matrix $A$, we have

$$A = V \Lambda V^T$$

with eigenvalues

$$\lambda_1(A) = 1 + \varepsilon$$
$$\lambda_2(A) = 1$$

and eigenvectors

$$v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

For the perturbed matrix $\tilde{A}$, we have

$$\lambda_1(\tilde{A}) = 1 + \varepsilon$$
$$\lambda_2(\tilde{A}) = 1$$

and eigenvectors

$$\tilde{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \tilde{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The difference $A - \tilde{A}$ is given by

$$A - \tilde{A} = \begin{bmatrix} \varepsilon & 0 \\ 0 & -\varepsilon \end{bmatrix}$$

The angle between $v_1$ and $\tilde{v}_1$ is

$$\sin \theta(v_1, \tilde{v}_1) = \frac{\varepsilon}{\varepsilon} = 1$$

The norm of the difference is

$$||A - \tilde{A}||_2 = \varepsilon$$
Claim 1 (Matrix Concentration): For $p \geq O\left(\frac{\log^4 n}{n}\right)$,

$$\|A - \mathbb{E}[A]\|_2 \leq O(\sqrt{pn}).$$

Claim 2 (Davis-Kahan): For $p \geq O\left(\frac{\log^4 n}{n}\right)$,

$$\sin \theta(v_2, \bar{v}_2) \leq \frac{O(\sqrt{pn})}{\min_{j \neq i} |\lambda_i - \lambda_j|}$$

A adjacency matrix of random stochastic block model graph. $p$: connection probability within clusters. $q < p$: connection probability between clusters. $n$: number of nodes. $v_2, \bar{v}_2$: second eigenvectors of $A$ and $\mathbb{E}[A]$ respectively.
Claim 1 (Matrix Concentration): For $p \geq O\left(\frac{\log^4 n}{n}\right)$,
\[ \|A - \mathbb{E}[A]\|_2 \leq O(\sqrt{pn}). \]

Claim 2 (Davis-Kahan): For $p \geq O\left(\frac{\log^4 n}{n}\right)$,
\[ \sin \theta(v_2, \bar{v}_2) \leq \frac{O(\sqrt{pn})}{\min_{j \neq i} |\lambda_i - \lambda_j|} \]

Recall: $\mathbb{E}[A]$, has eigenvalues $\lambda_1 = \frac{(p+q)n}{2}$, $\lambda_2 = \frac{(p-q)n}{2}$, $\lambda_i = 0$ for $i \geq 3$. 

A adjacency matrix of random stochastic block model graph. $p$: connection probability within clusters. $q < p$: connection probability between clusters. $n$: number of nodes. $v_2, \bar{v}_2$: second eigenvectors of $A$ and $\mathbb{E}[A]$ respectively.
Claim 1 (Matrix Concentration): For \( p \geq O \left( \frac{\log^4 n}{n} \right) \),
\[
\|A - \mathbb{E}[A]\|_2 \leq O(\sqrt{pn}).
\]

Claim 2 (Davis-Kahan): For \( p \geq O \left( \frac{\log^4 n}{n} \right) \),
\[
\sin \theta(v_2, \bar{v}_2) \leq \frac{O(\sqrt{pn})}{\min_{j \neq i} |\lambda_i - \lambda_j|}
\]

Recall: \( \mathbb{E}[A] \), has eigenvalues \( \lambda_1 = \frac{(p+q)n}{2} \), \( \lambda_2 = \frac{(p-q)n}{2} \), \( \lambda_i = 0 \) for \( i \geq 3 \).
\[
\min_{j \neq i} |\lambda_i - \lambda_j| = \min \left( qn, \frac{(p-q)n}{2} \right).
\]
APPLICATION TO STOCHASTIC BLOCK MODEL

Claim 1 (Matrix Concentration): For \( p \geq O \left( \frac{\log^4 n}{n} \right) \),
\[
\| A - \mathbb{E}[A] \|_2 \leq O(\sqrt{pn}).
\]

Claim 2 (Davis-Kahan): For \( p \geq O \left( \frac{\log^4 n}{n} \right) \),
\[
\sin \theta(v_2, \bar{v}_2) \leq \frac{O(\sqrt{pn})}{\min_{j \neq i} |\lambda_i - \lambda_j|}
\]

Recall: \( \mathbb{E}[A] \), has eigenvalues \( \lambda_1 = \frac{(p+q)n}{2}, \lambda_2 = \frac{(p-q)n}{2}, \lambda_i = 0 \) for \( i \geq 3 \).
\[
\min_{j \neq i} |\lambda_i - \lambda_j| = \min \left( qn, \frac{(p-q)n}{2} \right)
\]

Typically, \( \frac{(p-q)n}{2} \) will be the minimum of these two gaps.
APPLICATION TO STOCHASTIC BLOCK MODEL

Claim 1 (Matrix Concentration): For \( p \geq O \left( \frac{\log^4 n}{n} \right) \),

\[ \|A - \mathbb{E}[A]\|_2 \leq O(\sqrt{pn}). \]

Claim 2 (Davis-Kahan): For \( p \geq O \left( \frac{\log^4 n}{n} \right) \),

\[ \sin \theta(v_2, \bar{v}_2) \leq \frac{O(\sqrt{pn})}{\min_{j \neq i} |\lambda_i - \lambda_j|} \leq \frac{O(\sqrt{pn})}{(p - q)n/2} = O \left( \frac{\sqrt{p}}{(p - q)\sqrt{n}} \right) \]

Recall: \( \mathbb{E}[A] \), has eigenvalues \( \lambda_1 = \frac{(p+q)n}{2}, \lambda_2 = \frac{(p-q)n}{2}, \lambda_i = 0 \) for \( i \geq 3 \).

Typically, \( \frac{(p-q)n}{2} \) will be the minimum of these two gaps.

A adjacency matrix of random stochastic block model graph. \( p \): connection probability within clusters. \( q < p \): connection probability between clusters. \( n \): number of nodes. \( v_2, \bar{v}_2 \): second eigenvectors of \( A \) and \( \mathbb{E}[A] \) respectively.
So Far: \(\sin \theta(v_2, \bar{v}_2) \leq O\left(\frac{\sqrt{p}}{(p-q)\sqrt{n}}\right)\).
APPLICATION TO STOCHASTIC BLOCK MODEL

So Far: \( \sin \theta(v_2, \bar{v}_2) \leq O \left( \frac{\sqrt{p}}{(p-q)\sqrt{n}} \right) \). What does this give us?

- Can show that this implies \( \|v_2 - \bar{v}_2\|_2^2 \leq O \left( \frac{p}{(p-q)^2 n} \right) \) (exercise).

A adjacency matrix of random stochastic block model graph. \( p \): connection probability within clusters. \( q < p \): connection probability between clusters. \( n \): number of nodes. \( v_2, \bar{v}_2 \): second eigenvectors of \( A \) and \( \mathbb{E}[A] \) respectively.
APPLICATION TO STOCHASTIC BLOCK MODEL

So Far: \( \sin \theta(v_2, \bar{v}_2) \leq O \left( \frac{\sqrt{p}}{(p-q)\sqrt{n}} \right) \). What does this give us?

- Can show that this implies \( \|v_2 - \bar{v}_2\|_2^2 \leq O \left( \frac{p}{(p-q)^2n} \right) \) (exercise).
- \( \bar{v}_2 \) is \( \frac{1}{\sqrt{n}} \chi_{B,C} \): the community indicator vector.

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1
\end{bmatrix}
\]

\[\bar{v}_2\]

A adjacency matrix of random stochastic block model graph. \( p \): connection probability within clusters. \( q < p \): connection probability between clusters. \( n \): number of nodes. \( v_2, \bar{v}_2 \): second eigenvectors of \( A \) and \( \mathbb{E}[A] \) respectively.
So Far: \( \sin \theta(v_2, \bar{v}_2) \leq O \left( \frac{\sqrt{p}}{(p-q)\sqrt{n}} \right) \). What does this give us?

- Can show that this implies \( \|v_2 - \bar{v}_2\|_2^2 \leq O \left( \frac{p}{(p-q)^2n} \right) \) (exercise).
- \( \bar{v}_2 \) is \( \frac{1}{\sqrt{n}} \chi_{B,c} \): the community indicator vector.

\[
\begin{align*}
\begin{array}{c}
V_2(i) \\
\bar{V}_2(i)
\end{array}
& \leq 0 \\
& \frac{1}{\sqrt{n}} \\
& \text{second eig. of } A \\
& \text{Every } i \text{ where } v_2(i), \bar{v}_2(i) \text{ differ in sign contributes } \geq \frac{1}{n} \text{ to } \|v_2 - \bar{v}_2\|_2^2.
\end{align*}
\]

---

A adjacency matrix of random stochastic block model graph. \( p \): connection probability within clusters. \( q < p \): connection probability between clusters. \( n \): number of nodes. \( v_2, \bar{v}_2 \): second eigenvectors of \( A \) and \( \mathbb{E}[A] \) respectively.
APPLICATION TO STOCHASTIC BLOCK MODEL

So Far: \( \sin \theta(v_2, \bar{v}_2) \leq O \left( \frac{\sqrt{p}}{(p-q)\sqrt{n}} \right) \). What does this give us?

- Can show that this implies \( \|v_2 - \bar{v}_2\|_2^2 \leq O \left( \frac{p}{(p-q)^2 n} \right) \) (exercise).
- \( \bar{v}_2 \) is \( \frac{1}{\sqrt{n}} \chi_{B,C} \): the community indicator vector.

\[
\begin{align*}
\|v_2 - \bar{v}_2\|_2^2 &\geq \frac{1}{n} \sum_{i=1}^{n} |v_2(i) - \bar{v}_2(i)|^2 \\
\text{Every } i \text{ where } v_2(i), \bar{v}_2(i) \text{ differ in sign contributes } &\geq \frac{1}{n} \text{ to } \|v_2 - \bar{v}_2\|_2^2.
\end{align*}
\]

- So they differ in sign in at most \( O \left( \frac{p}{(p-q)^2} \right) \) positions.

\[
A \text{ adjacency matrix of random stochastic block model graph. } p: \text{ connection probability within clusters. } q < p: \text{ connection probability between clusters. } n: \text{ number of nodes. } v_2, \bar{v}_2: \text{ second eigenvectors of } A \text{ and } \mathbb{E}[A] \text{ respectively.}
\]
**Upshot:** If $G$ is a stochastic block model graph with adjacency matrix $A$, if we compute its second large eigenvector $v_2$ and assign nodes to communities according to the sign pattern of this vector, we will correctly assign all but $O\left(\frac{p}{(p-q)^2}\right)$ nodes.

![Diagram](image.png)
**Upshot:** If $G$ is a stochastic block model graph with adjacency matrix $A$, if we compute its second large eigenvector $v_2$ and assign nodes to communities according to the sign pattern of this vector, we will correctly assign all but $O\left(\frac{p}{(p-q)^2}\right)$ nodes.

- Why does the error increase as $q$ gets close to $p$?
APPLICATION TO STOCHASTIC BLOCK MODEL

\[ A = \begin{bmatrix} P & Q \\ Q & P \end{bmatrix} \]

Generative models

**Upshot:** If \( G \) is a stochastic block model graph with adjacency matrix \( A \), if we compute its second large eigenvector \( v_2 \) and assign nodes to communities according to the sign pattern of this vector, we will correctly assign all but \( O\left(\frac{p}{(p-q)^2}\right) \) nodes.

\[ B = \begin{bmatrix} \frac{1}{\sqrt{n}} & -\frac{1}{\sqrt{n}} \\ \frac{1}{\sqrt{n}} & -\frac{1}{\sqrt{n}} \end{bmatrix}, C = \begin{bmatrix} \frac{1}{\sqrt{n}} & -\frac{1}{\sqrt{n}} \\ \frac{1}{\sqrt{n}} & -\frac{1}{\sqrt{n}} \end{bmatrix} \]

- Why does the error increase as \( q \) gets close to \( p \)?
- Even when \( p - q = O(1/\sqrt{n}) \), assign all but an \( O(n) \) fraction of nodes correctly. E.g., assign 99% of nodes correctly.
Questions on spectral partitioning?
We have talked about the eigendecomposition and SVD as ways to compress data, to embed entities like words and documents, to compress/cluster non-linearly separable data.

How efficient are these techniques? Can they be run on massive datasets?
To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U \Sigma V^T$, first compute $V$. Then compute $U \Sigma = AV$.

$AV = U \Sigma V^T = U \Sigma$.\*
COMPUTING THE SVD

To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U \Sigma V^T$, first compute $V$. Then compute $U \Sigma = AV$.

- Compute $A^T A = O(n d^2)$ runtime.
COMPUTING THE SVD

To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U \Sigma V^T$, first compute $V$. Then compute $U \Sigma = AV$.

- Compute $A^T A - O(nd^2)$ runtime.
- Find eigendecomposition $A^T A = V \Lambda V^T - O(d^3)$ runtime.
To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U \Sigma V^T$, first compute $V$. Then compute $U \Sigma = AV$.

- Compute $A^T A - O(nd^2)$ runtime.
- Find eigendecomposition $A^T A = \Sigma \Lambda V^T - O(d^3)$ runtime.
- Compute $L = AV$. Set $\sigma_i = \|L_i\|_2$ and $U_i = L_i/\|L_i\|_2$. $- O(nd^2)$ runtime.
To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U \Sigma V^T$, first compute $V$. Then compute $U \Sigma = AV$.

- Compute $A^T A - O(nd^2)$ runtime.
- Find eigendecomposition $A^T A = V \Lambda V^T - O(d^3)$ runtime.
- Compute $L = AV$. Set $\sigma_i = \|L_i\|_2$ and $U_i = L_i/\|L_i\|_2$. $- O(nd^2)$ runtime.

**Total runtime:** $O(nd^2 + d^3)$
To compute the SVD of $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$, first compute $\mathbf{V}$. Then compute $\mathbf{U} \mathbf{\Sigma} = \mathbf{AV}$.

- Compute $\mathbf{A}^T \mathbf{A} - O(n d^2)$ runtime.
- Find eigendecomposition $\mathbf{A}^T \mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T - O(d^3)$ runtime.
- Compute $\mathbf{L} = \mathbf{AV}$. Set $\sigma_i = \|L_i\|_2$ and $\mathbf{U}_i = L_i / \|L_i\|_2$. $- O(n d^2)$ runtime.

**Total runtime:** $O(n d^2 + d^3) = O(n d^2)$ (assume w.l.o.g. $n \geq d$)
To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U \Sigma V^T$, first compute $V$. Then compute $U \Sigma = AV$.

- Compute $A^T A - O(nd^2)$ runtime.
- Find eigendecomposition $A^T A = V \Lambda V^T - O(d^3)$ runtime.
- Compute $L = AV$. Set $\sigma_i = \|L_i\|_2$ and $U_i = L_i / \|L_i\|_2$. $- O(nd^2)$ runtime.

**Total runtime:** $O(nd^2 + d^3) = O(nd^2)$ (assume w.l.o.g. $n \geq d$)

- If we have $n = 10$ million images with $200 \times 200 \times 3 = 120,000$ pixel values each, runtime is $1.5 \times 10^{17}$ operations!
To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U \Sigma V^T$, first compute $V$. Then compute $U \Sigma = AV$.

- Compute $A^T A = O(nd^2)$ runtime.
- Find eigendecomposition $A^T A = V \Lambda V^T = O(d^3)$ runtime.
- Compute $L = AV$. Set $\sigma_i = ||L_i||_2$ and $U_i = L_i/||L_i||_2$. $- O(nd^2)$ runtime.

**Total runtime:** $O(nd^2 + d^3) = O(nd^2)$ (assume w.l.o.g. $n \geq d$)

- If we have $n = 10$ million images with $200 \times 200 \times 3 = 120,000$ pixel values each, runtime is $1.5 \times 10^{17}$ operations!
- The worlds fastest super computers compute at $\approx 100$ petaFLOPS = $10^{17}$ FLOPS (floating point operations per second).
To compute the SVD of $A \in \mathbb{R}^{n \times d}$, $A = U\Sigma V^T$, first compute $V$. Then compute $U\Sigma = AV$.

- Compute $A^T A$ – $O(nd^2)$ runtime.
- Find eigendecomposition $A^T A = V\Lambda V^T$ – $O(d^3)$ runtime.
- Compute $L = AV$. Set $\sigma_i = \|L_i\|_2$ and $U_i = L_i/\|L_i\|_2$. – $O(nd^2)$ runtime.

**Total runtime:** $O(nd^2 + d^3) = O(nd^2)$ (assume w.l.o.g. $n \geq d$)

- If we have $n = 10$ million images with $200 \times 200 \times 3 = 120,000$ pixel values each, runtime is $1.5 \times 10^{17}$ operations!

- The worlds fastest super computers compute at $\approx 100$ petaFLOPS = $10^{17}$ FLOPS (floating point operations per second).

- This is an easy task for them – but no one else.
To speed up SVD computation we will take advantage of the fact that we typically only care about computing the top (or bottom) $k$ singular vectors for $k \ll d$. 
To speed up SVD computation we will take advantage of the fact that we typically only care about computing the top (or bottom) $k$ singular vectors for $k \ll d$.

- Suffices to compute $V_k \in \mathbb{R}^{d \times k}$ and then compute $U_k \Sigma_k = AV_k$. 

---

\[ \begin{array}{cc}
\end{array} \]
To speed up SVD computation we will take advantage of the fact that we typically only care about computing the top (or bottom) $k$ singular vectors for $k \ll d$.

- Suffices to compute $V_k \in \mathbb{R}^{d \times k}$ and then compute $U_k \Sigma_k = AV_k$.
- Use an iterative algorithm to compute an approximation to the top $k$ singular vectors $V_k$. 
To speed up SVD computation we will take advantage of the fact that we typically only care about computing the top (or bottom) $k$ singular vectors for $k \ll d$.

- Suffices to compute $V_k \in \mathbb{R}^{d \times k}$ and then compute $U_k \Sigma_k = AV_k$.
- Use an iterative algorithm to compute an approximation to the top $k$ singular vectors $V_k$.
- Runtime will be roughly $O(ndk)$ instead of $O(nd^2)$.
FASTER ALGORITHMS

To speed up SVD computation we will take advantage of the fact that we typically only care about computing the top (or bottom) $k$ singular vectors for $k \ll d$.

- Suffices to compute $V_k \in \mathbb{R}^{d \times k}$ and then compute $U_k \Sigma_k = AV_k$.
- Use an iterative algorithm to compute an approximation to the top $k$ singular vectors $V_k$.
- Runtime will be roughly $O(ndk)$ instead of $O(nd^2)$.

Won’t cover: randomized methods, which can be much faster in some cases.
In numerical linear algebra, two main types of methods:

Direct Methods: Gaussian elimination, QR decomposition, Cholesky decomposition, etc.

\[
O(n^2) \quad O(n^3)
\]

· Directly manipulate the entries of the input matrix \( A \). Typically run in \( O(n^3) \) time for an \( n \times n \) matrix.
In numerical linear algebra, two main types of methods:

**Direct Methods:** Gaussian elimination, QR decomposition, Cholesky decomposition, etc.

- Directly manipulate the entries of the input matrix $A$. Typically run in $O(n^3)$ time for an $n \times n$ matrix.

**Sparse (Iterative) Methods:** Conjugate gradient, Gauss-Seidel, Krylov subspace methods, Lanczos, gradient descent.

- Generally only access $A$ via a sequence of matrix vector multiplications. $Ax_1, Ax_2, \ldots, Ax_t$. 
In numerical linear algebra, two main types of methods:

**Direct Methods:** Gaussian elimination, QR decomposition, Cholesky decomposition, etc.

- Directly manipulate the entries of the input matrix $A$. Typically run in $O(n^3)$ time for an $n \times n$ matrix.

**Sparse (Iterative) Methods:** Conjugate gradient, Gauss-Seidel, Krylov subspace methods, Lanczos, gradient descent.

- Generally only access $A$ via a sequence of matrix vector multiplications. $Ax_1, Ax_2, \ldots, Ax_t$.
- Runtime is $\#$ iterations $t \times$ matrix vector multiplication time $= O(\text{nnz}(A) \cdot t) = O(ndt)$ where $\text{nnz}(A)$ is the number of nonzero entries in $A$. 
In numerical linear algebra, two main types of methods:

**Direct Methods:** Gaussian elimination, QR decomposition, Cholesky decomposition, etc.

- Directly manipulate the entries of the input matrix $A$. Typically run in $O(n^3)$ time for an $n \times n$ matrix.

**Sparse (Iterative) Methods:** Conjugate gradient, Gauss-Seidel, Krylov subspace methods, Lanczos, gradient descent.

- Generally only access $A$ via a sequence of matrix vector multiplications. $Ax_1, Ax_2, \ldots, Ax_t$.
- Runtime is $\#$ iterations $t \times$ matrix vector multiplication time $= O(\text{nnz}(A) \cdot t) = O(ntd)$ where nnz($A$) is the number of nonzero entries in $A$.
- Not just for sparse matrices!
Matlab:

```
svd and eig vs. svds and eigs
```

SciPy (Python):

```
scipy.linalg.svd vs. scipy.sparse.linalg.svd
```

\(\text{direct} \quad \text{(slow)}\)

\(\text{iterative} \quad \text{(fast)}\)
Power Method: The most fundamental iterative method for approximate SVD. Applies to computing $k = 1$ singular vectors.
Power Method: The most fundamental iterative method for approximate SVD. Applies to computing \( k = 1 \) singular vectors.

Goal: Given \( A \in \mathbb{R}^{n \times d} \), with SVD \( A = U \Sigma V \), find \( \tilde{z} \approx \tilde{v}_1 \).
**Power Method:** The most fundamental iterative method for approximate SVD. Applies to computing $k = 1$ singular vectors.

**Goal:** Given $A \in \mathbb{R}^{n \times d}$, with SVD $A = U\Sigma V$, find $\tilde{z} \approx \tilde{v}_1$.

- Choose $\tilde{z}^{(0)}$ randomly. E.g. $\tilde{z}^{(0)}(i) \sim \mathcal{N}(0, 1)$.
- For $i = 1, \ldots, t$
  - $\tilde{z}^{(i)} = A^T \cdot (A\tilde{z}^{(i-1)})$
  - $n_i = \|\tilde{z}^{(i)}\|_2$
  - $\tilde{z}^{(i)} = \tilde{z}^{(i)} / n_i$

Return $\tilde{Z}_t$
Power Method: The most fundamental iterative method for approximate SVD. Applies to computing $k = 1$ singular vectors.

Goal: Given $A \in \mathbb{R}^{n \times d}$, with SVD $A = U \Sigma V$, find $\tilde{z} \approx v_1$.

- Choose $\tilde{z}^{(0)}$ randomly. E.g. $\tilde{z}^{(0)}(i) \sim \mathcal{N}(0, 1)$.
- For $i = 1, \ldots, t$
  - $\tilde{z}^{(i)} = A^T \cdot (A \tilde{z}^{(i-1)})$
  - $n_i = \|\tilde{z}^{(i)}\|_2$
  - $\tilde{z}^{(i)} = \tilde{z}^{(i)} / n_i$

Return $\tilde{z}_t$

Total Runtime: $O(ndt)$
Write $\tilde{z}^{(0)}$ in the right singular vector basis:

$$
\tilde{z}^{(0)} = c_1 \tilde{V}_1 + c_2 \tilde{V}_2 + \ldots + c_d \tilde{V}_d
$$
Write $\vec{z}^{(0)}$ in the right singular vector basis:

$$\vec{z}^{(0)} = c_1 \vec{v}_1 + c_2 \vec{v}_2 + \ldots + c_d \vec{v}_d$$

Update step: $\vec{z}^{(i)} = A^T \cdot (A \vec{z}^{(i-1)}) = V \Sigma^2 V^T \vec{z}^{(i-1)}$ (then normalize)
POWER METHOD INTUITION

Write $\bar{z}^{(0)}$ in the right singular vector basis:

$$\bar{z}^{(0)} = c_1 \vec{v}_1 + c_2 \vec{v}_2 + \ldots + c_d \vec{v}_d$$

Update step: $\bar{z}^{(i)} = A^T \cdot (A\bar{z}^{(i-1)}) = V \Sigma^2 V^T \bar{z}^{(i-1)}$ (then normalize)

Claim:

$$\bar{z}^{(1)} = \frac{1}{n_1} \left[ c_1 \cdot \sigma_1^2 \vec{v}_1 + c_2 \cdot \sigma_2^2 \vec{v}_2 + \ldots + c_d \cdot \sigma_d^2 \vec{v}_d \right]$$
Claim:

\[ \begin{aligned}
  \tilde{z}^{(t)} &= \frac{1}{\prod_{i=1}^{t} n_i} \left[ c_1 \cdot \sigma_1^{2t} \vec{v}_1 + c_2 \cdot \sigma_2^{2t} \vec{v}_2 + \ldots + c_d \cdot \sigma_d^{2t} \vec{v}_d \right]
\end{aligned} \]

After \( t \) iterations, you have ‘powered’ up the singular values, making the component in the direction of \( \vec{v}_1 \) much larger, relative to the other components.
Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$ be parameter capturing the “gap” between the first and second largest singular values. If Power Method is initialized with a random Gaussian vector then, with high probability, after $t = O\left(\frac{\log d/\epsilon}{\gamma}\right)$ steps:

$$\|\vec{v}_1 - \vec{z}^{(t)}\|_2 \leq \epsilon.$$
Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$ be parameter capturing the “gap” between the first and second largest singular values. If Power Method is initialized with a random Gaussian vector then, with high probability, after $t = O\left(\frac{\log d/\epsilon}{\gamma}\right)$ steps:

$$\|\vec{v}_1 - \vec{z}^{(t)}\|_2 \leq \epsilon.$$ 

Total runtime: $O\left(\text{nnz}(A) \cdot \frac{\log d/\epsilon}{\gamma}\right) = O\left(nd \cdot \frac{\log d/\epsilon}{\gamma}\right).$
Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$ be parameter capturing the “gap” between the first and second largest singular values. If Power Method is initialized with a random Gaussian vector then, with high probability, after $t = O \left( \frac{\log d/\epsilon}{\gamma} \right)$ steps:

$$\| \vec{v}_1 - \vec{z}^{(t)} \|_2 \leq \epsilon.$$

**Total runtime:** $O \left( \text{nnz}(A) \cdot \frac{\log d/\epsilon}{\gamma} \right) = O \left( nd \cdot \frac{\log d/\epsilon}{\gamma} \right)$.

**Next Time:** Will analyze this method formally.