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:

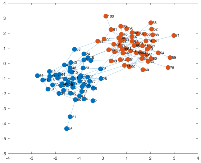
- Finish up stochastic block model.
- Efficient algorithms for SVD/eigendecomposition.
- Iterative methods: power method, Krylov subspace methods.

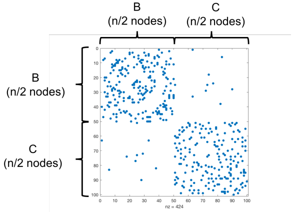
Goal: Argue the effectiveness of spectral clustering in a natural, if oversimplified, generative model.

Goal: Argue the effectiveness of spectral clustering in a natural, if oversimplified, generative model.

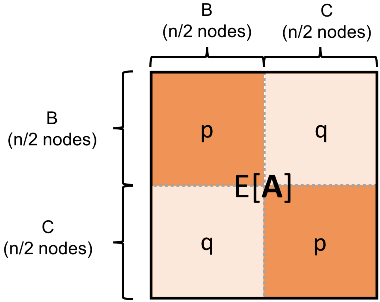
Stochastic Block Model (Planted Partition Model): Let $G_n(p, q)$ be a distribution over graphs on n nodes, split equally into two groups B and C, each with $n/2$ nodes.

- 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.





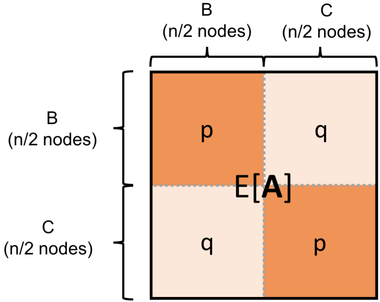
Letting G be a stochastic block model graph drawn from $G_n(p, q)$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$ be its adjacency matrix. $(\mathbb{E}[\mathbf{A}])_{i,j} = p$ for i, j in same group, $(\mathbb{E}[\mathbf{A}])_{i,j} = q$ otherwise.



$G_n(p, q)$: 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.

EXPECTED ADJACENCY SPECTRUM

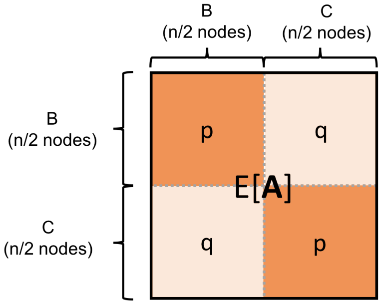
Letting G be a stochastic block model graph drawn from $G_n(p,q)$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$ be its adjacency matrix. $(\mathbb{E}[\mathbf{A}])_{i,j} = p$ for i,j in same group, $(\mathbb{E}[\mathbf{A}])_{i,j} = q$ otherwise.



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

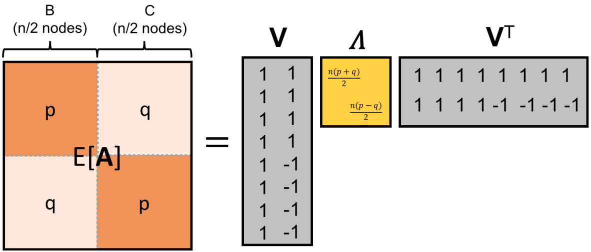
$G_n(p,q)$: stochastic block model distribution. B, C: groups with $n/2$ nodes each. Connections are independent with probability p between nodes in the same group, and probability q between nodes not in the same group.

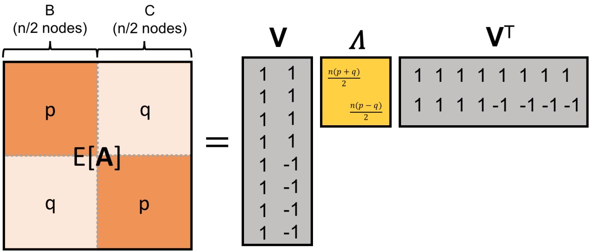
Letting G be a stochastic block model graph drawn from $G_n(p, q)$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$ be its adjacency matrix. $(\mathbb{E}[\mathbf{A}])_{i,j} = p$ for i, j in same group, $(\mathbb{E}[\mathbf{A}])_{i,j} = q$ otherwise.



What is the rank of $\mathbb{E}[\mathbf{A}]$ and how can you see this quickly?
How many nonzero eigenvalues does $\mathbb{E}[\mathbf{A}]$ have?

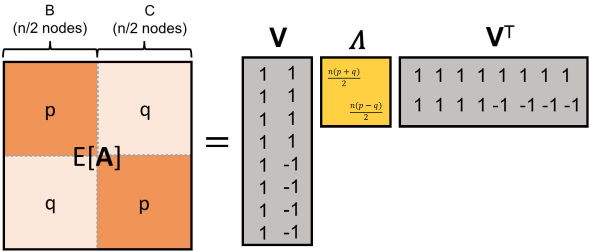
$G_n(p, q)$: stochastic block model distribution. B, C: groups with $n/2$ nodes each. Connections are independent with probability p between nodes in the same group, and probability q between nodes not in the same group.


B
(n/2 nodes)
C
(n/2 nodes)
p
q
q
p
E[A]
=
V
Λ
V^T
1 1
1 1
1 1
1 1
1 -1
1 -1
1 -1
1 -1
n(p + q)/2
n(p − q)/2
1 1 1 1 1 1 1 1
1 1 1 1 -1 -1 -1 -1



- $\vec{v}_1 = \vec{1}$ with eigenvalue $\lambda_1 = \frac{(p+q)n}{2}$.
- $\vec{v}_2 = \chi_{B,C}$ with eigenvalue $\lambda_2 = \frac{(p-q)n}{2}$.
- $\chi_{B,C}(i) = 1$ if $i \in B$ and $\chi_{B,C}(i) = -1$ for $i \in C$.

EXPECTED ADJACENCY SPECTRUM



- $\vec{v}_1 = \vec{1}$ with eigenvalue $\lambda_1 = \frac{(p+q)n}{2}$.
- $\vec{v}_2 = \chi_{B,C}$ with eigenvalue $\lambda_2 = \frac{(p-q)n}{2}$.
- $\chi_{B,C}(i) = 1$ if $i \in B$ and $\chi_{B,C}(i) = -1$ for $i \in C$.

If we compute $\vec{v}_2$ then we recover the communities B and C!

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 $\mathbb{E}[\mathbf{L}]$?

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

Upshot: The second small eigenvector of $\mathbb{E}[\mathbf{L}]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

Upshot: The second small eigenvector of $\mathbb{E}[\mathbf{L}]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph G (equivilantly $\mathbf{A}$ and $\mathbf{L}$) 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 $\mathbb{E}[\mathbf{L}]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph G (equivilantly $\mathbf{A}$ and $\mathbf{L}$) 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., $\mathbf{A}$) is close to its expectation? Matrix concentration inequalities.

Upshot: The second small eigenvector of $\mathbb{E}[\mathbf{L}]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph G (equivilantly $\mathbf{A}$ and $\mathbf{L}$) 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., $\mathbf{A}$) is close to its expectation? Matrix concentration inequalities.

- Analogous to scalar concentration inequalities like Markovs, Chebyshevs, Bernsteins.

Upshot: The second small eigenvector of $\mathbb{E}[\mathbf{L}]$ is $\chi_{B,C}$ – the indicator vector for the cut between the communities.

- If the random graph G (equivilantly $\mathbf{A}$ and $\mathbf{L}$) 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., $\mathbf{A}$) 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

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

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

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

Matrix Concentration Inequality: If $p \geq O\left(\frac{\log^4 n}{n}\right)$, then with high probability

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

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

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

Exercise: Show that $\|\mathbf{X}\|_2$ is equal to the largest singular value of $\mathbf{X}$. For symmetric $\mathbf{X}$ (like $\mathbf{A} - \mathbb{E}[\mathbf{A}]$) 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

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

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

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

Exercise: Show that $\|\mathbf{X}\|_2$ is equal to the largest singular value of $\mathbf{X}$. For symmetric $\mathbf{X}$ (like $\mathbf{A} - \mathbb{E}[\mathbf{A}]$) 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 $\mathbf{A}$ and $\mathbb{E}[\mathbf{A}]$ are close. How does this relate to their difference in spectral norm?

Davis-Kahan Eigenvector Perturbation Theorem: Suppose $\mathbf{A}, \overline{\mathbf{A}} \in \mathbb{R}^{d \times d}$ are symmetric with $\|\mathbf{A} - \overline{\mathbf{A}}\|_2 \leq \epsilon$ and eigenvectors $v_1, v_2, \ldots, v_d$ and $\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_d$. Letting $\theta(v_i, \bar{v}_i)$ denote the angle between v_i and $\bar{v}_i$, for all i:

$$\sin[\theta(v_i, \bar{v}_i)] \leq \frac{\epsilon}{\min_{j \neq i} |\lambda_i - \lambda_j|}$$

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

The errors get large if there are eigenvalues with similar magnitudes.

EIGENVECTOR PERTURBATION

$$\overset{\mathbf{A}}{\begin{bmatrix} 1+\varepsilon & 0 \\ 0 & 1 \end{bmatrix}} - \overset{\bar{\mathbf{A}}}{\begin{bmatrix} 1 & 0 \\ 0 & 1+\varepsilon \end{bmatrix}} = \overset{\mathbf{A}-\bar{\mathbf{A}}}{\begin{bmatrix} \varepsilon & 0 \\ 0 & \varepsilon \end{bmatrix}}$$

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

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

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 $\mathbb{E}[\mathsf{A}]$ respectively.

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

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

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

$\mathbf{A}$ 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 $\mathbf{A}$ and $\mathbb{E}[\mathbf{A}]$ respectively.

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

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

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

$$\min_{j \neq i} |\lambda_i - \lambda_j| = \min\left(qn, \frac{(p-q)n}{2}\right).$$

$\mathbf{A}$ 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 $\mathbf{A}$ and $\mathbb{E}[\mathbf{A}]$ respectively.

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

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

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: $\mathbb{E}[\mathbf{A}]$, 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.

> $\mathbf{A}$ 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 $\mathbf{A}$ and $\mathbb{E}[\mathbf{A}]$ respectively.

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

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

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: $\mathbb{E}[\mathbf{A}]$, 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.

$\mathbf{A}$ 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 $\mathbf{A}$ and $\mathbb{E}[\mathbf{A}]$ respectively.

So Far: $\sin\theta(v_2, \bar{v}_2) \leq O\left(\frac{\sqrt{p}}{(p-q)\sqrt{n}}\right)$.

$\mathbf{A}$ 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 $\mathbf{A}$ and $\mathbb{E}[\mathbf{A}]$ respectively.

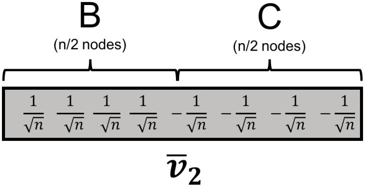
So Far: $\sin\theta(v_2, \bar{v}_2) \leq O\left(\frac{\sqrt{p}}{(p-q)\sqrt{n}}\right)$. What does this give us?

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

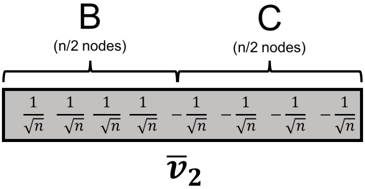


A adjacency matrix of random stochastic block model graph. p: connection probability within clusters. $q < p$: connection probability between clusters. n: number of nodes. $v_2, \bar{v}_2$: second eigenvectors of A and $\mathbb{E}[A]$ respectively.

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.



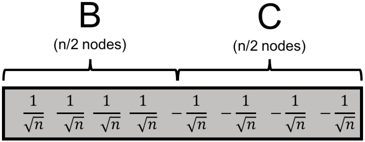
- Every i where $v_2(i), \bar{v}_2(i)$ differ in sign contributes $\geq \frac{1}{n}$ to $\|v_2 - \bar{v}_2\|_2^2$.

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 $\mathbb{E}[\mathbf{A}]$ 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.

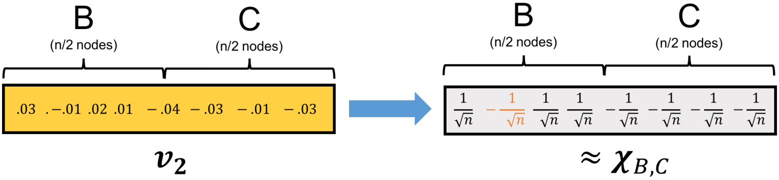


$\bar{v}_2$

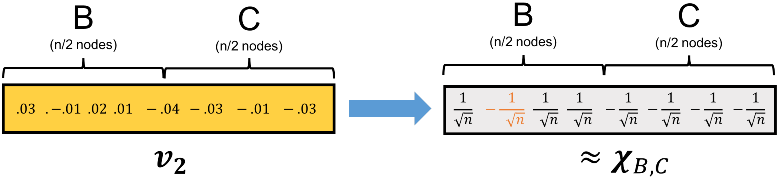
- Every i where $v_2(i), \bar{v}_2(i)$ differ in sign contributes $\geq \frac{1}{n}$ to $\|v_2 - \bar{v}_2\|_2^2$.
- So they differ in sign in at most $O\left(\frac{p}{(p-q)^2}\right)$ positions.

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 $\mathbb{E}[\mathbf{A}]$ respectively.

Upshot: If G is a stochastic block model graph with adjacency matrix $\mathbf{A}$, 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.

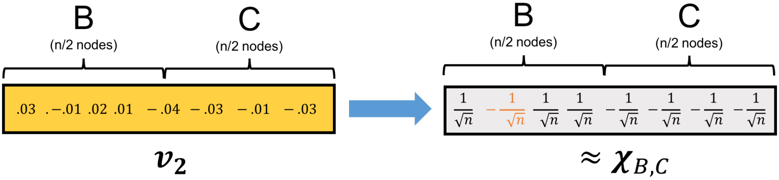


Upshot: If G is a stochastic block model graph with adjacency matrix $\mathbf{A}$, if we compute its second large eigenvector v_2 and assign nodes to communities according to the sign pattern of this vector, we will correctly assign all but $O\left(\frac{p}{(p-q)^2}\right)$ nodes.



- Why does the error increase as q gets close to p?

Upshot: If G is a stochastic block model graph with adjacency matrix $\mathbf{A}$, 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?
- 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 $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, first compute $\mathbf{V}$. Then compute $\mathbf{U}\boldsymbol{\Sigma} = \mathbf{A}\mathbf{V}$.

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

- Compute $\mathbf{A}^T\mathbf{A}$ – $O(nd^2)$ runtime.

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

- Compute $\mathbf{A}^T\mathbf{A}$ – $O(nd^2)$ runtime.
- Find eigendecomposition $\mathbf{A}^T\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$ – $O(d^3)$ runtime.

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 $\mathbf{U}\mathbf{\Sigma} = \mathbf{A}\mathbf{V}$.

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

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

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

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

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

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

- Compute $\mathbf{A}^T\mathbf{A}$ – $O(nd^2)$ runtime.
- Find eigendecomposition $\mathbf{A}^T\mathbf{A} = \mathbf{V\Lambda V}^T$ – $O(d^3)$ runtime.
- Compute $\mathbf{L} = \mathbf{AV}$. Set $\sigma_i = \|\mathbf{L}_i\|_2$ and $\mathbf{U}_i = \mathbf{L}_i / \|\mathbf{L}_i\|_2$. – $O(nd^2)$ 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 1.5×10^{17} operations!

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

- Compute $\mathbf{A}^T\mathbf{A}$ – $O(nd^2)$ runtime.
- Find eigendecomposition $\mathbf{A}^T\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$ – $O(d^3)$ runtime.
- Compute $\mathbf{L} = \mathbf{A}\mathbf{V}$. Set $\sigma_i = \|\mathbf{L}_i\|_2$ and $\mathbf{U}_i = \mathbf{L}_i / \|\mathbf{L}_i\|_2$. – $O(nd^2)$ 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 1.5×10^{17} operations!
- The worlds fastest super computers compute at $\approx$ 100 petaFLOPS = 10^{17} FLOPS (floating point operations per second).

COMPUTING THE SVD

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 $\mathbf{U}\mathbf{\Sigma} = \mathbf{A}\mathbf{V}$.

- Compute $\mathbf{A}^T\mathbf{A}$ – $O(nd^2)$ runtime.
- Find eigendecomposition $\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ – $O(d^3)$ runtime.
- Compute $\mathbf{L} = \mathbf{A}\mathbf{V}$. Set $\sigma_i = \|\mathbf{L}_i\|_2$ and $\mathbf{U}_i = \mathbf{L}_i / \|\mathbf{L}_i\|_2$. – $O(nd^2)$ 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 1.5×10^{17} operations!
- The worlds fastest super computers compute at ≈ 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 $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ and then compute $\mathbf{U}_k \boldsymbol{\Sigma}_k = \mathbf{A}\mathbf{V}_k$.

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 $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ and then compute $\mathbf{U}_k \boldsymbol{\Sigma}_k = \mathbf{A}\mathbf{V}_k$.
- Use an *iterative algorithm* to compute an *approximation* to the top k singular vectors $\mathbf{V}_k$.

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 $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ and then compute $\mathbf{U}_k \boldsymbol{\Sigma}_k = \mathbf{A}\mathbf{V}_k$.
- Use an *iterative algorithm* to compute an *approximation* to the top k singular vectors $\mathbf{V}_k$.
- Runtime will be roughly $O(ndk)$ instead of $O(nd^2)$.

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 $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ and then compute $\mathbf{U}_k \mathbf{\Sigma}_k = \mathbf{A}\mathbf{V}_k$.
- Use an *iterative algorithm* to compute an *approximation* to the top k singular vectors $\mathbf{V}_k$.
- 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.

- Directly manipulate the entries of the input matrix $\mathbf{A}$. 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 $\mathbf{A}$. 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 $\mathbf{A}$ via a sequence of matrix vector multiplications. $\mathbf{A}\mathbf{x}_1, \mathbf{A}\mathbf{x}_2, \ldots, \mathbf{A}\mathbf{x}_t$.

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 $\mathbf{A}$. 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 $\mathbf{A}$ via a sequence of matrix vector multiplications. $\mathbf{A}\mathbf{x}_1, \mathbf{A}\mathbf{x}_2, \ldots, \mathbf{A}\mathbf{x}_t$.
- Runtime is $\#$ iterations $t \times$ matrix vector multiplication time $= O(\text{nnz}(\mathbf{A}) \cdot t) = O(ndt)$ where $\text{nnz}(\mathbf{A})$ is the number of nonzero entries in $\mathbf{A}$.

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 $\mathbf{A}$. 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 $\mathbf{A}$ via a sequence of matrix vector multiplications. $\mathbf{A}\mathbf{x}_1, \mathbf{A}\mathbf{x}_2, \ldots, \mathbf{A}\mathbf{x}_t$.
- Runtime is $\#$ iterations $t \times$ matrix vector multiplication time $= O(\text{nnz}(\mathbf{A}) \cdot t) = O(ndt)$ where $\text{nnz}(\mathbf{A})$ is the number of nonzero entries in $\mathbf{A}$.
- Not just for sparse matrices!

Matlab:

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

SciPy (Python):

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

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 $\mathbf{A} \in \mathbb{R}^{n \times d}$, with SVD $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}$, find $\vec{z} \approx \vec{v}_1$.

Power Method: The most fundamental iterative method for approximate SVD. Applies to computing $k = 1$ singular vectors.

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

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

  Return $\vec{z}_t$

Power Method: The most fundamental iterative method for approximate SVD. Applies to computing $k = 1$ singular vectors.

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

- Choose $\vec{z}^{(0)}$ randomly. E.g. $\vec{z}^{(0)}(i) \sim \mathcal{N}(0, 1)$.
- For $i = 1, \ldots, t$
  - $\vec{z}^{(i)} = \mathbf{A}^T \cdot (\mathbf{A}\vec{z}^{(i-1)})$ Runtime: $2 \cdot nd$
  - $n_i = \|\vec{z}^{(i)}\|_2$ Runtime: d
  - $\vec{z}^{(i)} = \vec{z}^{(i)}/n_i$ Runtime: d

  Return $\vec{z}_t$

Total Runtime: $O(ndt)$

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

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

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

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

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

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

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

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

Claim:

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

Claim:

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

After t iterations, you have 'powered' up the singular values, making the component in the direction of 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: $O\left(\text{nnz}(\mathbf{A}) \cdot \frac{\log d/\epsilon}{\gamma} \cdot\right) = O\left(nd \cdot \frac{\log d/\epsilon}{\gamma} \cdot\right).$

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

Next Time: Will analyze this method formally.