COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2024.

Lecture 21

Logistics

- Problem Set 4 due Monday.
- No class or office hours next week. No quiz due.
- Office hours tomorrow 10am-11am in CS 142.
- Practice final exams have been posted in Canvas. I will release a more complete study guide with additional practice questions soon.

Summary

Last Class: Fast computation of the SVD/eigendecomposition.

- Power method for approximating the top eigenvector of a matrix.
- Start on analysis of convergence.

This Class (+ Rest of Semester):

krylov methods

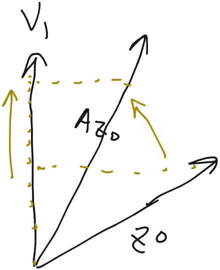
- connections to random walks/ markov chains

- Finish up power method analysis.
- General iterative algorithms for optimization, specifically gradient descent and its variants.
- What are these methods, when are they applied, and how do you analyze their performance?
- Small taste of what you can find in COMPSCI 651.

Power Method Wrap Up

Basic Power Method: $A \in \mathbb{R}^{d \times 2}$

- **Initialize:** Choose $\vec{z}^{(0)}$ randomly. E.g. $\vec{z}^{(0)}(i) \sim \mathcal{N}(0,1)$.
- For $i = 1, \ldots, t$
  - $\vec{z}^{(i)} := \mathbf{A} \cdot \vec{z}^{(i-1)}$
- $\vec{z}_i := \frac{\vec{z}^{(i)}}{\|\vec{z}^{(i)}\|_2}$.
- Return $\vec{z}_t$.



Power Method Convergence Rate

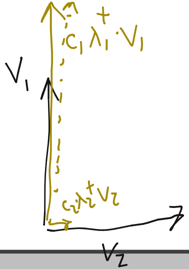
λ_1 λ_2 very large very small (relatively)

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$

Write $|\lambda_2| = (1-\gamma)|\lambda_1|$ for 'gap' $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$.

How many iterations t does it take to have $|\lambda_2|^t \leq \delta \cdot |\lambda_1|^t$ for $\delta > 0$?

$z^{(t)}$:



$$(1-\gamma)^t |\lambda_1|^t \leq \delta |\lambda_1|^t$$

$$(1-\gamma)^t \leq \delta$$

$$t \cdot \log(1-\gamma) \leq \log(\delta)$$

$$t \cdot \log\left(\frac{1}{1-\gamma}\right) \geq \log(1/\delta)$$

$$t \geq \frac{\log(1/\delta)}{\log\left(\frac{1}{1-\gamma}\right)}$$

$\vec{v}_1$: top eigenvector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$. $\lambda_1, \lambda_2, \ldots \lambda_n$: eigenvalues of $\mathbf{A}$, $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$: eigengap controlling convergence rate

Power Method Convergence Rate

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$

Write $|\lambda_2| = (1-\gamma)|\lambda_1|$ for 'gap' $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$.

How many iterations t does it take to have $|\lambda_2|^t \leq \delta \cdot |\lambda_1|^t$ for $\delta > 0$?

$$|\lambda_2|^t = (1-\gamma)^t \cdot |\lambda_1|^t$$

$\vec{v}_1$: top eigenvector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$. $\lambda_1, \lambda_2, \ldots \lambda_n$: eigenvalues of $\mathbf{A}$, $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$: eigengap controlling convergence rate

Power Method Convergence Rate

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$

Write $|\lambda_2| = (1-\gamma)|\lambda_1|$ for 'gap' $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$.

How many iterations t does it take to have $|\lambda_2|^t \le \delta \cdot |\lambda_1|^t$ for $\delta > 0$?

$$\begin{aligned}|\lambda_2|^t &= (1-\gamma)^t \cdot |\lambda_1|^t \\ &= \underbrace{(1-\gamma)^{1/\gamma}}_{\frac{1}{e}}{}^{\gamma t} \cdot |\lambda_1|^t\end{aligned}$$

$$\frac{1}{e^{\gamma t}}$$

$\vec{v}_1$: top eigenvector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$. $\lambda_1, \lambda_2, \ldots \lambda_n$: eigenvalues of $\mathbf{A}$, $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$: eigengap controlling convergence rate

Power Method Convergence Rate

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$

Write $|\lambda_2| = (1-\gamma)|\lambda_1|$ for 'gap' $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$.

How many iterations t does it take to have $|\lambda_2|^t \leq \delta \cdot |\lambda_1|^t$ for $\delta > 0$?

$$\begin{aligned} |\lambda_2|^t &= (1-\gamma)^t \cdot |\lambda_1|^t \\ &= (1-\gamma)^{1/\gamma \cdot \gamma t} \cdot |\lambda_1|^t \\ &\leq e^{-\gamma t} \cdot |\lambda_1|^t \end{aligned}$$

$\vec{v}_1$: top eigenvector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$. $\lambda_1, \lambda_2, \ldots \lambda_n$: eigenvalues of $\mathbf{A}$, $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$: eigengap controlling convergence rate

Power Method Convergence Rate

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$

Write $|\lambda_2| = (1-\gamma)|\lambda_1|$ for 'gap' $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$.

How many iterations t does it take to have $|\lambda_2|^t \le \delta \cdot |\lambda_1|^t$ for $\delta > 0$?

don't know γ

$$\begin{aligned} |\lambda_2|^t &= (1-\gamma)^t \cdot |\lambda_1|^t \\ &= (1-\gamma)^{1/\gamma \cdot \gamma t} \cdot |\lambda_1|^t \\ &\le e^{-\gamma t} \cdot |\lambda_1|^t \end{aligned}$$

So it suffices to set $\gamma t = \ln(1/\delta)$. Or $t = \frac{\ln(1/\delta)}{\gamma}$. $\approx \frac{\ln(1/\delta)}{\ln\left(\frac{1}{1-\gamma}\right)}$

$\vec{v}_1$: top eigenvector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$. $\lambda_1, \lambda_2, \ldots \lambda_n$: eigenvalues of $\mathbf{A}$, $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$: eigengap controlling convergence rate

Power Method Convergence Rate

$\delta|\lambda_1|^t$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$

Write $|\lambda_2| = (1-\gamma)|\lambda_1|$ for 'gap' $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$.

How many iterations t does it take to have $|\lambda_2|^t \leq \delta \cdot |\lambda_1|^t$ for $\delta > 0$?

$$\begin{aligned}|\lambda_2|^t &= (1-\gamma)^t \cdot |\lambda_1|^t \\ &= (1-\gamma)^{1/\gamma \cdot \gamma t} \cdot |\lambda_1|^t \\ &\leq e^{-\gamma t} \cdot |\lambda_1|^t\end{aligned}$$

So it suffices to set $\gamma t = \ln(1/\delta)$. Or $t = \frac{\ln(1/\delta)}{\gamma}$.

How small must we set δ to ensure that $c_1\lambda_1^t$ dominates all other components and so $\vec{z}^{(t)}$ is very close to $\vec{v}_1$?

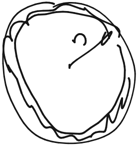
$\vec{v}_1$: top eigenvector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$. $\lambda_1, \lambda_2, \ldots \lambda_n$: eigenvalues of $\mathbf{A}$, $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$: eigengap controlling convergence rate

Claim: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = \underbrace{c_1}_{N(0,1)}\vec{v}_1 + \underbrace{c_2}_{N(0,1)}\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, for all i:

$$\underbrace{O(1/d^2)}_{\text{anti-concentration bound}} \leq |c_i| \leq \underbrace{O(\log d)}_{\text{concentration bound}} \qquad \mathbb{E}\, c_i = 0$$

Corollary:

$$\max_j \left| \frac{c_j^{\,\log d}}{c_1{}_{\,d^2}} \right| \leq O(d^2 \log d).$$



$\mathbf{A} \in \mathbb{R}^{d\times d}$: input matrix with eigendecomposition $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$. $\vec{v}_1$: top eigenvector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

Random Initialization

Claim 1: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, $\max_j \left|\frac{c_j}{c_1}\right| \leq O(d^2 \log d)$.

Claim 2: For gap $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$, and $t = \frac{\ln(1/\delta)}{\gamma}$, $\left|\frac{\lambda_i^t}{\lambda_1^t}\right| \leq \delta$ for all i.

$\mathbf{A} \in \mathbb{R}^{d \times d}$: input with eigenvalues $\lambda_1 \ldots, \lambda_d$ and eigenvectors $\vec{v}_1, \ldots, \vec{v}_d$. $\vec{z}^{(i)}$: iterate at step i. $c_1, \ldots, c_d$: coefficients of $\vec{z}^{(0)}$ in the eigenvector basis.

Random Initialization

Claim 1: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, $\max_j \left|\frac{c_j}{c_1}\right| \leq O(d^2 \log d)$.

Claim 2: For gap $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$, and $t = \frac{\ln(1/\delta)}{\gamma}$, $\left|\frac{\lambda_i^t}{\lambda_1^t}\right| \leq \delta$ for all i.

$$\vec{z}^{(t)} := \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d\|_2}$$

$\mathbf{A} \in \mathbb{R}^{d\times d}$: input with eigenvalues $\lambda_1 \ldots, \lambda_d$ and eigenvectors $\vec{v}_1, \ldots, \vec{v}_d$. $\vec{z}^{(i)}$: iterate at step i. $c_1, \ldots, c_d$: coefficients of $\vec{z}^{(0)}$ in the eigenvector basis.

Random Initialization

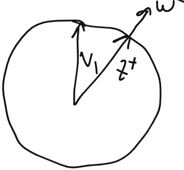
Claim 1: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, $\max_j \left|\frac{c_j}{c_1}\right| \leq O(d^2 \log d)$.

Claim 2: For gap $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$, and $t = \frac{\ln(1/\delta)}{\gamma}$, $\left|\frac{\lambda_i^t}{\lambda_1^t}\right| \leq \delta$ for all i.

$$\vec{z}^{(t)} := \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d\|_2} \implies \left\| \frac{c_1\lambda_1^t v_1 + \cdots c_d\lambda_d^t v_d}{\| \cdots \|} - v_1 \right\|_2$$

$$\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \left\| \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1\|_2} - \vec{v}_1 \right\|_2$$

w w



$\mathbf{A} \in \mathbb{R}^{d\times d}$: input with eigenvalues $\lambda_1 \ldots, \lambda_d$ and eigenvectors $\vec{v}_1, \ldots, \vec{v}_d$. $\vec{z}^{(i)}$: iterate at step i. $c_1, \ldots, c_d$: coefficients of $\vec{z}^{(0)}$ in the eigenvector basis.

Claim 1: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, $\max_j \left|\frac{c_j}{c_1}\right| \leq O(d^2 \log d)$.

Claim 2: For gap $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$, and $t = \frac{\ln(1/\delta)}{\gamma}$, $\left|\frac{\lambda_i^t}{\lambda_1^t}\right| \leq \delta$ for all i.

$$\vec{z}^{(t)} := \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d\|_2} \implies$$

v_1

$$\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \left\|\frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1\|_2} - \vec{v}_1\right\|_2$$

$$= \left\|\frac{c_2\lambda_2^t}{c_1\lambda_1^t}\vec{v}_2 + \ldots + \frac{c_d\lambda_d^t}{\lambda_1^t}\vec{v}_d\right\|_2$$

$\mathbf{A} \in \mathbb{R}^{d\times d}$: input with eigenvalues $\lambda_1 \ldots, \lambda_d$ and eigenvectors $\vec{v}_1, \ldots, \vec{v}_d$. $\vec{z}^{(i)}$: iterate at step i. $c_1, \ldots, c_d$: coefficients of $\vec{z}^{(0)}$ in the eigenvector basis.

Random Initialization

Claim 1: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, $\max_j \left|\frac{c_j}{c_1}\right| \leq O(d^2 \log d)$.

Claim 2: For gap $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$, and $t = \frac{\ln(1/\delta)}{\gamma}$, $\left|\frac{\lambda_i^t}{\lambda_1^t}\right| \leq \delta$ for all i.

$$\vec{z}^{(t)} := \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d\|_2} \implies$$

$$\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \left\|\frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1\|_2} - \vec{v}_1\right\|_2$$

$$= \left\|\frac{c_2\lambda_2^t}{c_1\lambda_1^t}\vec{v}_2 + \ldots + \frac{c_d\lambda_d^t}{\lambda_1^t}\vec{v}_d\right\|_2 = \left|\frac{c_2\lambda_2^t}{c_1\lambda_1^t}\right| + \ldots + \left|\frac{c_d\lambda_d^t}{\lambda_1^t}\right|$$

$\mathbf{A} \in \mathbb{R}^{d\times d}$: input with eigenvalues $\lambda_1 \ldots, \lambda_d$ and eigenvectors $\vec{v}_1, \ldots, \vec{v}_d$. $\vec{z}^{(i)}$: iterate at step i. $c_1, \ldots, c_d$: coefficients of $\vec{z}^{(0)}$ in the eigenvector basis.

Claim 1: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, $\max_j \left|\frac{c_j}{c_1}\right| \leq O(d^2 \log d)$.

Claim 2: For gap $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$, and $t = \frac{\ln(1/\delta)}{\gamma}$, $\left|\frac{\lambda_i^t}{\lambda_1^t}\right| \leq \delta$ for all i.

$$\vec{z}^{(t)} := \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d\|_2} \implies$$

$$\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \left\| \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\underline{\|c_1\lambda_1^t\vec{v}_1\|_2}} - \vec{v}_1 \right\|_2$$

$$= \left\| \frac{c_2\lambda_2^t}{c_1\lambda_1^t}\vec{v}_2 + \ldots + \frac{c_d\lambda_d^t}{c_1\lambda_1^t}\vec{v}_d \right\|_2 \leq \left|\underbrace{\frac{c_2\lambda_2^t}{c_1\lambda_1^t}}_{O(d^2\log d)\ \cdot\ \leq\delta}\right| + \ldots + \left|\frac{c_d\lambda_d^t}{c_1\lambda_1^t}\right| \leq \delta \cdot O(d^2 \log d) \cdot d.$$

$$= \sqrt{\sum \left|\frac{c_i\lambda_i^t}{c_1\lambda_1^t}\right|^2} \qquad \delta \cdot O(d^3 \log d)$$

$\mathbf{A} \in \mathbb{R}^{d\times d}$: input with eigenvalues $\lambda_1 \ldots, \lambda_d$ and eigenvectors $\vec{v}_1, \ldots, \vec{v}_d$. $\vec{z}^{(i)}$: iterate at step i. $c_1, \ldots, c_d$: coefficients of $\vec{z}^{(0)}$ in the eigenvector basis.

Random Initialization

Claim 1: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, $\max_j \left|\frac{c_j}{c_1}\right| \leq O(d^2 \log d)$.

Claim 2: For gap $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$, and $t = \frac{\ln(1/\delta)}{\gamma}$, $\left|\frac{\lambda_i^t}{\lambda_1^t}\right| \leq \delta$ for all i.

$$\vec{z}^{(t)} := \frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d\|_2} \implies$$

$$\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \left\|\frac{c_1\lambda_1^t\vec{v}_1 + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1\|_2} - \vec{v}_1\right\|_2$$

$$= \left\|\frac{c_2\lambda_2^t}{c_1\lambda_1^t}\vec{v}_2 + \ldots + \frac{c_d\lambda_d^t}{\lambda_1^t}\vec{v}_d\right\|_2 = \left|\frac{c_2\lambda_2^t}{c_1\lambda_1^t}\right| + \ldots + \left|\frac{c_d\lambda_d^t}{\lambda_1^t}\right| \leq \underline{\delta \cdot O(d^2 \log d) \cdot d}.$$

Setting $\underline{\delta = O\left(\frac{\epsilon}{d^3 \log d}\right)}$ gives $\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \epsilon$.

$$\frac{\log(1/\delta)}{\gamma} = \frac{\log\left(\frac{d^3 \log d}{\epsilon}\right)}{\gamma} = O\left(\frac{\log(1/\epsilon)}{\gamma}\right)$$

$A \in \mathbb{R}^{d \times d}$: input with eigenvalues $\lambda_1 \ldots, \lambda_d$ and eigenvectors $\vec{v}_1, \ldots, \vec{v}_d$. $\vec{z}^{(i)}$: iterate at step i. $c_1, \ldots, c_d$: coefficients of $\vec{z}^{(0)}$ in the eigenvector basis.

Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$ *be the relative gap between the first and second eigenvalues. If Power Method is initialized with a random Gaussian vector* $\vec{v}^{(0)}$ *then, with high probability, after* $t = O\left(\frac{\ln(d/\epsilon)}{\gamma}\right)$ *steps:*

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

Power Method Theorem

Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{|\lambda_1| - |\lambda_2|}{|\lambda_1|}$ be the relative gap between the first and second eigenvalues. If Power Method is initialized with a random Gaussian vector $\vec{v}^{(0)}$ then, with high probability, after $t = O\left(\frac{\ln(d/\epsilon)}{\gamma}\right)$ steps:

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

$X^T(Xz)$

Total runtime: $O(t)$ matrix-vector multiplications. If $A = X^T X$:

$$O\left(\text{nnz}(X) \cdot \frac{\ln(d/\epsilon)}{\gamma} \cdot\right) = O\left(nd \cdot \frac{\ln(d/\epsilon)}{\gamma}\right).$$

$O(nd)$

$O(nd^2)$ for full eigen decomp.

Power Method Theorem

Theorem (Basic Power Method Convergence)
Let $\gamma = \frac{|\lambda_1|-|\lambda_2|}{|\lambda_1|}$ *be the relative gap between the first and second eigenvalues. If Power Method is initialized with a random Gaussian vector* $\vec{v}^{(0)}$ *then, with high probability, after* $t = O\left(\frac{\ln(d/\epsilon)}{\gamma}\right)$ *steps:*

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

Total runtime: $O(t)$ matrix-vector multiplications. If $\mathbf{A} = \mathbf{X}^T\mathbf{X}$:

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

How is ϵ dependence? very good

How is γ dependence? not great

Krylov subspace methods (Lanczos method, Arnoldi method.)

- How `svds`/`eigs` are actually implemented. Only need $t = O\left(\frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

Krylov subspace methods (Lanczos method, Arnoldi method.)

- How `svds`/`eigs` are actually implemented. Only need $t = O\left(\frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

Main Idea: Need to separate λ_1 from λ_i for $i \geq 2$.

Krylov Subspace Methods

Krylov subspace methods (Lanczos method, Arnoldi method.)

- How `svds`/`eigs` are actually implemented. Only need $t = O\left(\frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

Main Idea: Need to separate λ_1 from λ_i for $i \geq 2$.

- Power method: power up to λ_1^t and λ_i^t.

Krylov subspace methods (Lanczos method, Arnoldi method.)

- How `svds`/`eigs` are actually implemented. Only need $t = O\left(\frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

Main Idea: Need to separate λ_1 from λ_i for $i \geq 2$.

- Power method: power up to λ_1^t and λ_i^t. ~ $f(x) = x^t$
- Krylov methods: apply a better degree t polynomial $T_t(\cdot)$ to the eigenvalues to separate $T_t(\lambda_1)$ from $T_t(\lambda_i)$.

Krylov subspace methods (Lanczos method, Arnoldi method.)

- How `svds`/`eigs` are actually implemented. Only need $t = O\left(\frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

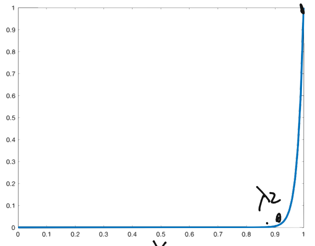
Main Idea: Need to separate λ_1 from λ_i for $i \geq 2$.

- Power method: power up to λ_1^t and λ_i^t.
- Krylov methods: apply a better degree t polynomial $T_t(\cdot)$ to the eigenvalues to separate $T_t(\lambda_1)$ from $T_t(\lambda_i)$.
- Still requires just t matrix vector multiplies. Why?

$$c_1 x + c_2 x^2 + \ldots c_t x^t$$

$$c_1 A v^0 + c_2 A^2 v_0 + \ldots c_t A^t v_0$$

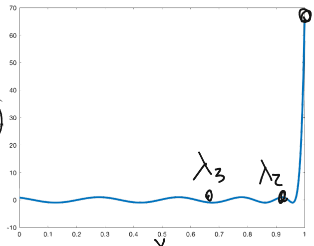
transformation of eigenvalues x^t



eigenvalues

vs.

$T_t(x)$



Optimal ‘jump’ polynomial in general is given by a degree t Chebyshev polynomial. Krylov methods find a polynomial tuned to the input matrix that does at least as well.

Generalizations to Larger k

- Block Power Method (a.k.a. Simultaneous Iteration, Subspace Iteration, or Orthogonal Iteration)
- Block Krylov methods

$$\textbf{Runtime: } O\left(ndk \cdot \frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$$

to accurately compute the top k singular vectors.

Generalizations to Larger k

- Block Power Method (a.k.a. Simultaneous Iteration, Subspace Iteration, or Orthogonal Iteration)

$$A\overset{d\times k}{Z} \to Z = \mathrm{orth}(AZ)$$

- Block Krylov methods

$$\text{Runtime: } O\left(ndk \cdot \frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$$

to accurately compute the top k singular vectors.

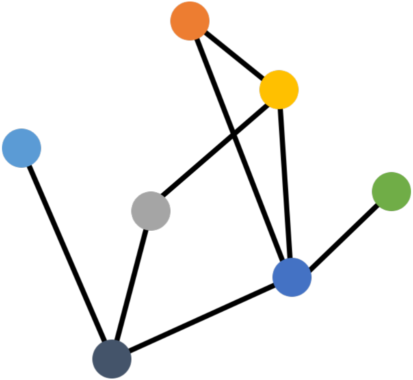
$$\text{‘Gapless’ Runtime: } O\left(ndk \cdot \frac{\ln(d/\epsilon)}{\sqrt{\epsilon}}\right)$$

if you just want a set of vectors that gives an ϵ-optimal low-rank approximation when you project onto them.

Connection Between Random Walks, Eigenvectors, and Power Method

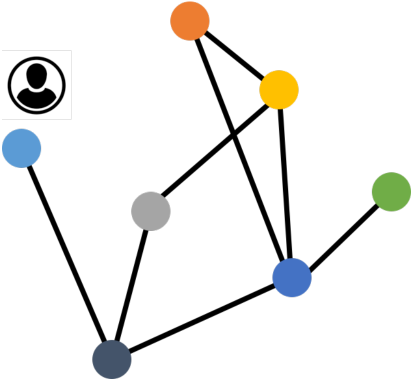
Connection to Random Walks

Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.



Connection to Random Walks

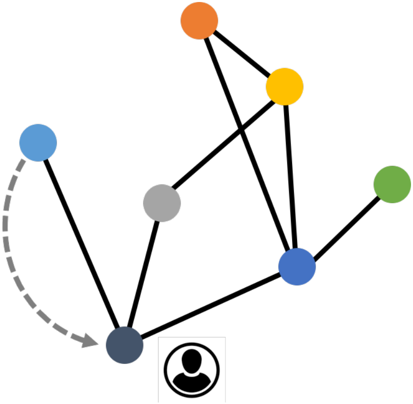
Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.



At each step, move to a random vertex, chosen uniformly at random from the neighbors of the current vertex.

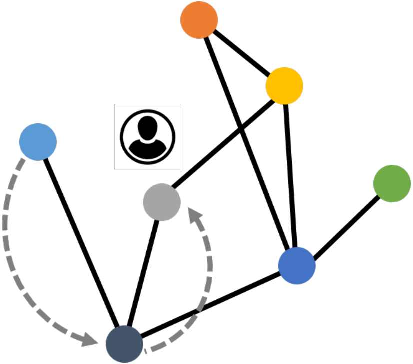
Connection to Random Walks

Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.



Connection to Random Walks

Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.



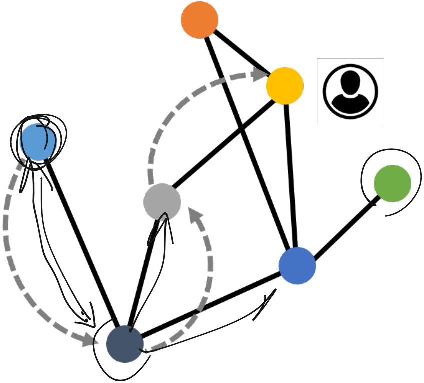
Connection to Random Walks

Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.

$$P^{(0)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$P^{(1)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 1/3 \\ 0 \\ 1/3 \\ 1/3 \\ 1 \\ 0 \end{bmatrix}$$



Connection to Random Walks

Let $\vec{p}^{(t)} \in \mathbb{R}^n$ have i^{th} entry $\vec{p}_i^{(t)} = \Pr$(walk at node i at step t).

Connection to Random Walks

Let $\vec{p}^{(t)} \in \mathbb{R}^n$ have i^{th} entry $\vec{p}_i^{(t)} = \Pr(\text{walk at node i at step t})$.

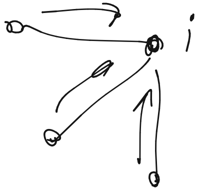
- **Initialize:** $\vec{p}^{(0)} = [1, 0, 0, \ldots, 0]$.

Connection to Random Walks

Let $\vec{p}^{(t)} \in \mathbb{R}^n$ have i^{th} entry $\vec{p}_i^{(t)} = \Pr(\text{walk at node i at step t})$.

- **Initialize:** $\vec{p}^{(0)} = [1, 0, 0, \ldots, 0]$.
- **Update:**

$$\Pr(\text{walk at i at step t}) = \sum_{j \in neigh(i)} \Pr(\text{walk at j at step t-1}) \cdot \frac{1}{degree(j)}$$



Connection to Random Walks

Let $\vec{p}^{(t)} \in \mathbb{R}^n$ have i^{th} entry $\vec{p}_i^{(t)} = \Pr(\text{walk at node i at step t})$.

- **Initialize:** $\vec{p}^{(0)} = [1, 0, 0, \ldots, 0]$.
- **Update:**

$$\Pr(\text{walk at i at step t}) = \sum_{j \in neigh(i)} \Pr(\text{walk at j at step t-1}) \cdot \frac{1}{degree(j)}$$

$[0\ \frac{1}{d_2}\ 0\ 0\ \frac{1}{d_5}\ 0\ 0\ \frac{1}{d_7}]$

$$= \vec{z}^T \vec{p}^{(t-1)}$$

where $\vec{z}(j) = \frac{1}{degree(j)}$ for all $j \in neigh(i)$, $\vec{z}(j) = 0$ for all $j \notin neigh(i)$.

Connection to Random Walks

Let $\vec{p}^{(t)} \in \mathbb{R}^n$ have i^{th} entry $\vec{p}_i^{(t)} = \Pr(\text{walk at node i at step t})$.

- **Initialize:** $\vec{p}^{(0)} = [1, 0, 0, \ldots, 0]$.
- **Update:**

$$\Pr(\text{walk at i at step t}) = \sum_{j \in neigh(i)} \Pr(\text{walk at j at step t-1}) \cdot \frac{1}{degree(j)}$$
$$= \vec{z}^T \vec{p}^{(t-1)}$$

  where $\vec{z}(j) = \frac{1}{degree(j)}$ for all $j \in neigh(i)$, $\vec{z}(j) = 0$ for all $j \notin neigh(i)$.

- $\vec{z}$ is the i^{th} row of the right normalized adjacency matrix $\mathbf{AD}^{-1}$.

$$\overset{A}{\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ & & & & & \end{bmatrix}} \begin{bmatrix} 1/d_1 & & & \\ & 1/d_2 & & \\ & & \ddots & \\ & & & 1/d_n \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{d_2} & 0 & \frac{1}{d_4} & 0 & \frac{1}{d_6} \\ & & & & & \end{bmatrix}$$

Connection to Random Walks

Let $\vec{p}^{(t)} \in \mathbb{R}^n$ have i^{th} entry $\vec{p}_i^{(t)} = \Pr(\text{walk at node i at step t})$.

- **Initialize:** $\vec{p}^{(0)} = [1, 0, 0, \ldots, 0]$.
- **Update:**

$$\begin{aligned}\Pr(\text{walk at i at step t}) &= \sum_{j \in neigh(i)} \Pr(\text{walk at j at step t-1}) \cdot \frac{1}{degree(j)} \\ &= \vec{z}^T \vec{p}^{(t-1)}\end{aligned}$$

  where $\vec{z}(j) = \frac{1}{degree(j)}$ for all $j \in neigh(i)$, $\vec{z}(j) = 0$ for all $j \notin neigh(i)$.

- $\vec{z}$ is the i^{th} row of the right normalized adjacency matrix $\mathbf{AD}^{-1}$.
- $\vec{p}^{(t)} = \mathbf{AD}^{-1}\vec{p}^{(t-1)}$

Connection to Random Walks

Let $\vec{p}^{(t)} \in \mathbb{R}^n$ have i^{th} entry $\vec{p}_i^{(t)} = \Pr(\text{walk at node i at step t})$.

- **Initialize:** $\vec{p}^{(0)} = [1, 0, 0, \ldots, 0]$.
- **Update:**

$$\begin{aligned}\Pr(\text{walk at i at step t}) &= \sum_{j \in neigh(i)} \Pr(\text{walk at j at step t-1}) \cdot \frac{1}{degree(j)} \\ &= \vec{z}^T \vec{p}^{(t-1)}\end{aligned}$$

  where $\vec{z}(j) = \frac{1}{degree(j)}$ for all $j \in neigh(i)$, $\vec{z}(j) = 0$ for all $j \notin neigh(i)$.

- $\vec{z}$ is the i^{th} row of the right normalized adjacency matrix $\mathbf{AD}^{-1}$.
- $\vec{p}^{(t)} = \mathbf{AD}^{-1}\vec{p}^{(t-1)} = \underbrace{\mathbf{AD}^{-1}\mathbf{AD}^{-1}\ldots\mathbf{AD}^{-1}}_{t \text{ times}}\vec{p}^{(0)}$

Random Walking as Power Method

Claim: After t steps, the probability that a random walk is at node i is given by the i^{th} entry of

$$\vec{p}^{(t)} = \underbrace{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\ldots\mathbf{A}\mathbf{D}^{-1}}_{t \text{ times}}\vec{p}^{(0)}.$$

Random Walking as Power Method

Claim: After t steps, the probability that a random walk is at node i is given by the i^{th} entry of

$$D^{-1/2}\vec{p}^{(t)} = D^{-1/2}\underbrace{AD^{-1}AD^{-1}\dots AD^{-1}}_{t\text{ times}}\vec{p}^{(0)}.$$

$$D^{-1/2}\vec{p}^{(t)} = \underbrace{(D^{-1/2}AD^{-1/2})(D^{-1/2}AD^{-1/2})\dots(D^{-1/2}AD^{-1/2})}_{t\text{ times}}(D^{-1/2}\vec{p}^{(0)}).$$

(t step power method

Random Walking as Power Method

Claim: After t steps, the probability that a random walk is at node i is given by the i^{th} entry of

$$\vec{p}^{(t)} = \underbrace{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\ldots\mathbf{A}\mathbf{D}^{-1}}_{t \text{ times}}\vec{p}^{(0)}.$$

$$\mathbf{D}^{-1/2}\vec{p}^{(t)} = \underbrace{(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\ldots(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})}_{t \text{ times}}(\mathbf{D}^{-1/2}\vec{p}^{(0)}).$$

- $\mathbf{D}^{-1/2}\vec{p}^{(t)}$ is exactly what would obtained by applying $t/2$ iterations of power method to $\mathbf{D}^{-1/2}\vec{p}^{(0)}$!

Random Walking as Power Method

Claim: After t steps, the probability that a random walk is at node i is given by the i^{th} entry of

$$\vec{p}^{(t)} = \underbrace{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\ldots\mathbf{A}\mathbf{D}^{-1}}_{t \text{ times}}\vec{p}^{(0)}.$$

$$\mathbf{D}^{-1/2}\vec{p}^{(t)} = \underbrace{(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\ldots(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})}_{t \text{ times}}(\mathbf{D}^{-1/2}\vec{p}^{(0)}).$$

- $\mathbf{D}^{-1/2}\vec{p}^{(t)}$ is exactly what would obtained by applying $t/2$ iterations of power method to $\mathbf{D}^{-1/2}\vec{p}^{(0)}$!
- Will converge to the top eigenvector of the normalized adjacency matrix $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Stationary distribution.

Random Walking as Power Method

Claim: After t steps, the probability that a random walk is at node i is given by the i^{th} entry of

$$\vec{p}^{(t)} = \underbrace{\mathbf{A}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\ldots\mathbf{A}\mathbf{D}^{-1}}_{t \text{ times}}\vec{p}^{(0)}.$$

$$\mathbf{D}^{-1/2}\vec{p}^{(t)} = \underbrace{(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\ldots(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})}_{t \text{ times}}(\mathbf{D}^{-1/2}\vec{p}^{(0)}).$$

- $\mathbf{D}^{-1/2}\vec{p}^{(t)}$ is exactly what would obtained by applying $t/2$ iterations of power method to $\mathbf{D}^{-1/2}\vec{p}^{(0)}$!
- Will converge to the top eigenvector of the normalized adjacency matrix $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Stationary distribution.
- Like the power method, the time a random walk takes to converge to its stationary distribution (mixing time) is dependent on the gap between the top two eigenvalues of $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. The spectral gap.

Continuous Optimization and Gradient Descent

Discrete vs. Continuous Optimization

Discrete (Combinatorial) Optimization: (traditional CS algorithms)

- Graph Problems: min-cut, max flow, shortest path, matchings, maximum independent set, traveling salesman problem
- Problems with discrete constraints or outputs: bin-packing, scheduling, sequence alignment, submodular maximization
- Generally searching over a finite but exponentially large set of possible solutions. Many of these problems are NP-Hard.

Continuous Optimization: (maybe seen in ML/advanced algorithms)

- Unconstrained convex and non-convex optimization.
- Linear programming, quadratic programming, semidefinite programming

Continuous Optimization Examples