COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2023.

Lecture 21

Logistics

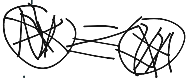
See Piazza post about upcoming schedule information.

- No quiz due this week.
- Problem Set 4 is due 12/1.
- No class Thursday.
- Office hours next Monday at 10am in CS234.
- No class next Tuesday
- Class over Zoom next Thursday 11/30 at 10am. Office hours over Zoom at 9am. See Piazza for Zoom link.
- Second Linear Algebra Review Session on Monday 12/4 at 3pm in CS140.

Summary

Last Few Classes: Spectral Graph Partitioning

- Focus on separating graphs with small but relatively balanced cuts.
- Connection to second smallest eigenvector of graph Laplacian.
- Provable guarantees for stochastic block model.
- Expectation analysis in class. See slides for full analysis.



This Class: Computing the SVD/eigendecomposition.

- Efficient algorithms for SVD/eigendecomposition.
- Iterative methods: power method, Krylov subspace methods.
- High level: a glimpse into fast methods for linear algebraic computation, which are workhorses behind data science.

Quiz Question

$\sum z(i) = \langle z, 1 \rangle \simeq 0$

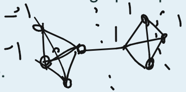
Consider solving the optimization problem: $\min_{\{z \in \{-1,1\}^n :\ \text{not all entries of } z \text{ are equal}\}} z^T L z$.

$\begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

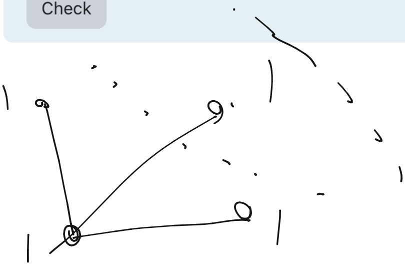
What is this optimization problem commonly known as?

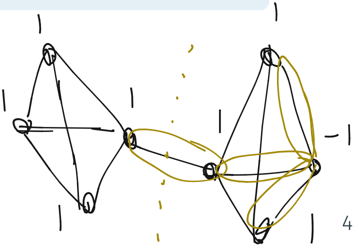
- a. Computing the lowest eigenvalue of the graph Laplacian.
- b. Computing the second lowest eigenvalue of the graph Laplacian.
- c. Computing the minimum cut.
- d. Computing the smallest node degree.
- e. Computing the maximum eigenvalue of the graph Laplacian.

$z^T L z = 4\,\mathrm{cut}(z)$



Check





Efficient Eigendecomposition and SVD

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 large datasets?

Computing the SVD

Basic Algorithm: To compute the SVD of full-rank $\mathbf{X} \in \mathbb{R}^{n \times d}$, $\mathbf{X} = \underbrace{\mathbf{U}}_{n\times d}\overset{d\times d}{\mathbf{\Sigma}}\underbrace{\mathbf{V}^T}_{d\times d}$:

$\text{rank}(X) = d$

$$n\begin{bmatrix} \overset{d}{X} \end{bmatrix}$$

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

$X = U\Sigma V^T V$

$= U\Sigma$

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

$$\begin{bmatrix} X^T \end{bmatrix}\begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} X^T X \end{bmatrix}$$

$$U = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_d \\ | & | & & | \end{bmatrix} \qquad U\Sigma = \begin{bmatrix} | & & | \\ u_1\sigma_1 & \cdots & u_d\sigma_d \\ | & & | \end{bmatrix}$$

Computing the SVD

Basic Algorithm: To compute the SVD of full-rank $\mathbf{X} \in \mathbb{R}^{n \times d}$, $\mathbf{X} = \mathbf{U\Sigma V}^T$:

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

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

Computing the SVD

Basic Algorithm: To compute the SVD of full-rank $\mathbf{X} \in \mathbb{R}^{n\times d}$, $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$:

$X^T = V\Sigma U^T$

$n < d$ start by compute XX^T let U be eigenvectors of XX^T $O(n^2 d)$

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

$L = XV = U\Sigma V^T V = U\Sigma$

- Compute $\mathbf{L} = \mathbf{X}\mathbf{V}$ – $O(nd^2)$ runtime. Note that $\mathbf{L} = \mathbf{U}\boldsymbol{\Sigma}$.
- Set $\sigma_i = \|\mathbf{L}_i\|_2$ and $\mathbf{U}_i = \mathbf{L}_i / \|\mathbf{L}_i\|_2$. – $O(nd)$ 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 a relatively easy task for them – but no one else.

Faster Algorithms

$\begin{bmatrix} v_1 & v_k & v_d \end{bmatrix}$

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 of a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ for $k \ll d$.

- Suffices to compute $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ and then compute $\mathbf{U}_k \boldsymbol{\Sigma}_k = \mathbf{X}\mathbf{V}_k$.

  $XV_k = U_k \Sigma_k$

- Use an *iterative algorithm* to compute an *approximation* to the top k singular vectors $\mathbf{V}_k$ (the top k eigenvectors of $\mathbf{X}^T\mathbf{X}$.)
- Runtime will be roughly $O(ndk)$ instead of $O(nd^2)$.

  $\ll$

$\cancel{O(ndk)}$

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 of a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ for $k \ll d$.

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

slow   fast iterative

Sparse (iterative) vs. Direct Method. `svd` vs. `svds`.

eig   eigs

svds

Power Method: The most fundamental iterative method for approximate SVD/eigendecomposition. Applies to computing $k = 1$ eigenvectors, but can be generalized to larger k.

$X^T X$

Goal: Given symmetric $\mathbf{A} \in \mathbb{R}^{d \times d}$, with eigendecomposition $\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$, find $\vec{z} \approx \vec{v}_1$. I.e., the top eigenvector of $\mathbf{A}$.

Power Method: The most fundamental iterative method for approximate SVD/eigendecomposition. Applies to computing $k = 1$ eigenvectors, but can be generalized to larger k.

Goal: Given symmetric $\mathbf{A} \in \mathbb{R}^{d \times d}$, with eigendecomposition $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, find $\vec{z} \approx \vec{v}_1$. I.e., the top eigenvector of $\mathbf{A}$.

$\|v_1\|_2 = 1$

- **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 $\vec{z}_t$

$z^{(i-1)} \approx z^{(i)}$

$\leq 10^{-5}$

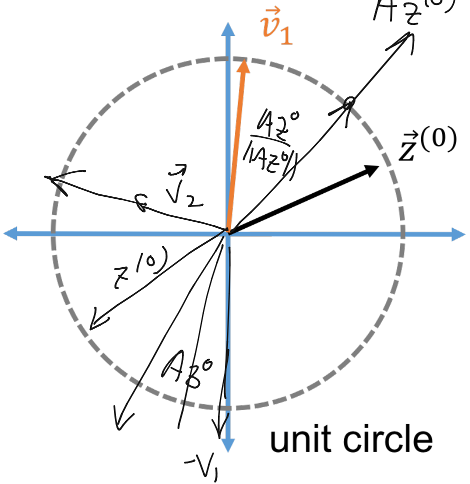
$z^{(i-1)} = v_1$

$z^{(i)} = A z^{(i-1)} = A v_1 = \lambda_1 \cdot v_1$

$z^{(i)} = \frac{z^{(i)}}{\|z^{(i)}\|_2} = \frac{\lambda_1 v_1}{\|\lambda_1 v_1\|} = v_1$

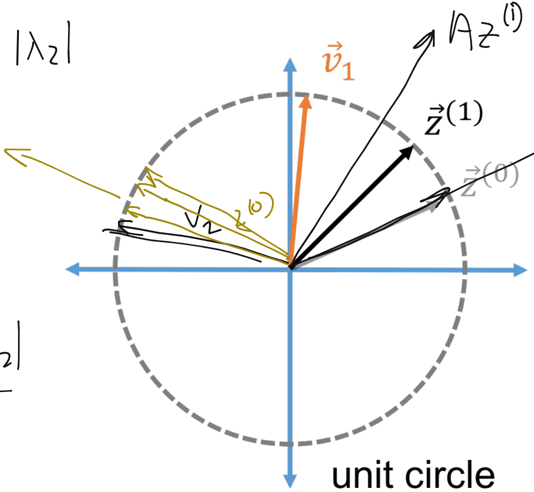
Power Method

$A \in \mathbb{R}^{2\times 2}$



Power Method

$$|\lambda_1| = |\lambda_2|$$

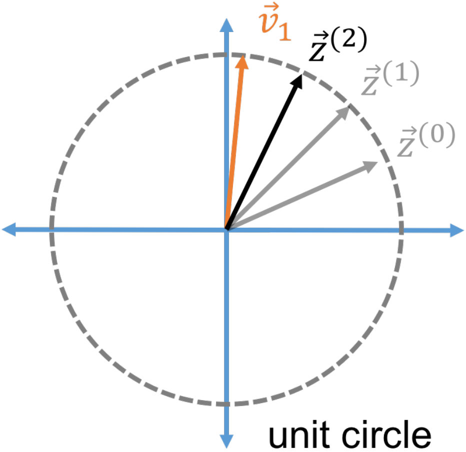


$$\lambda_1 = \lambda_2$$

$$Az^{(0)} \stackrel{?}{=} \lambda_1 \cdot z^{(0)}$$

if $\lambda_1 = \lambda_2$

$$\frac{|\lambda_1 - \lambda_2|}{|\lambda_1|}$$


$\vec{v}_1$
$\vec{z}^{(2)}$
$\vec{z}^{(1)}$
$\vec{z}^{(0)}$
unit circle

Power Method Analysis

Power method:

- **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:

- **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 $\vec{z}_t$.

– in practice to limit roundoff

Theoretically equivalent to:

- 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 Analysis

Write $\vec{z}^{(0)}$ in A's eigenvector basis: d orthogonal eigenvector

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

$\mathbf{A} \in \mathbb{R}^{d\times d}$: 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 $\vec{v}_1$.

Power Method Analysis

Write $\vec{z}^{(0)}$ in $\mathbf{A}$'s eigenvector basis:

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

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

$d \times d \qquad d \times 1$

$$\mathbf{V}^T\vec{z}^{(0)} = \begin{bmatrix} — v_1^T — \\ V^T \vdots \\ v_d^T \end{bmatrix} \begin{bmatrix} z^{(0)} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_d \end{bmatrix}$$

$$\mathbf{\Lambda}\mathbf{V}^T\vec{z}^{(0)} = \begin{bmatrix} \lambda_1 & & \Lambda \\ & \ddots & \\ & & \lambda_d \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_d \end{bmatrix} = \begin{bmatrix} \lambda_1 c_1 \\ \vdots \\ \lambda_d c_d \end{bmatrix}$$

$$\vec{z}^{(1)} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \cdot \vec{z}^{(0)} = \begin{bmatrix} | & V & | \\ v_1 & \cdots & v_d \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 c_1 \\ \vdots \\ \lambda_d c_d \end{bmatrix} = \vec{v}_1 \cdot \lambda_1 c_1 + \vec{v}_2 \lambda_2 c_2 + \ldots$$

$V^T(z^0)$

$= V^T(c_1 v_1 + c_2 v_2 + \ldots c_d v_d)$

$v_1^T(c_1 v_1 + \ldots c_d v_d)$

$= c_1 \overbrace{v_1^T v_1}^{1} + c_2 \overbrace{v_1^T v_2}^{0} + \ldots$

$\mathbf{A} \in \mathbb{R}^{d\times d}$: 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 $\vec{v}_1$.

Power Method Analysis

Claim 1 : Writing $\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$,

$$\vec{z}^{(1)} = c_1 \cdot \lambda_1\vec{v}_1 + c_2 \cdot \lambda_2\vec{v}_2 + \ldots + c_d \cdot \lambda_d\vec{v}_d.$$

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

Power Method Analysis

Claim 1 : Writing $\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$,

$$\vec{z}^{(1)} = c_1 \cdot \lambda_1\vec{v}_1 + c_2 \cdot \lambda_2\vec{v}_2 + \ldots + c_d \cdot \lambda_d\vec{v}_d.$$

$\lambda_1 \cdot c_1\lambda_1 \quad \lambda_2 \cdot c_2\lambda_2$

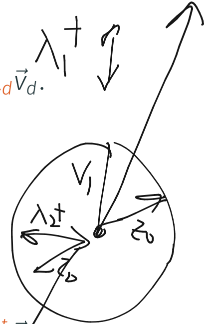
$$\vec{z}^{(2)} = \mathbf{A}\vec{z}^{(1)} = \mathbf{V\Lambda V}^T\vec{z}^{(1)} = c_1\lambda_1^2\vec{v}_1 + c_2\lambda_2^2\vec{v}_2 + \ldots$$

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

Power Method Analysis

Claim 1 : Writing $\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$,

$$\vec{z}^{(1)} = c_1 \cdot \lambda_1\vec{v}_1 + c_2 \cdot \lambda_2\vec{v}_2 + \ldots + c_d \cdot \lambda_d\vec{v}_d.$$



$$\vec{z}^{(2)} = \mathbf{A}\vec{z}^{(1)} = \mathbf{V\Lambda V}^T\vec{z}^{(1)} =$$

relatively very small

Claim 2:

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

very large

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

Power Method Convergence

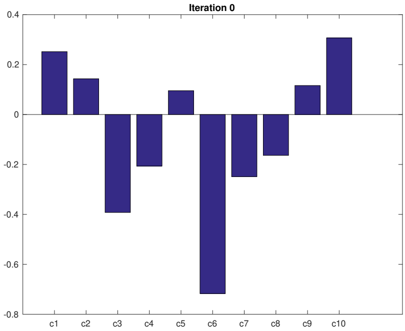
After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

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

Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

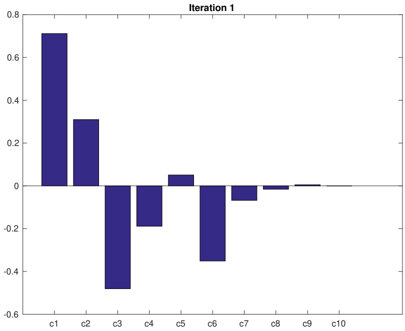
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

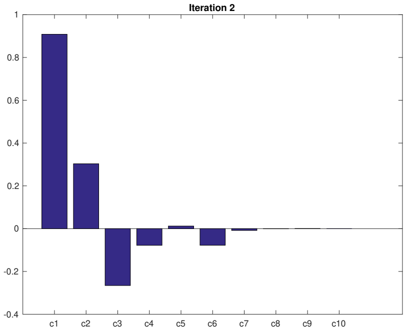
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

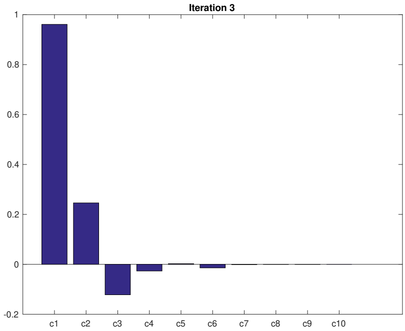
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

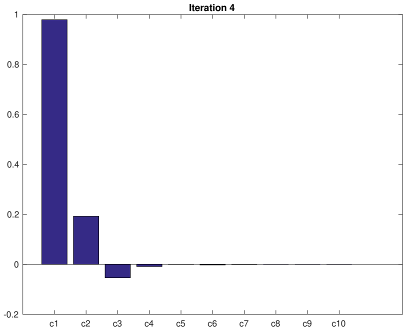
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

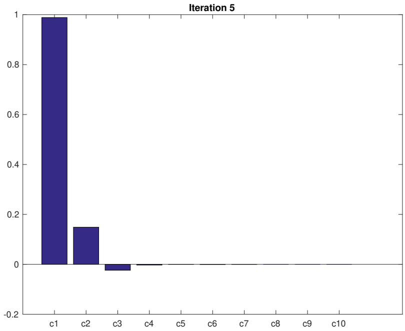
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

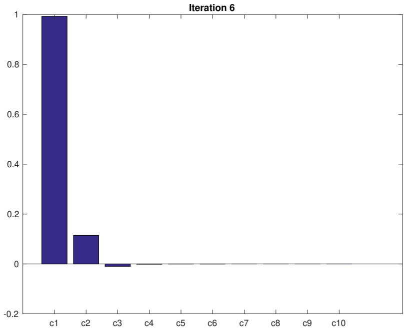
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

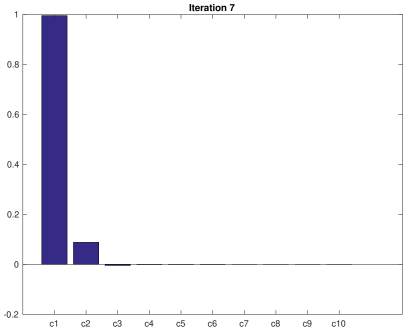
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

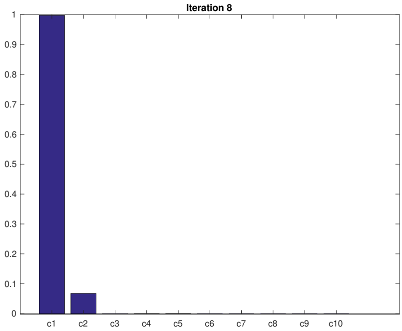
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

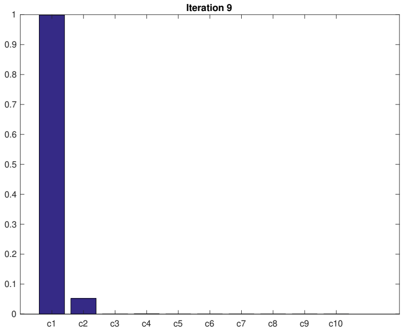
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

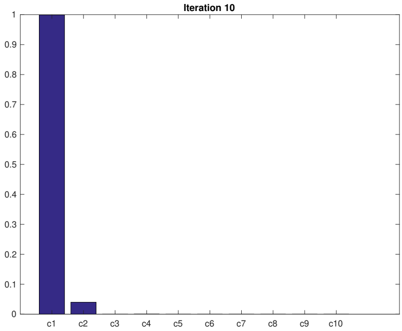
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

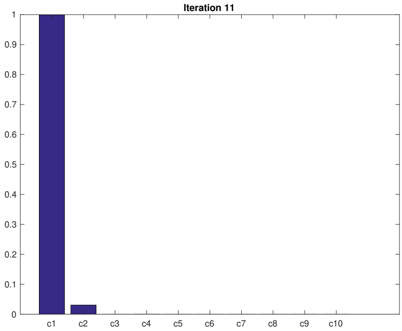
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

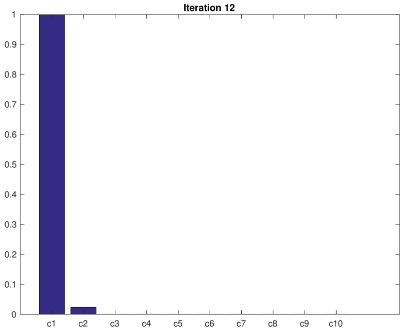
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

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

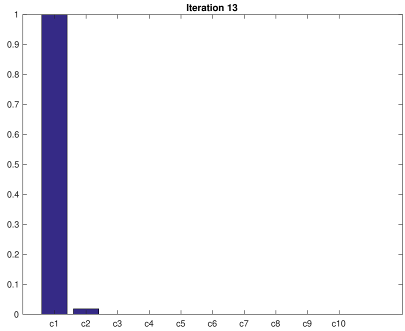


Power Method Convergence

After t iterations, we have 'powered' up the eigenvalues, making the component in the direction of v_1 much larger, relative to the other components.

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

$\lambda_1 \approx \lambda_2$



When will convergence be slow?

Power Method Slow Convergence

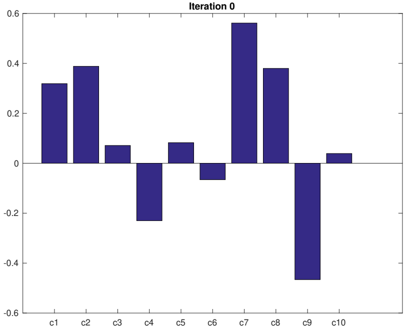
Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

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

Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

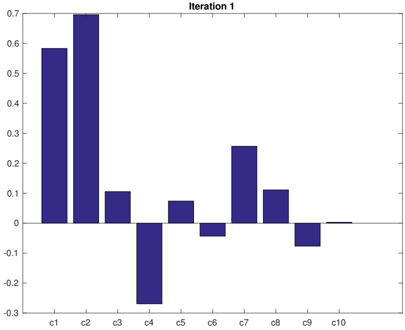
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

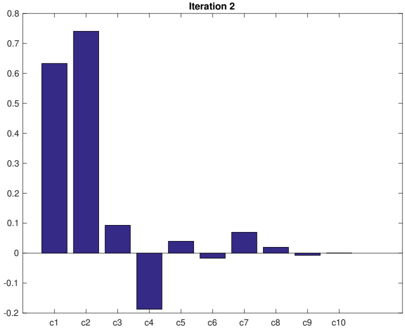
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

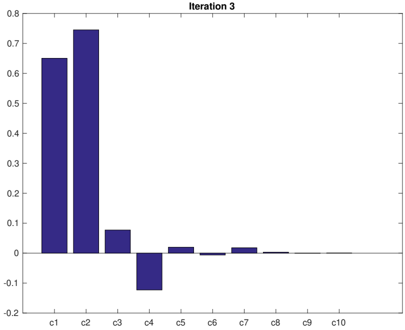
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

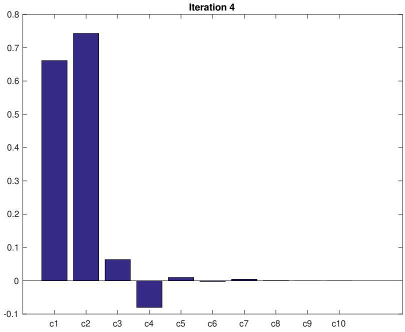
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

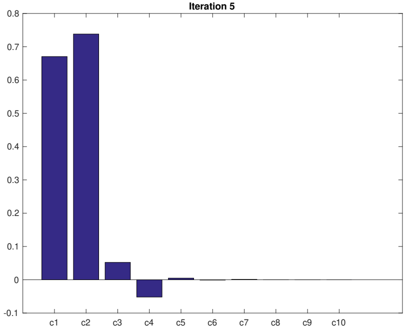
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: $\mathbf{A}$ has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

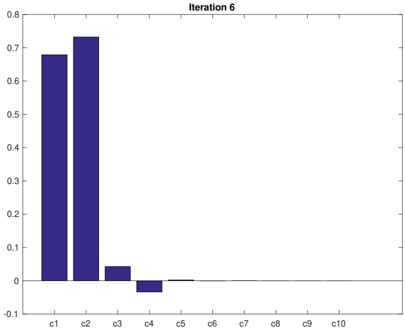
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

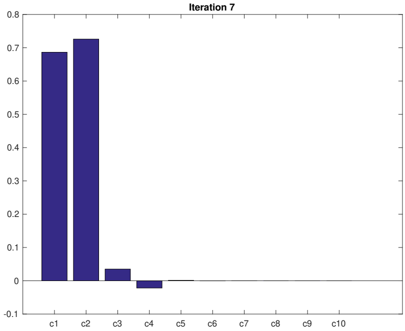
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

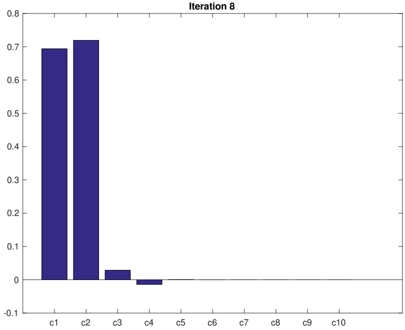
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

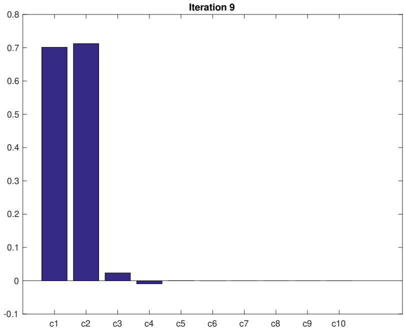
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

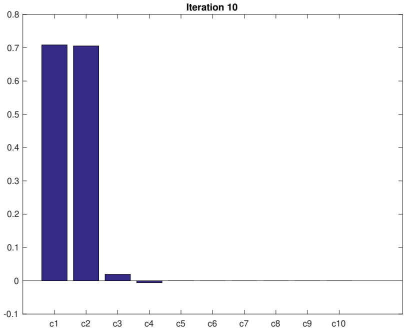
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: **A** has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

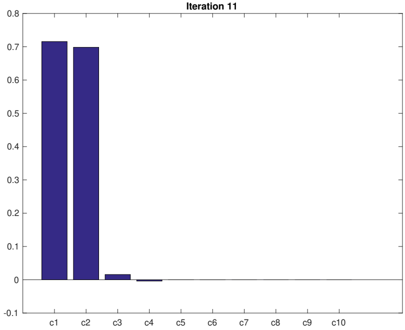
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

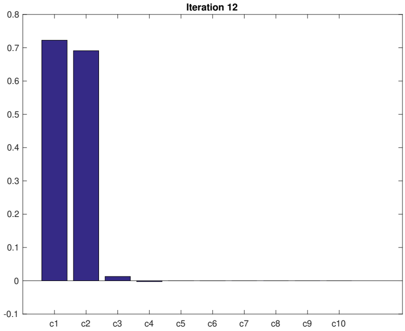
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$



Power Method Slow Convergence

Slow Case: A has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

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



Power Method Slow Convergence

Slow Case: **A** has eigenvalues: $\lambda_1 = 1, \lambda_2 = .99, \lambda_3 = .9, \lambda_4 = .8, \ldots$

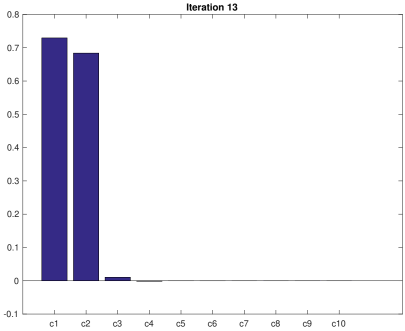
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\lambda_1^t\vec{v}_1 + c_2\lambda_2^t\vec{v}_2 + \ldots + c_d\lambda_d^t\vec{v}_d$$

$A^2 z^{(0)}$

d^3

$A A A z^0$

d^2



Power Method Convergence Rate

$$1 - .99$$

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

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 $\delta > 0$?

$$|\lambda_i|^t \le \delta|\lambda_1|^t$$

$\vec{v}_1$: 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 **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 $\delta > 0$?

$$|\lambda_2|^t = \underbrace{(1-\gamma)^t}_{\leq \delta} \cdot |\lambda_1|^t$$

$\vec{v}_1$: 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 $|\lambda_2|^t \leq \delta \cdot |\lambda_1|^t$ for $\delta > 0$?

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

$t = 1/\gamma \cdot \gamma t$

$\vec{v}_1$: 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 $\delta > 0$?

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

$\vec{v}_1$: 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

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

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 $\delta > 0$?

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

(1/d)

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

$t = \frac{-\ln(d)}{\gamma}$

$\vec{v}_1$: 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 $\delta > 0$?

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

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

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

Random Initialization

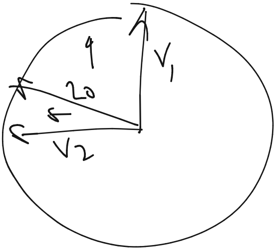
Claim: When $z^{(0)}$ is chosen with random Gaussian entries, writing $z^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d$, with very high probability, for all i:

$N(0,1)$

$$O(1/d^2) \le |c_i| \le O(\log d)$$

Corollary:

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



$\mathbf{A} \in \mathbb{R}^{d\times d}$: 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.

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

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

$\mathbf{A} \in \mathbb{R}^{d \times d}$: 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.

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

$$\|\vec{z}^{(t)} - \vec{v}_1\|_2 \leq \left\|\frac{\underbrace{c_1\lambda_1^t\vec{v}_1}_{= V_1} + \ldots + c_d\lambda_d^t\vec{v}_d}{\|c_1\lambda_1^t\vec{v}_1\|_2} - \vec{v}_1\right\|_2$$

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

$\mathbf{A} \in \mathbb{R}^{d\times d}$: 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.

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.

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

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

$d^2\log d$ δ

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

$A \in \mathbb{R}^{d\times d}$: 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.

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

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

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

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

$A \in \mathbb{R}^{d\times d}$: 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.

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:*

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

[illegible]

$d^2 \cdot t$

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 $\mathbf{A} = \mathbf{X}^T\mathbf{X}$:

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

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?

How is γ dependence?

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 (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

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 $i \geq 2$.

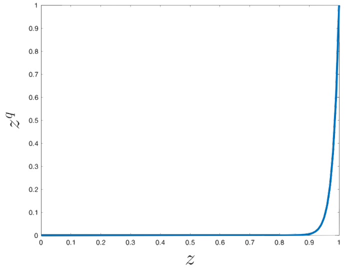
- Power method: power up to λ_1^t and λ_i^t.
- Krylov methods: apply a better degree t polynomial $T_t(\cdot)$ to the eigenvalues to separate $T_t(\lambda_1)$ from $T_t(\lambda_i)$.

Krylov subspace methods (Lanczos method, Arnoldi method.)

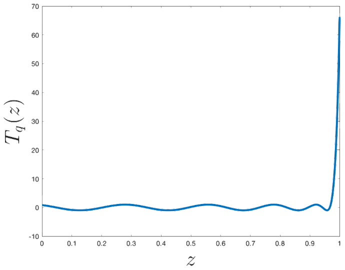
- How `svds`/`eigs` are actually implemented. Only need $t = O\left(\frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

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?



vs.



generalizations to larger k

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

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

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)
- Block Krylov methods

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

to accurately compute the top k singular vectors.

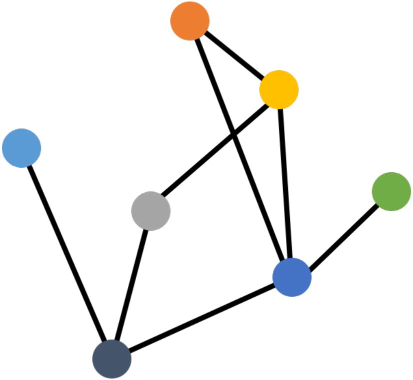
$$\textbf{`Gapless' Runtime: } O\left(ndk \cdot \frac{\ln(d/\epsilon)}{\sqrt{\epsilon}}\right)$$

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 (Bonus Material)

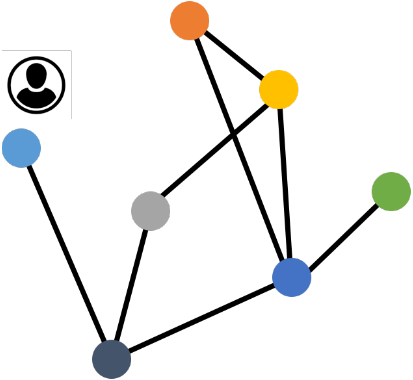
Connection to Random Walks

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



Connection to Random Walks

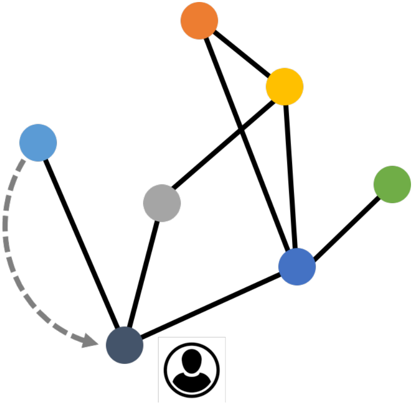
Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.



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

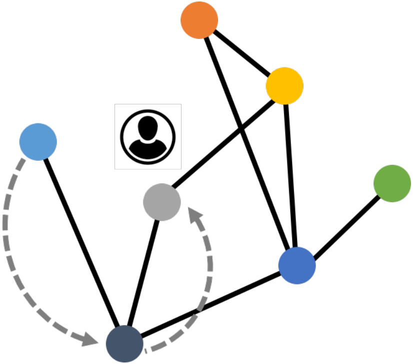
Connection to Random Walks

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



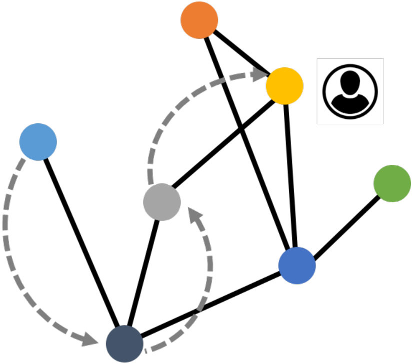
Connection to Random Walks

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 $\mathbf{A}$.



Connection to Random Walks

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

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:**

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

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

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 $\mathbf{AD}^{-1}$.

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 $\mathbf{A}\mathbf{D}^{-1}$.
- $\vec{p}^{(t)} = \mathbf{A}\mathbf{D}^{-1}\vec{p}^{(t-1)}$

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

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

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

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

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

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

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

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

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

- $\mathbf{D}^{-1/2}\vec{p}^{(t)}$ 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 $\mathbf{D}^{-1/2}\mathbf{AD}^{-1/2}$. 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.