COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Spring 2020.

Lecture 20

LOGISTICS

- Problem Set 3 is due tomorrow at 8pm. Problem Set 4 will be released very shortly.
- This is the last day of our spectral unit. Then will have 4 classes on optimization before end of semester.

Last Two 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.
- Idealized analysis in class. See slides for full analysis.

This Class: Computing the SVD/eigendecomposition.

- Discuss 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.

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?

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

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

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

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

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

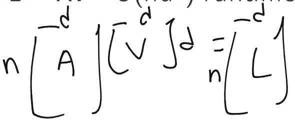
"direct method"

COMPUTING THE SVD

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

$$A^T A = V \Sigma^2 V^T$$

- 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 $\mathbf{L} = \mathbf{A}\mathbf{V}$ – $O(nd^2)$ runtime. Note that $\mathbf{L} = \mathbf{U}\boldsymbol{\Sigma}$.

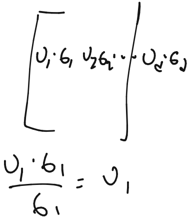


$$AV = U\Sigma V^T V = U\Sigma$$

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

- 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 $\mathbf{L} = \mathbf{A}\mathbf{V}$ – $O(nd^2)$ 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: $O(nd^2 + d^3)$



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

- 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 $\mathbf{L} = \mathbf{A}\mathbf{V}$ – $O(nd^2)$ 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.

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

COMPUTING THE SVD

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

- 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 $\mathbf{L} = \mathbf{AV}$ – $O(nd^2)$ 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!

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

- 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 $\mathbf{L} = \mathbf{A}\mathbf{V}$ – $O(nd^2)$ 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.

  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).

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

- 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 $\mathbf{L} = \mathbf{AV}$ – $O(nd^2)$ 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.

$O(nd^{1.7})$

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.

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 $\mathbf{U}_k \boldsymbol{\Sigma}_k = \mathbf{X}\mathbf{V}_k$.
- 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 of a matrix $\mathbf{X} \in \mathbb{R}^{n \times k}$ for $k \ll d$.

- Suffices to compute $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ and then compute $\mathbf{U}_k \boldsymbol{\Sigma}_k = \mathbf{X}\mathbf{V}_k$.
- 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 $O(nd^2)$.

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

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

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

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

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

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

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

  Return $\vec{z}_t$

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

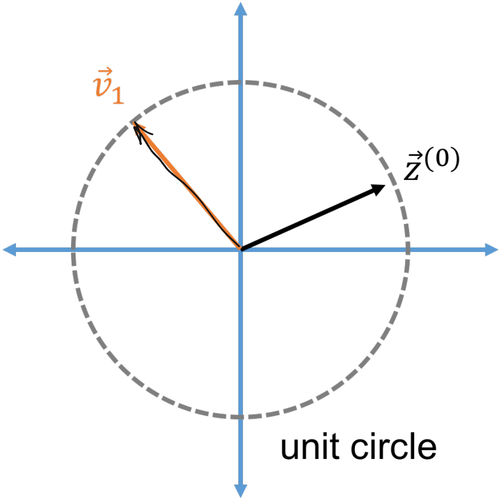
Goal: Given $\mathbf{X} \in \mathbb{R}^{n \times d}$, with SVD $\mathbf{X} = \mathbf{U\Sigma V}$, find $\vec{z} \approx \vec{v}_1$.

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

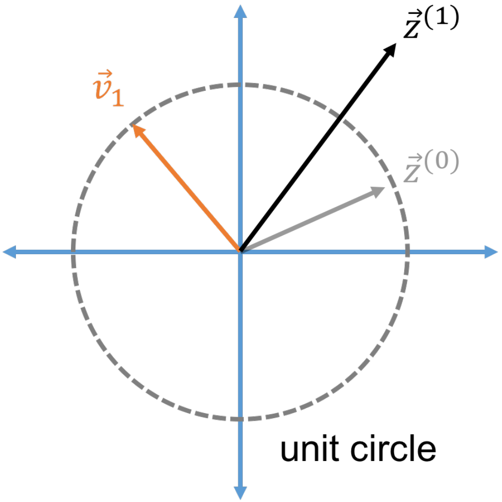
  Return $\vec{z}_t$ $\vec{z}^{(t)}$

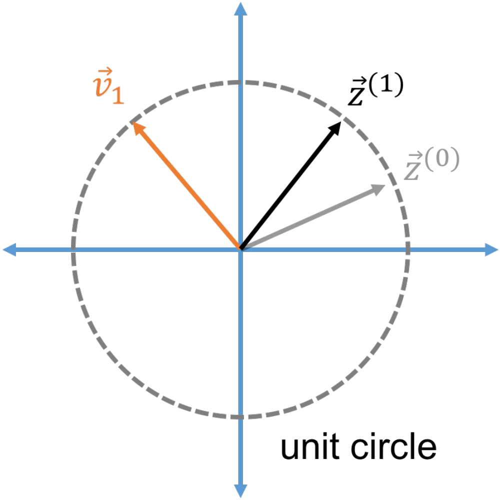
$O(nd^2) + O(dt)$

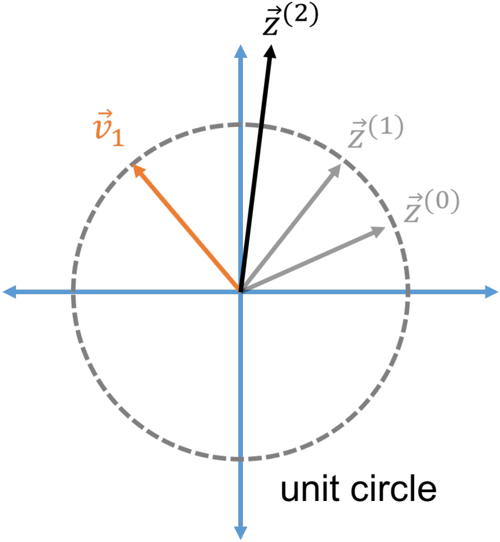
Total Runtime: $O(ndt)$

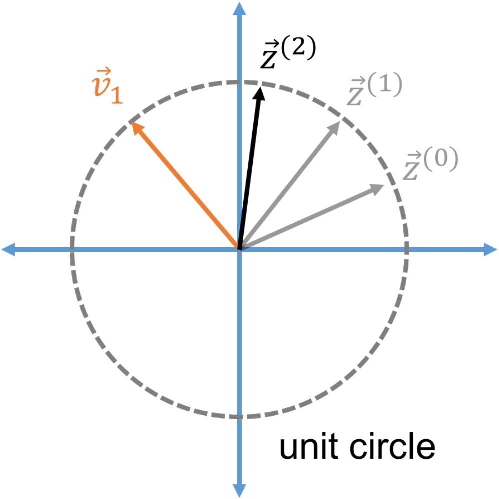


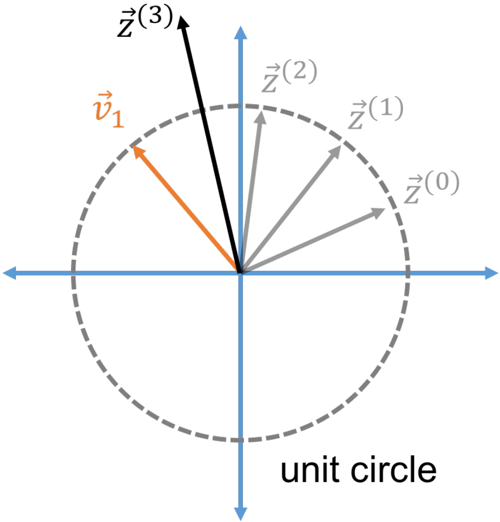
$\vec{v}_1$
$\vec{z}^{(0)}$
unit circle


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

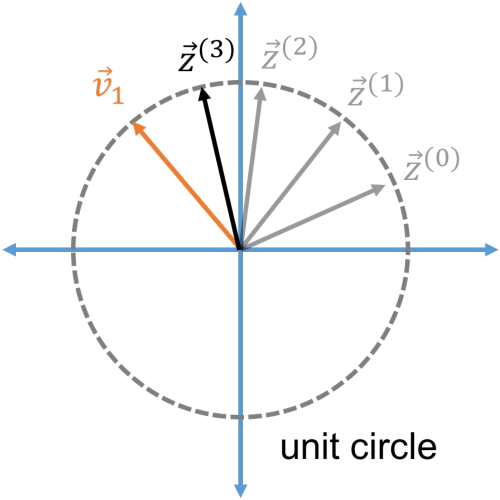

$\vec{v}_1$
$\vec{z}^{(1)}$
$\vec{z}^{(0)}$
unit circle

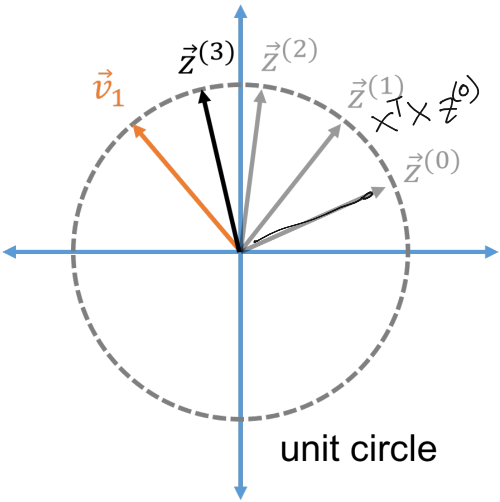

$\vec{z}^{(2)}$
$\vec{v}_1$
$\vec{z}^{(1)}$
$\vec{z}^{(0)}$
unit circle


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



$\vec{z}^{(3)}$
$\vec{z}^{(2)}$
$\vec{v}_1$
$\vec{z}^{(1)}$
$\vec{z}^{(0)}$
unit circle


$\vec{z}^{(3)}$
$\vec{z}^{(2)}$
$\vec{v}_1$
$\vec{z}^{(1)}$
$\vec{z}^{(0)}$
unit circle



Why is it converging towards $\vec{v}_1$?

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

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

$\mathbf{X} \in \mathbb{R}^{n\times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

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

$X = U\Sigma V^T$

$X^T X = V\Sigma^2 V^T$

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

$\mathbf{X} \in \mathbb{R}^{n\times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

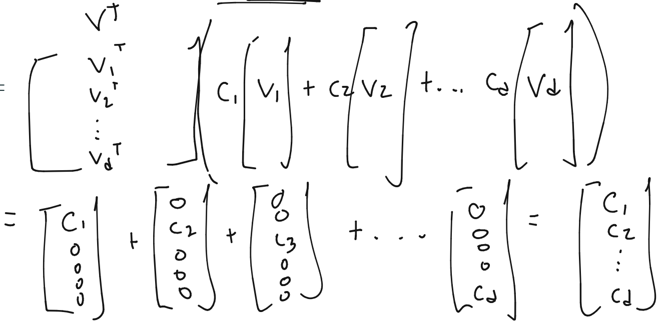
POWER METHOD INTUITION

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

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

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

$\mathbf{V}^T\vec{z}^{(0)} =$



$\mathbf{X} \in \mathbb{R}^{n\times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

POWER METHOD INTUITION

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

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

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

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

$$\boldsymbol{\Sigma}^2\mathbf{V}^T\vec{z}^{(0)} = \begin{bmatrix} c_1\sigma_1^2 \\ c_2\sigma_2^2 \\ \vdots \\ c_d \cdot \sigma_d^2 \end{bmatrix}$$

$\mathbf{X} \in \mathbb{R}^{n\times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

POWER METHOD INTUITION

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

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

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

$$\mathbf{V}^T\vec{z}^{(0)} =$$

$$\mathbf{\Sigma}^2\mathbf{V}^T\vec{z}^{(0)} = \begin{bmatrix} \sigma_1^2 c_1 \\ \vdots \\ \sigma_d^2 c_d \end{bmatrix}$$

$$\vec{z}^{(1)} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \cdot \vec{z}^{(0)} = \begin{bmatrix} v_1 \cdots v_d \end{bmatrix} \begin{bmatrix} \sigma_1^2 c_1 \\ \sigma_2^2 c_2 \\ \vdots \\ \sigma_d^2 c_d \end{bmatrix} = \underbrace{\sigma_1^2 c_1 v_1 + \sigma_2^2 c_2 v_2 + \ldots \sigma_d^2 c_d v_d}_{z^{(1)}}$$

$\mathbf{X} \in \mathbb{R}^{n\times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

$$\vec{z}^{(1)} = c_1 \cdot \sigma_1^2\vec{v}_1 + c_2 \cdot \sigma_2^2\vec{v}_2 + \ldots + c_d \cdot \sigma_d^2\vec{v}_d.$$

$\mathbf{X} \in \mathbb{R}^{n\times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

POWER METHOD INTUITION

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

$$\vec{z}^{(1)} = c_1 \cdot \sigma_1^2 \vec{v}_1 + c_2 \cdot \sigma_2^2 \vec{v}_2 + \ldots + c_d \cdot \sigma_d^2 \vec{v}_d.$$

$$\vec{z}^{(2)} = \mathbf{X}^T\mathbf{X}\vec{z}^{(1)} = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T\vec{z}^{(1)} = c_1\sigma_1^4 v_1 + \ldots \; c_d\sigma_d^4 v_d$$

$\mathbf{X} \in \mathbb{R}^{n\times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

$$\vec{z}^{(1)} = c_1 \cdot \sigma_1^2 \vec{v}_1 + c_2 \cdot \sigma_2^2 \vec{v}_2 + \ldots + c_d \cdot \sigma_d^2 \vec{v}_d.$$

$$\vec{z}^{(2)} = \mathbf{X}^T\mathbf{X}\vec{z}^{(1)} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T\vec{z}^{(1)} =$$

Claim 2:

$$\vec{z}^{(t)} = c_1 \cdot \sigma_1^{2t} \vec{v}_1 + c_2 \cdot \sigma_2^{2t} \vec{v}_2 + \ldots + c_d \cdot \sigma_d^{2t} \vec{v}_d.$$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: input matrix with SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. $\vec{v}_1$: top right singular vector, 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 singular values, making the component in the direction of v_1 much larger, relative to the other components.

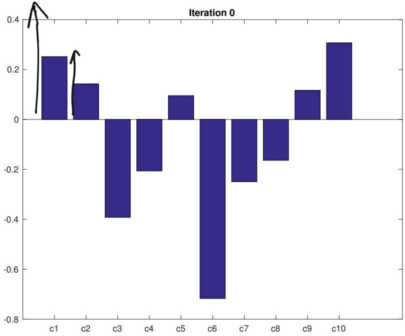
$z^t \approx v_1$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

POWER METHOD CONVERGENCE

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

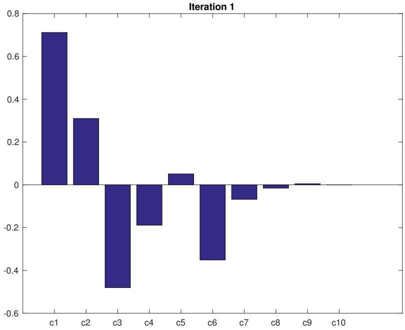
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



POWER METHOD CONVERGENCE

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

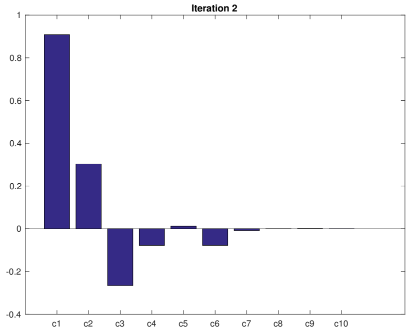
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



POWER METHOD CONVERGENCE

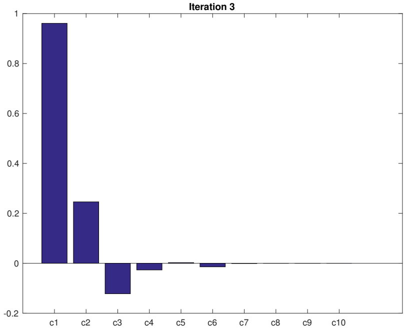
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



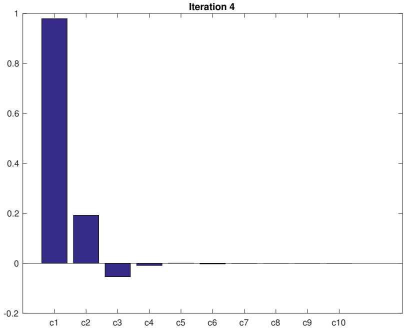
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



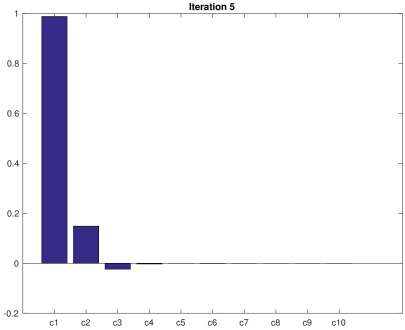
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



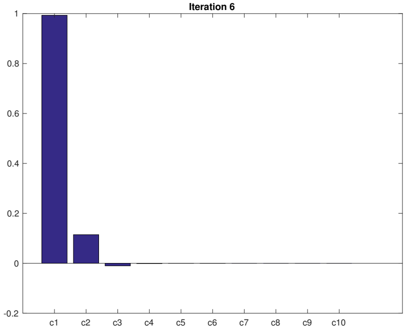
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



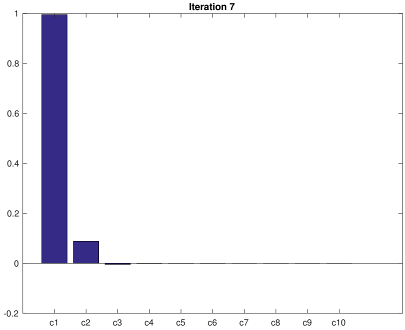
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



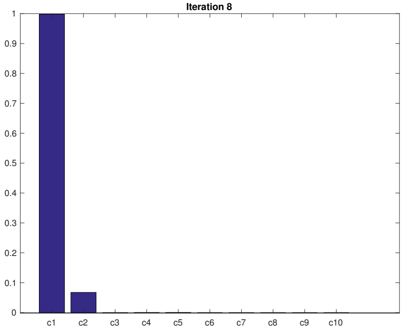
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



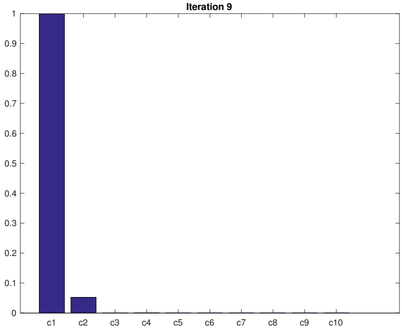
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



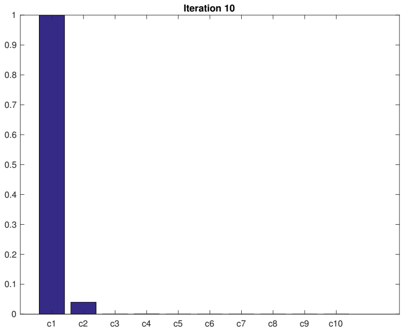
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



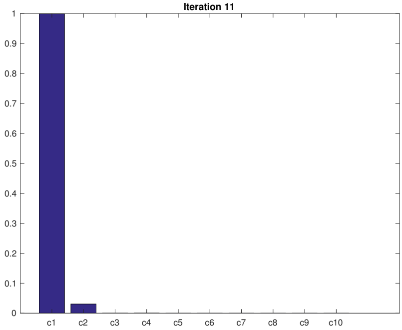
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



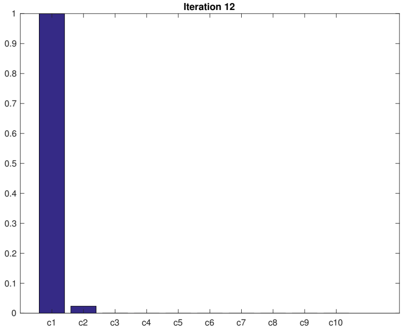
After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



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

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

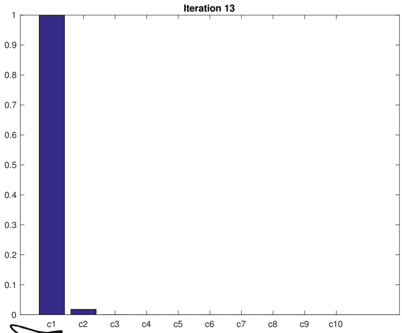


After t iterations, we have 'powered' up the singular values, making the component in the direction of v_1 much larger, relative to the other components.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

$z^{(t)} \approx v_1$

$z^{(t)} \approx -v_1$



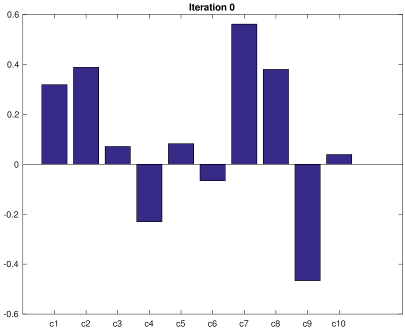
When will convergence be slow?

Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

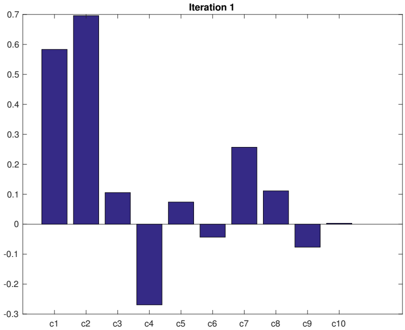
Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



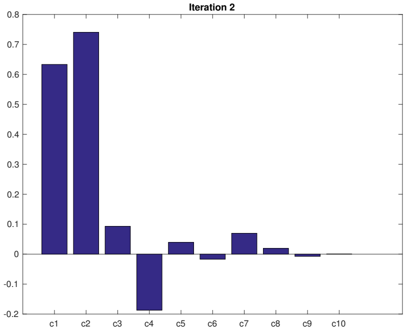
Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



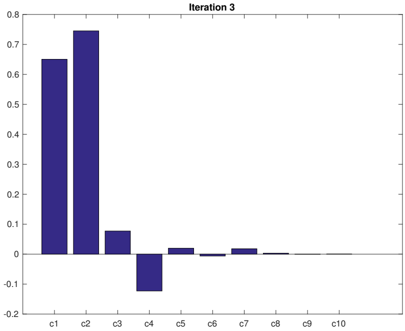
Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



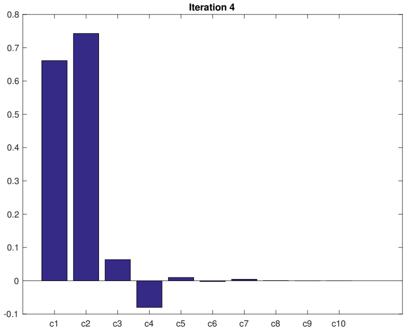
Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

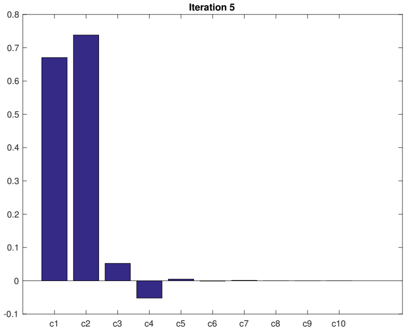
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



POWER METHOD SLOW CONVERGENCE

Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

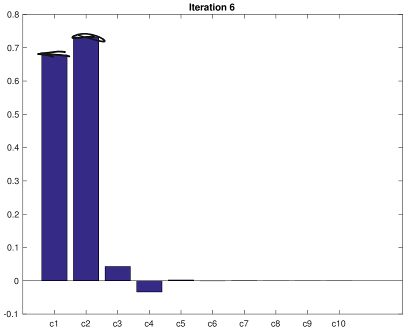
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



POWER METHOD SLOW CONVERGENCE

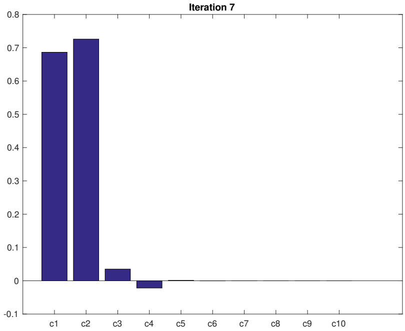
Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

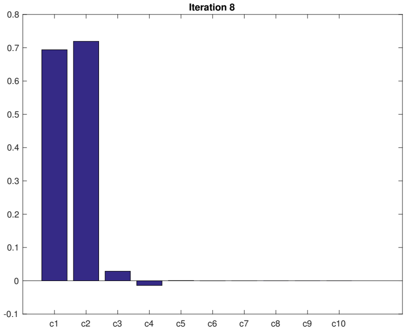
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



POWER METHOD SLOW CONVERGENCE

Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

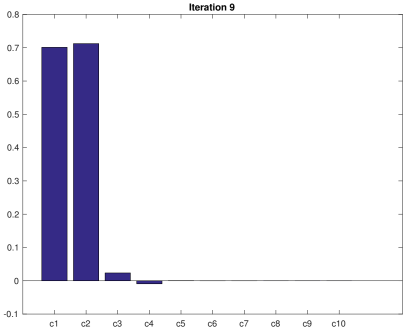
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



POWER METHOD SLOW CONVERGENCE

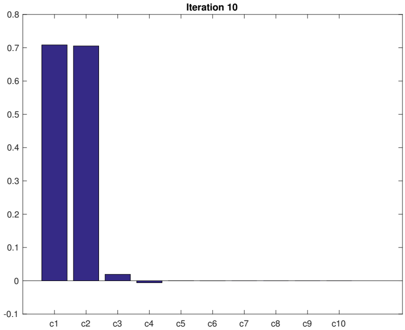
Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

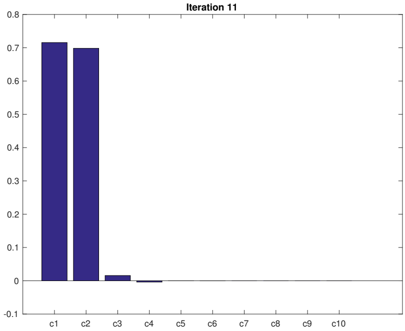
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



POWER METHOD SLOW CONVERGENCE

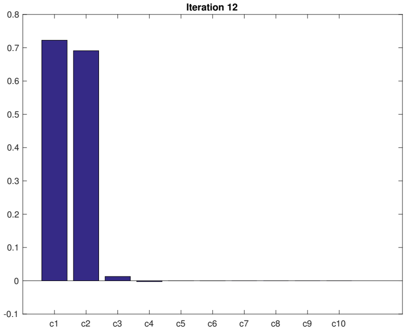
Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$



Slow Case: X has singular values: $\sigma_1 = 1, \sigma_2 = .99, \sigma_3 = .9, \sigma_4 = .8, \ldots$

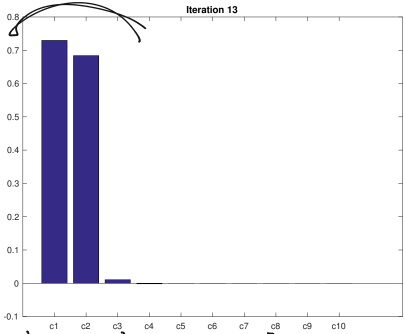
$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

$$X^T X v_1 = \lambda_1 v_1$$

$$X^T X v_2 = \lambda_1 v_2$$

$$X^T X (v_1 + v_2) = \lambda_1 (v_1 + v_2)$$

$$\sum_{i=k+1}^{n} \sigma_i^2$$



$$\|X - X v_1 v_1^T\|_F^2 = \sum_{i=2}^{d} \sigma_i(X)^2$$

$$\|X - X v_2 v_2^T\|_F^2 = \sigma_1(X)^2 + \sigma_3(X)^2 + \ldots \sigma_d(X)^2$$

$\gamma \in [0,1]$

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

Write $\sigma_2 = (1-\gamma)\sigma_1$ for 'gap' $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$. How many iterations t does it take to have $\sigma_2^{2t} \leq \frac{1}{2} \cdot \sigma_1^{2t}$?

$1 - \frac{\sigma_2}{\sigma_1}$ $\qquad$ $1-\gamma = \frac{\sigma_2}{\sigma_1}$

$\frac{\sigma_2^{2t}}{\sigma_1^{2t}} \leq \frac{1}{2}$ $\qquad$ $\left(\frac{\sigma_2}{\sigma_1}\right)^{2t} \leq \frac{1}{2}$

$(1-\gamma)^{2t} \leq \frac{1}{2}$ $\qquad$ $2t = \log_{1-\gamma} 1/2$

$(1-\gamma)^{1/\gamma} \leq \frac{1}{e} < \frac{1}{2}$ $\qquad$ $t = \frac{1}{2\gamma}$ $\qquad$ $(1-\gamma)^{2t} = (1-\gamma)^{1/\gamma} \leq \frac{1}{2}$

$t = O(1/\gamma)$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

Write $\sigma_2 = (1-\gamma)\sigma_1$ for 'gap' $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$. How many iterations t does it take to have $\sigma_2^{2t} \leq \frac{1}{2} \cdot \sigma_1^{2t}$? $O(1/\gamma)$.

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

Write $\sigma_2 = (1-\gamma)\sigma_1$ for 'gap' $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$. How many iterations t does it take to have $\sigma_2^{2t} \leq \frac{1}{2} \cdot \sigma_1^{2t}$? $O(1/\gamma)$.

$\delta \geq 0$

How many iterations t does it take to have $\sigma_2^{2t} \leq \delta \cdot \sigma_1^{2t}$?

after $1/\gamma$ iterations

$\sigma_2^{2t} = \sigma_2^{2/\gamma} \leq \frac{1}{2}\sigma_1^{2/\gamma}$

$\sigma_2^{\frac{\log(1/\delta)}{\gamma}} = \left(\sigma_2^{1/\gamma}\right)^{\log(1/\delta)} = \frac{1}{2^{\log(1/\delta)}}\left(\sigma_1^{1/\gamma}\right)^{\log(1/\delta)}$

$\leq \delta\, \sigma_1^{1/\gamma \cdot \log(1/\delta)}$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$

Write $\sigma_2 = (1-\gamma)\sigma_1$ for 'gap' $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$. How many iterations t does it take to have $\sigma_2^{2t} \leq \frac{1}{2} \cdot \sigma_1^{2t}$? $O(1/\gamma)$.

How many iterations t does it take to have $\sigma_2^{2t} \leq \delta \cdot \sigma_1^{2t}$? $O\left(\frac{\log(1/\delta)}{\gamma}\right)$.

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

$$\vec{z}^{(0)} = c_1\vec{v}_1 + c_2\vec{v}_2 + \ldots + c_d\vec{v}_d \implies \vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d$$

Write $\sigma_2 = (1-\gamma)\sigma_1$ for 'gap' $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$. How many iterations t does it take to have $\sigma_2^{2t} \leq \frac{1}{2} \cdot \sigma_1^{2t}$? $O(1/\gamma)$.

How many iterations t does it take to have $\sigma_2^{2t} \leq \delta \cdot \sigma_1^{2t}$? $O\left(\frac{\log(1/\delta)}{\gamma}\right)$.

How small must we set δ to ensure that $c_1\sigma_1^{2t}$ dominates all other components and so $\vec{z}^{(t)}$ is very close to $\vec{v}_1$?

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

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

$\frac{1}{100d^2}$ $10\log(d)$

$\mathbf{X} \in \mathbb{R}^{n\times d}$: matrix with SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

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

Corollary:

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

$\mathbf{X} \in \mathbb{R}^{n\times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

Claim 2: For gap $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$, after $t = O\left(\frac{\log(1/\delta)}{\gamma}\right)$ iterations:

$$\vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d \propto c_1\vec{v}_1 + c_2\delta\vec{v}_2 + \ldots + c_d\delta\vec{v}_d$$

$\mathbf{X} \in \mathbb{R}^{n\times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

Claim 2: For gap $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$, after $t = O\left(\frac{\log(1/\delta)}{\gamma}\right)$ iterations:

$$\vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d \propto c_1\vec{v}_1 + c_2\delta\vec{v}_2 + \ldots + c_d\delta\vec{v}_d$$

If we set $\delta = O\left(\frac{\epsilon}{d^3 \log d}\right)$ by Claim 1 will have:

$$\vec{z}^{(t)} \propto \vec{v}_1 + \frac{\epsilon}{d}\left(\vec{v}_2 + \ldots + \vec{v}_d\right).$$

$c_j \delta \leq d^2 \log d \cdot c_1 \cdot \delta$

$\leq \frac{\epsilon}{d} \cdot c_1$

$\|v_1\| = 1$ $\quad$ $\|\text{error}\| \leq \epsilon$

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

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

Claim 2: For gap $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$, after $t = O\left(\frac{\log(1/\delta)}{\gamma}\right)$ iterations:

$\vec{z}^{(t)} = c_1\sigma_1^{2t}\vec{v}_1 + c_2\sigma_2^{2t}\vec{v}_2 + \ldots + c_d\sigma_d^{2t}\vec{v}_d \propto c_1\vec{v}_1 + c_2\delta\vec{v}_2 + \ldots + c_d\delta\vec{v}_d$

If we set $\delta = O\left(\frac{\epsilon}{d^3 \log d}\right)$ by Claim 1 will have:

$$\vec{z}^{(t)} \propto \vec{v}_1 + \frac{\epsilon}{d}\left(\vec{v}_2 + \ldots + \vec{v}_d\right).$$

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

$\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix with SVD $\mathbf{X} = \mathbf{U\Sigma V}^T$. Singular values $\sigma_1, \sigma_2, \ldots, \sigma_d$. $\vec{v}_1$: top right singular vector, being computed, $\vec{z}^{(i)}$: iterate at step i, converging to $\vec{v}_1$.

Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$ be the relative gap between the first and second largest singular values. If Power Method is initialized with a random Gaussian vector $\vec{v}^{(0)}$ then, with high probability, after $t = O\left(\frac{\log d/\epsilon}{\gamma}\right)$ steps:

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

$X - X v_1 v_1^T$

$X^T X \begin{bmatrix} z_1 & z_2 & \cdots & z_k \end{bmatrix}$

Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$ *be the relative gap between the first and second largest singular values. If Power Method is initialized with a random Gaussian vector* $\vec{v}^{(0)}$ *then, with high probability, after* $t = O\left(\frac{\log d/\epsilon}{\gamma}\right)$ *steps:*

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

Total runtime: $O(t)$ matrix-vector multiplications.

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

Theorem (Basic Power Method Convergence)

Let $\gamma = \frac{\sigma_1 - \sigma_2}{\sigma_1}$ be the relative gap between the first and second largest singular values. If Power Method is initialized with a random Gaussian vector $\vec{v}^{(0)}$ then, with high probability, after $t = O\left(\frac{\log d/\epsilon}{\gamma}\right)$ steps:

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

Total runtime: $O(t)$ matrix-vector multiplications.

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

How is ϵ dependence?

How is γ dependence?

Krylov subspace methods (Lanczos method, Arnoldi method.)

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

Krylov subspace methods (Lanczos method, Arnoldi method.)

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

Main Idea: Need to separate σ_1 from σ_i for $i \geq 2$.

Krylov subspace methods (Lanczos method, Arnoldi method.)

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

Main Idea: Need to separate σ_1 from σ_i for $i \geq 2$.

- Power method: power up to $\sigma_1^{2\cdot t}$ and $\sigma_i^{2\cdot t}$.

Krylov subspace methods (Lanczos method, Arnoldi method.)

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

Main Idea: Need to separate σ_1 from σ_i for $i \geq 2$.

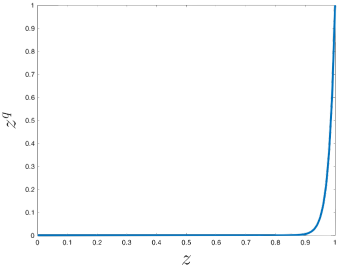
- Power method: power up to $\sigma_1^{2\cdot t}$ and $\sigma_i^{2\cdot t}$.
- Krylov methods: apply a better degree t polynomial $T_t(\sigma_1^2)$ and $T_t(\sigma_i^2)$.

Krylov subspace methods (Lanczos method, Arnoldi method.)

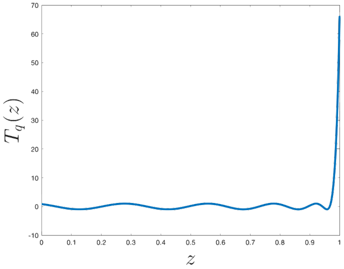
- How `svds`/`eigs` are actually implemented. Only need $t = O\left(\frac{\log d/\epsilon}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

Main Idea: Need to separate σ_1 from σ_i for $i \geq 2$.

- Power method: power up to $\sigma_1^{2\cdot t}$ and $\sigma_i^{2\cdot t}$.
- Krylov methods: apply a better degree t polynomial $T_t(\sigma_1^2)$ and $T_t(\sigma_i^2)$.
- Still requires just $2t$ matrix vector multiplies. Why?



vs.



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 aka Simultaneous Iteration aka Subspace Iteration aka Orthogonal Iteration
- Block Krylov methods

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

to accurately compute the top k singular vectors.

- Block Power Method aka Simultaneous Iteration aka Subspace Iteration aka Orthogonal Iteration
- Block Krylov methods

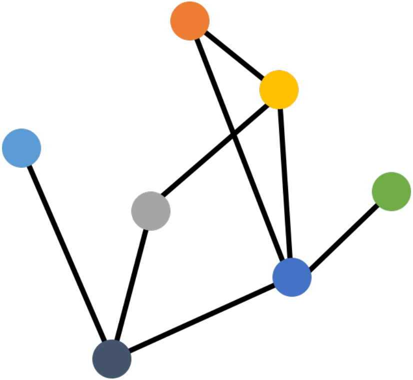
$$\text{Runtime: } O\left(ndk \cdot \frac{\log d/\epsilon}{\sqrt{\gamma}}\right)$$

to accurately compute the top k singular vectors.

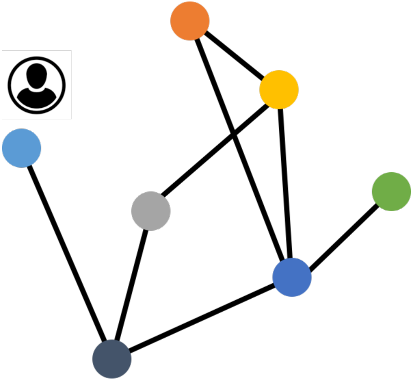
$$\text{`Gapless' Runtime: } O\left(ndk \cdot \frac{\log d/\epsilon}{\sqrt{\epsilon}}\right)$$

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

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

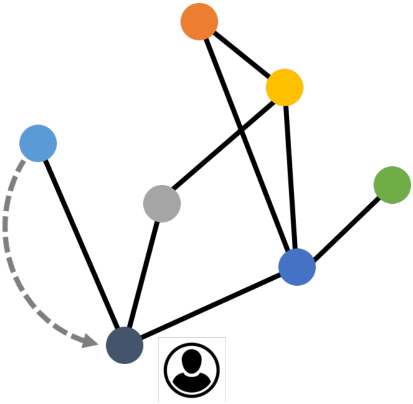


Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.

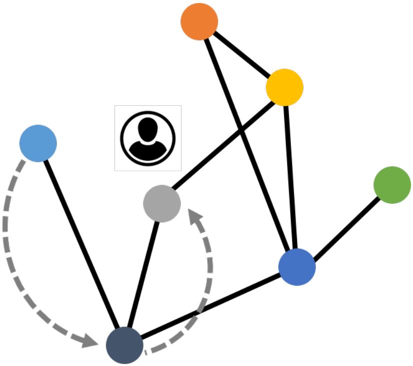


At each step, move to a random vertex, chosen uniformly at random from the neighbors of the current vertex.

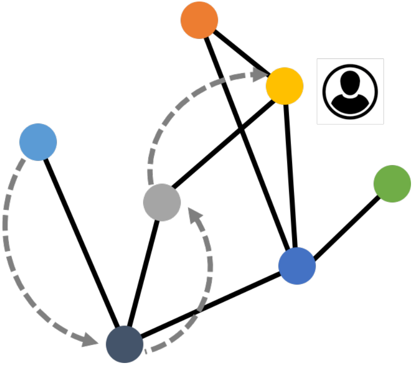
Consider a random walk on a graph G with adjacency matrix $\mathbf{A}$.



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



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 $\vec{p}_i^{(t)} = \Pr(\text{walk at node i at step t})$.

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

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

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

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

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 $j \notin neigh(i)$.

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

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

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

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

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

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

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

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

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

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

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

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 singular vector (eigenvector) of the normalized adjacency matrix $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Stationary distribution.

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 singular vector (eigenvector) of the normalized adjacency matrix $\mathbf{D}^{-1/2}\mathbf{AD}^{-1/2}$. 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 $\mathbf{AD}^{-1}$. The spectral gap.