COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2021.

Lecture 22

LOGISTICS

- Problem Set 4 due December 1.
- No quiz this week.
- We're going to start on optimization after break. And just cover a bit less material.

SUMMARY

Last Class:

- Efficient algorithms for SVD/eigendecomposition.
- Start on iterative methods: intuition behind the power method.

This Class:

- Finish power method analysis.
- Krylov subspace methods.
- Connections to random walks and Markov chains.

POWER METHOD

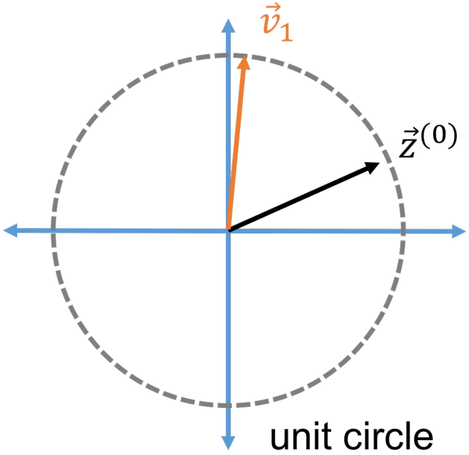
Power Method: The most fundamental iterative method for approximate SVD/eigendecomposition. Applies to computing $k = 1$ eigenvectors, but can be generalized to larger k.

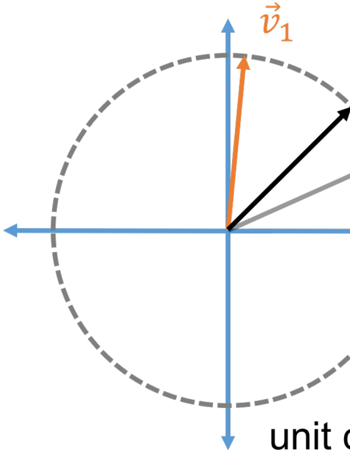
Goal: Given symmetric $\mathbf{A} \in \mathbb{R}^{d \times d}$, with eigendecomposition $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, find $\vec{z} \approx \vec{v}_1$ – the top eigenvector of $\mathbf{A}$.

- **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

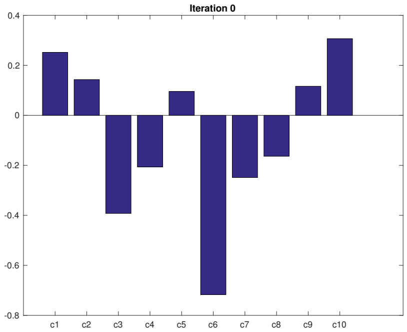


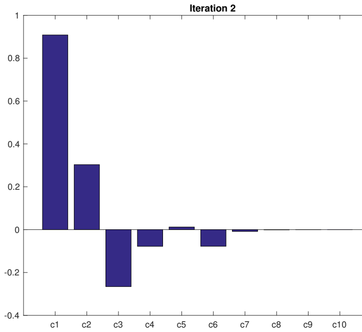


POWER METHOD CONVERGENCE

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 RATE

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

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 $|\lambda_2|^t \leq \frac{1}{e}\cdot|\lambda_1|^t$? $\mathbf{1/\gamma}$.

How many iterations t does it take to have $|\lambda_2|^t \leq \delta\cdot|\lambda_1|^t$? $\frac{\ln(1/\delta)}{\gamma}$.

Will have for all $i > 1$, $|\lambda_i|^t \leq |\lambda_2|^t \leq \delta\cdot|\lambda_1|^t$.

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 $\vec{v}_1$?

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

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:

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

Corollary:

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

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

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

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

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

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

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

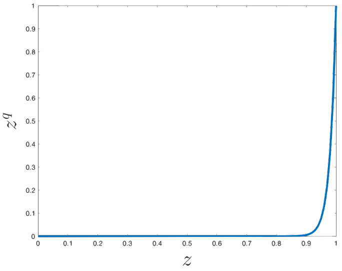
KRYLOV SUBSPACE METHODS

Krylov subspace methods (Lanczos method, Arnoldi method.)

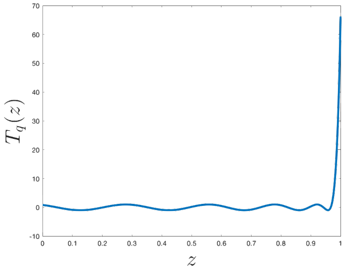
- How `svds/eigs` are actually implemented. Only need $t = O\left(\frac{\ln(d/\epsilon)}{\sqrt{\gamma}}\right)$ steps for the same guarantee.

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.



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

GENERALIZATIONS TO LARGER k

- Block Power Method (a.k.a. Simultaneous Iteration, Subspace Iteration, or Orthogonal Iteration)
- Standard Krylov methods (i.e., `svds`/`eigs`)
- Block Krylov methods

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

to accurately compute the top k singular vectors.

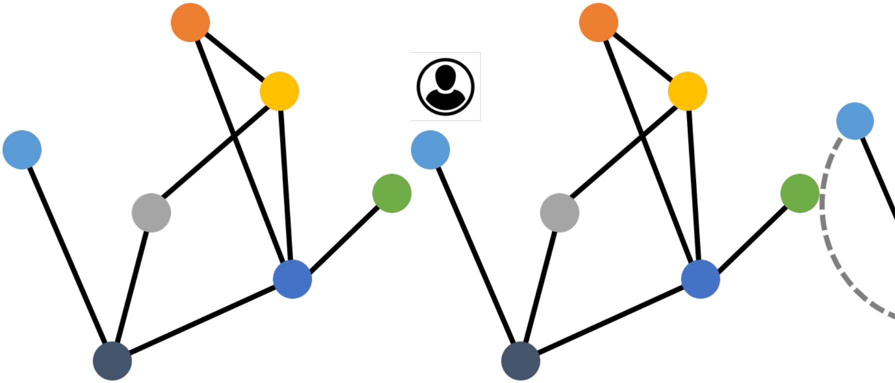
$$\text{‘Gapless’ Runtime: } O\left(ndk \cdot \frac{\ln(d/\epsilon)}{\sqrt{\epsilon}}\right)$$

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

Connection Between Random Walks, Eigenvectors, and Power Method

CONNECTION TO RANDOM WALKS

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



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

CONNECTION TO RANDOM WALKS

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

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

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

  where $\vec{z}(j) = \frac{1}{degree(j)}$ for all $j \in neigh(i)$, $\vec{z}(j) = 0$ for all $j \notin neigh(i)$.
- $\vec{z}$ is the i^{th} row of the right normalized adjacency matrix $\mathbf{AD}^{-1}$.
- $\vec{p}^{(t)} = \mathbf{AD}^{-1}\vec{p}^{(t-1)} = \underbrace{\mathbf{AD}^{-1}\mathbf{AD}^{-1}\ldots\mathbf{AD}^{-1}}_{t \text{ times}}\vec{p}^{(0)}$

RANDOM WALKING AS POWER METHOD

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

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

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

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

RANDOM WALKING AS POWER METHOD

A small spectral gap for $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ corresponds to a small second smallest eigenvalue for the normalized Laplacian $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. Why?

Why does this make sense intuitively given what we know about the second smallest eigenvalue of the Laplacian?