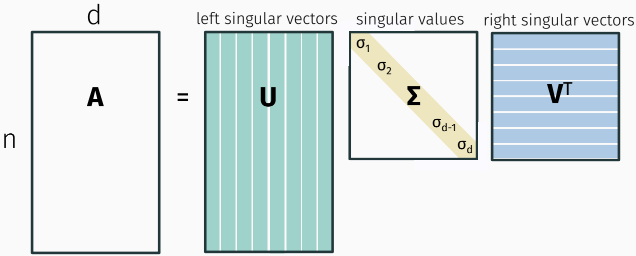
RANDOMIZED BLOCK KRYLOV METHODS FOR STRONGER AND FASTER APPROXIMATE SVD

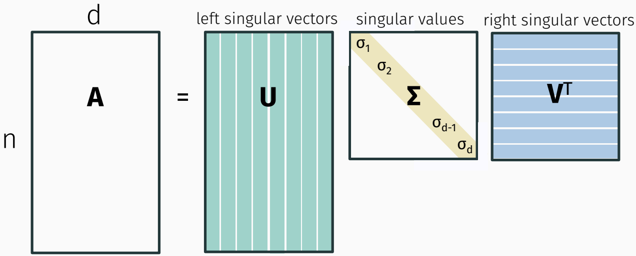
Cameron Musco and Christopher Musco

December 11, 2015

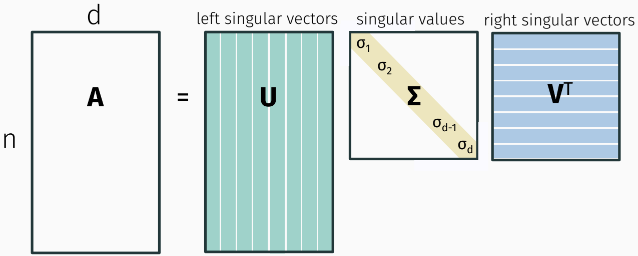
Massachusetts Institute of Technology, EECS

SINGULAR VALUE DECOMPOSITION



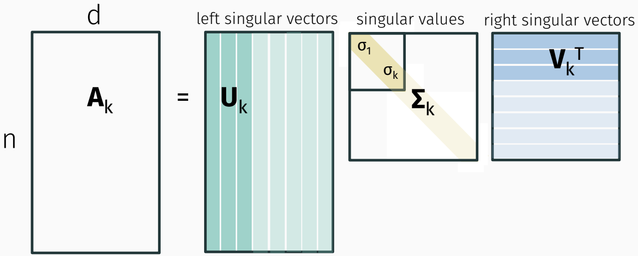


- Extremely important primitive for dimensionality reduction, low-rank approximation, PCA, etc.



- Extremely important primitive for dimensionality reduction, low-rank approximation, PCA, etc.

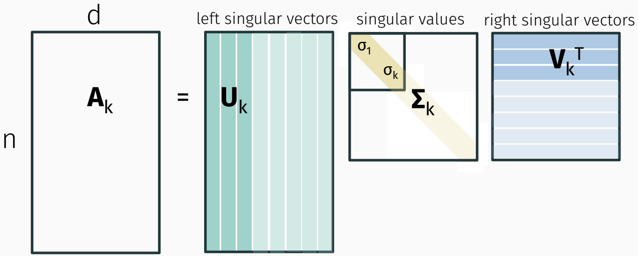
$$u_i = \underset{x:\|x\|=1, x \perp u_1, \ldots, u_{i-1}}{\arg\max} x^T \mathbf{A}\mathbf{A}^T x$$



- Extremely important primitive for dimensionality reduction, low-rank approximation, PCA, etc.

$$u_i = \operatorname*{arg\,max}_{x:\|x\|=1, x \perp u_1,\ldots,u_{i-1}} x^T \mathbf{A}\mathbf{A}^T x$$

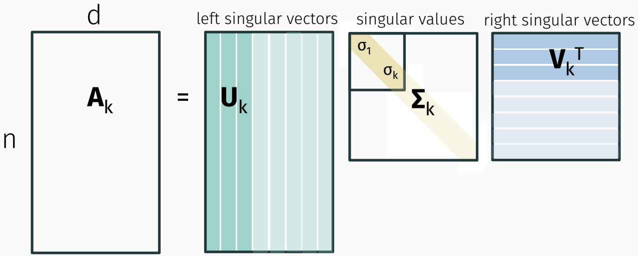
$$\mathbf{A}_k = \operatorname*{arg\,min}_{\mathbf{B}:rank(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_F$$



- Extremely important primitive for dimensionality reduction, low-rank approximation, PCA, etc.

$$u_i = \underset{x:\|x\|=1, x\perp u_1,\ldots,u_{i-1}}{\arg\max} \; x^T \mathbf{A}\mathbf{A}^T x$$

$$\mathbf{A}_k = \underset{\mathbf{B}:rank(\mathbf{B})=k}{\arg\min} \; \|\mathbf{A} - \mathbf{B}\|_2$$



- Extremely important primitive for dimensionality reduction, low-rank approximation, PCA, etc.

$$u_i = \underset{x:\|x\|=1, x \perp u_1, \ldots, u_{i-1}}{\arg\max} x^T \mathbf{A}\mathbf{A}^T x$$

$$\mathbf{U}_k \mathbf{U}_k^T \mathbf{A} = \underset{\mathbf{B}: rank(\mathbf{B})=k}{\arg\min} \|\mathbf{A} - \mathbf{B}\|_2$$

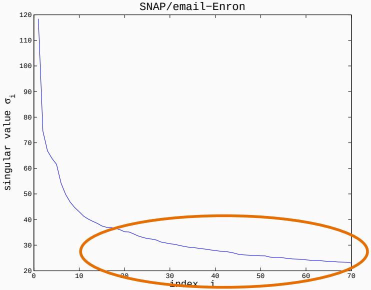
- Frobenius Norm Low-Rank Approximation:

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon) \|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F$$

- Frobenius Norm Low-Rank Approximation (weak):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F$$

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^d \sigma_i^2 \text{ and } \|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F^2 = \|\mathbf{A} - \mathbf{A}_k\|_F^2 = \sum_{i=k+1}^d \sigma_i^2.$$

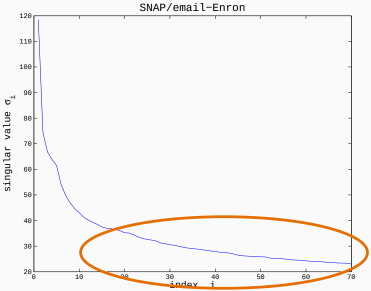


STANDARD SVD APPROXIMATION METRICS

- Frobenius Norm Low-Rank Approximation (weak):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F$$

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^d \sigma_i^2 \text{ and } \|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F^2 = \|\mathbf{A} - \mathbf{A}_k\|_F^2 = \sum_{i=k+1}^d \sigma_i^2.$$



For many datasets literally any $\tilde{\mathbf{U}}_k$ would work!

- Frobenius Norm Low-Rank Approximation (weak):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F$$

- Spectral Norm Low-Rank Approximation (stronger):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_2$$

STANDARD SVD APPROXIMATION METRICS

- Frobenius Norm Low-Rank Approximation (weak):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F$$

- Spectral Norm Low-Rank Approximation (stronger):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2 \leq (1+\epsilon)\sigma_{k+1}$$

- Frobenius Norm Low-Rank Approximation (weak):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F$$

- Spectral Norm Low-Rank Approximation (stronger):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2 \leq (1+\epsilon)\sigma_{k+1}$$

- Per Vector Principal Component Error (strongest):

$$\tilde{u}_i^T \mathbf{A}\mathbf{A}^T \tilde{u}_i \geq (1-\epsilon) u_i^T \mathbf{A}\mathbf{A}^T u_i \qquad \text{for all } i \leq k.$$

Classic Full SVD Algorithms (e.g. QR Algorithm):

All of these goals in roughly $O(nd^2)$ time (error dependence is $\log\log 1/\epsilon$ on lower order terms).

Unfortunately, this is much too slow for many data sets.

Classic Full SVD Algorithms (e.g. QR Algorithm):

All of these goals in roughly $O(nd^2)$ time (error dependence is $\log\log 1/\epsilon$ on lower order terms).

Unfortunately, this is much too slow for many data sets.

How fast can we approximately compute just $u_1, \ldots, u_k$?

‘Weak’ Approximation Algorithms:

- Strong Rank Revealing QR (Gu, Eisenstat 1996):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq \text{poly}(n, k)\|\mathbf{A} - \mathbf{A}_k\|_F \text{ in time } O(ndk)$$

'Weak' Approximation Algorithms:

- Strong Rank Revealing QR (Gu, Eisenstat 1996):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq \text{poly}(n, k)\|\mathbf{A} - \mathbf{A}_k\|_F \text{ in time } O(ndk)$$

- Sparse Subspace Embeddings (Clarkson, Woodruff 2013):

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F \text{ in time } O(\text{nnz}(\mathbf{A})) + \tilde{O}\left(\frac{nk^2}{\epsilon^4}\right)$$

Iterative methods are the only game in town for stronger guarantees. Runtime is approximately:

$$O(\mathrm{nnz}(\mathbf{A})k \cdot \#iterations)$$

- Power method (Müntz 1913, von Mises 1929)
- Krylov/Lanczos methods (Lanczos 1950)

Iterative methods are the only game in town for stronger guarantees. Runtime is approximately:

$$O(\text{nnz}(\mathbf{A})k \cdot \#iterations) \leq O(ndk \cdot \#iterations) << O(nd^2)$$

- Power method (Müntz 1913, von Mises 1929)
- Krylov/Lanczos methods (Lanczos 1950)

Iterative methods are the only game in town for stronger guarantees. Runtime is approximately:

$$O(\text{nnz}(\mathbf{A})k \cdot \#iterations) \leq O(ndk \cdot \#iterations) << O(nd^2)$$

- Power method (Müntz 1913, von Mises 1929)
- Krylov/Lanczos methods (Lanczos 1950)
- Stochastic Methods?

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d \times k}, \qquad \mathbf{X}_{i+1} \leftarrow \textit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 2 & \\ & & 1 \end{bmatrix}} \underset{\mathbf{X}_0}{\begin{bmatrix} .60 & .61 \\ .59 & -.10 \\ .54 & .78 \end{bmatrix}} \rightarrow \underset{\mathbf{X}_1}{\begin{bmatrix} .92 & -.18 \\ .37 & .54 \\ -.04 & .82 \end{bmatrix}} \quad \Big| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

POWER METHOD REVIEW

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d\times k}, \qquad \mathbf{X}_{i+1} \leftarrow \textit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$

$$\underbrace{\begin{bmatrix} 3 & & \\ & 2 & \\ & & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} .92 & -.18 \\ .37 & .54 \\ -.04 & .82 \end{bmatrix}}_{\mathbf{X}_1} \longrightarrow \underbrace{\begin{bmatrix} .98 & -.15 \\ .21 & .81 \\ -.05 & .56 \end{bmatrix}}_{\mathbf{X}_2} \quad \Big| \quad \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{\mathbf{U}_2}$$

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d\times k}, \qquad \mathbf{X}_{i+1} \leftarrow \textit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 2 & \\ & & 1 \end{bmatrix}} \underset{\mathbf{X}_2}{\begin{bmatrix} .98 & -.15 \\ .21 & .81 \\ -.05 & .56 \end{bmatrix}} \longrightarrow \underset{\mathbf{X}_3}{\begin{bmatrix} 1 & -.06 \\ .08 & .95 \\ -.04 & .31 \end{bmatrix}} \qquad \Big| \qquad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d\times k}, \qquad \mathbf{X}_{i+1} \leftarrow \textit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 2 & \\ & & 1 \end{bmatrix}} \quad \underset{\mathbf{X}_3}{\begin{bmatrix} 1 & -.06 \\ .08 & .95 \\ -.04 & .31 \end{bmatrix}} \longrightarrow \underset{\mathbf{X}_4}{\begin{bmatrix} 1 & -.02 \\ .02 & .99 \\ -.02 & .16 \end{bmatrix}} \quad \Bigg| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d\times k}, \qquad \mathbf{X}_{i+1} \leftarrow \textit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$

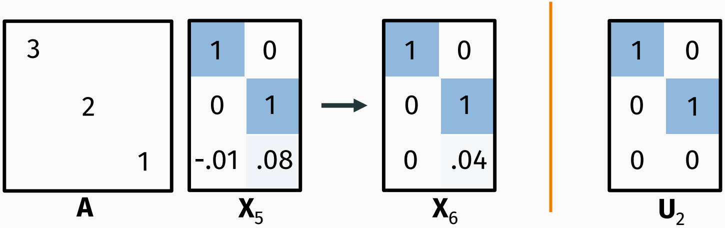
$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 2 & \\ & & 1 \end{bmatrix}} \underset{\mathbf{X}_4}{\begin{bmatrix} 1 & -.02 \\ .02 & .99 \\ -.02 & .16 \end{bmatrix}} \rightarrow \underset{\mathbf{X}_5}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -.01 & .08 \end{bmatrix}} \quad \Big| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d\times k}, \qquad \mathbf{X}_{i+1} \leftarrow \mathit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$

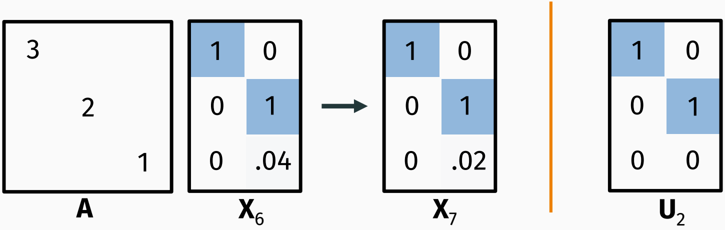


Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d\times k}, \qquad \mathbf{X}_{i+1} \leftarrow \mathit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$



Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{\mathbf{A}x_i}{\|\mathbf{A}x_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$\mathbf{X}_0 \in \mathbb{R}^{d \times k}, \qquad \mathbf{X}_{i+1} \leftarrow \mathit{orthonormalize}(\mathbf{A}\mathbf{X}_i)$$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 2 & \\ & & 1 \end{bmatrix}} \underset{\mathbf{X}_7}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & .02 \end{bmatrix}} \longrightarrow \underset{\mathbf{X}_8}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & .01 \end{bmatrix}} \quad \Big| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional Power Method:

$$x_0 \in \mathbb{R}^d, \qquad x_{i+1} \leftarrow \frac{Ax_i}{\|Ax_i\|}$$

Block Power Method (Simultaneous/Subspace Iteration):

$$X_0 \in \mathbb{R}^{d\times k}, \qquad X_{i+1} \leftarrow \mathit{orthonormalize}(AX_i)$$

$$\underbrace{\begin{bmatrix} 3 & & \\ & 2 & \\ & & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & .01 \end{bmatrix}}_{\mathbf{X}_8} \longrightarrow \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{\mathbf{X}_9} \quad \Bigg| \quad \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{\mathbf{U}_2}$$

Runtime for Block Power method is roughly:

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right)$$

Runtime for Block Power method is roughly:

$$O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right)$$

Runtime for Block Power method is roughly:

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right)$$

Runtime for Block Power method is roughly:

$$O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right)$$

- Linear dependence on the singular value gap:

$$\mathrm{gap} = \frac{\sigma_k - \sigma_{k+1}}{\sigma_k}$$

Runtime for Block Power method is roughly:

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right)$$

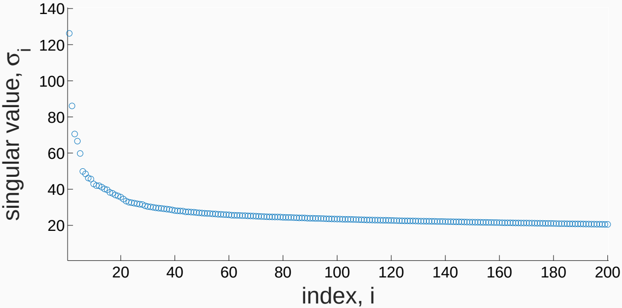
- Linear dependence on the singular value gap:

$$\text{gap} = \frac{\sigma_k - \sigma_{k+1}}{\sigma_k}$$

- While this gap is traditionally assumed to be constant, it is the dominant factor in the iteration count for many datasets.

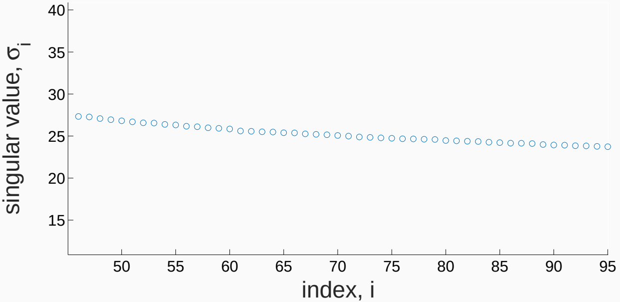
TYPICAL GAP VALUES

Stanford Network Analysis Project – Slashdot Social Network

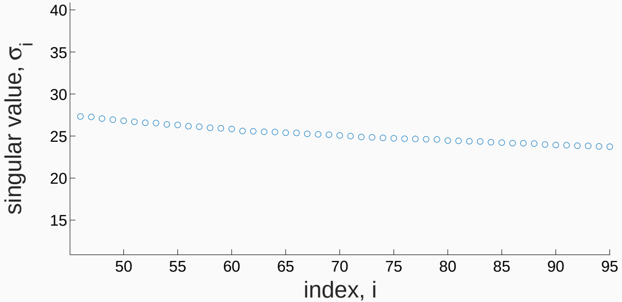


TYPICAL GAP VALUES

Stanford Network Analysis Project – Slashdot Social Network



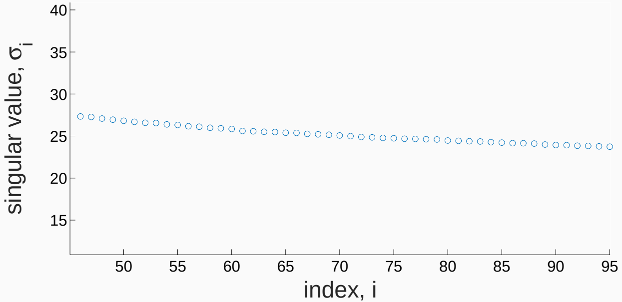
Stanford Network Analysis Project – Slashdot Social Network



Median value of $\mathrm{gap}_k = \frac{\sigma_k - \sigma_{k+1}}{\sigma_k}$ for $k \leq 200$:

.0019

Stanford Network Analysis Project – Slashdot Social Network

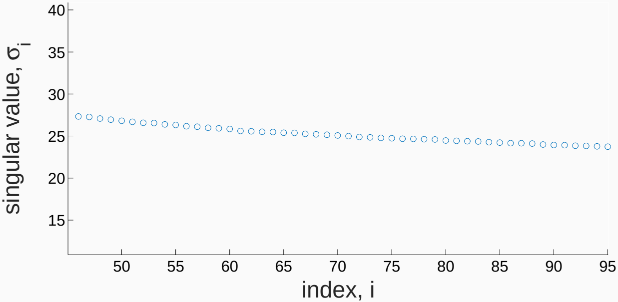


Minimum value of $\text{gap}_k = \frac{\sigma_k - \sigma_{k+1}}{\sigma_k}$ for $k \leq 200$:

.00004

TYPICAL GAP VALUES

Stanford Network Analysis Project – Slashdot Social Network



Minimum value of $\text{gap}_k = \frac{\sigma_k - \sigma_{k+1}}{\sigma_k}$ for $k \leq 200$:

Runtime = $O(25{,}000 \cdot \text{nnz}(\mathbf{A}) k \log(d/\epsilon))$

Recent work shows Block Power method (with randomized start vectors) gives:

Recent work shows Block Power method (with randomized start vectors) gives:

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_2 \text{ in time } O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\epsilon}\right)$$

GAP INDEPENDENT BOUNDS

Recent work shows Block Power method (with randomized start vectors) gives:

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_2 \text{ in time } O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\epsilon}\right)$$

Improves on classical bounds when $\epsilon > (\sigma_k - \sigma_{k+1})/\sigma_k$.

Recent work shows Block Power method (with randomized start vectors) gives:

$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_2 \text{ in time } O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log d}{\epsilon}\right)$$

Improves on classical bounds when $\epsilon > (\sigma_k - \sigma_{k+1})/\sigma_k$.

Long series of refinements and improvements:

- Rokhlin, Szlam, Tygert 2009
- Halko, Martinsson, Tropp 2011
- Boutsidis, Drineas, Magdon-Ismail 2011
- Witten, Candès 2014
- Woodruff 2014

Randomized Block Power Method is widely cited and implemented – simple algorithm with simple bounds.

RANDOMIZED BLOCK POWER METHOD

Randomized Block Power Method is widely cited and implemented – simple algorithm with simple bounds.

redSVD

scikit learn

IBM

libSkylark

ScaleNLP (Breeze)

Randomized Block Power Method is widely cited and implemented – simple algorithm with simple bounds.

redSVD

scikit learn

IBM

libSkylark

ScaleNLP (Breeze)

But in the numerical linear algebra community, Krylov/Lanczos methods have long been prefered over power iteration.

RANDOMIZED BLOCK POWER METHOD

Randomized Block Power Method is widely cited and implemented – simple algorithm with simple bounds.

But in the numerical linear algebra community, Krylov/Lanczos methods have long been prefered over power iteration.

Power Method

Krylov Methods

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right) \rightarrow O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{\sqrt{(\sigma_k - \sigma_{k+1})/\sigma_k}}\right)$$

Power Method | **Krylov Methods**

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right) \to O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{\sqrt{(\sigma_k - \sigma_{k+1})/\sigma_k}}\right)$$

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\epsilon}\right) \to \quad ?$$

No gap independent analysis of Krylov methods!

Power Method | **Krylov Methods**

$$O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{(\sigma_k - \sigma_{k+1})/\sigma_k}\right) \to O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log(d/\epsilon)}{\sqrt{(\sigma_k - \sigma_{k+1})/\sigma_k}}\right)$$

$$O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log d}{\epsilon}\right) \to \underbrace{O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{\epsilon}}\right)}_{\text{Our Contribution}}$$

A simple randomized Block Krylov Iteration gives all three of our target error bounds in time:

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{\epsilon}}\right)$$

$$\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^T\mathbf{A}\|_2 \leq (1+\epsilon)\sigma_{k+1} \quad \text{and} \quad \tilde{u}_i^T\mathbf{A}\mathbf{A}^T\tilde{u}_i \geq \sigma_i^2 - \epsilon\sigma_{k+1}^2$$

A simple randomized Block Krylov Iteration gives all three of our target error bounds in time:

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{\epsilon}}\right)$$

$$\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^T\mathbf{A}\|_2 \leq (1+\epsilon)\sigma_{k+1} \quad \text{and} \quad \tilde{u}_i^T\mathbf{A}\mathbf{A}^T\tilde{u}_i \geq \sigma_i^2 - \epsilon\sigma_{k+1}^2$$

- Gives a runtime bound that is independent of $\mathbf{A}$.

A simple randomized Block Krylov Iteration gives all three of our target error bounds in time:

$$O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{\epsilon}}\right)$$

$$\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^T\mathbf{A}\|_2 \leq (1+\epsilon)\sigma_{k+1} \quad \text{and} \quad \tilde{u}_i^T\mathbf{A}\mathbf{A}^T\tilde{u}_i \geq \sigma_i^2 - \epsilon\sigma_{k+1}^2$$

- Gives a runtime bound that is independent of **A**.
- Beats runtime of Block Power Method: $.0001 \rightarrow .01$.

A simple randomized Block Krylov Iteration gives all three of our target error bounds in time:

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{\epsilon}}\right)$$

$$\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^T\mathbf{A}\|_2 \leq (1+\epsilon)\sigma_{k+1} \quad \text{and} \quad \tilde{u}_i^T\mathbf{A}\mathbf{A}^T\tilde{u}_i \geq \sigma_i^2 - \epsilon\sigma_{k+1}^2$$

- Gives a runtime bound that is independent of $\mathbf{A}$.
- Beats runtime of Block Power Method: $10{,}000 \rightarrow 100$.

A simple randomized Block Krylov Iteration gives all three of our target error bounds in time:

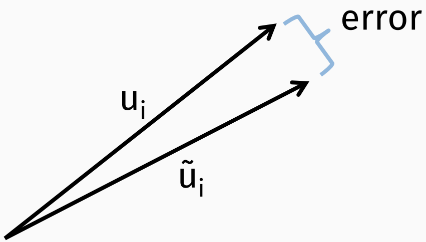
$$O\left(\mathrm{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{\epsilon}}\right)$$

$$\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^T\mathbf{A}\|_2 \leq (1+\epsilon)\sigma_{k+1} \quad \text{and} \quad \tilde{u}_i^T\mathbf{A}\mathbf{A}^T\tilde{u}_i \geq \sigma_i^2 - \epsilon\sigma_{k+1}^2$$

- Gives a runtime bound that is independent of $\mathbf{A}$.
- Beats runtime of Block Power Method: 10,000 $\rightarrow$ 100.
- Improves classic Lanczos bounds when $(\sigma_k - \sigma_{k+1})/\sigma_k < \epsilon$.

First Step: Where does gap dependence actually comes from?

To prove guarantees like: $\tilde{u}_i^T \mathbf{A}\mathbf{A}^T \tilde{u}_i \geq (1-\epsilon)\sigma_i^2$, classical analysis argues about convergence to $\mathbf{A}$'s true singular vectors.



Traditional objective function: $\|u_i - \tilde{u}_i\|_2$.

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

- Simple potential function, easy to work with.

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

- Simple potential function, easy to work with.
- Can be used to prove strong per-vector error or spectral norm guarantees for $\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2$.

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

- Simple potential function, easy to work with.
- Can be used to prove strong per-vector error or spectral norm guarantees for $\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2$.
- Inherently requires an iteration count that depends on singular value gaps.

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underbrace{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}}_{\mathbf{A}} \quad \underbrace{\begin{bmatrix} .93 & .26 \\ -.35 & .51 \\ -.08 & .81 \end{bmatrix}}_{\mathbf{X}_0} \longrightarrow \underbrace{\begin{bmatrix} 1 & .01 \\ -.07 & .62 \\ .04 & .78 \end{bmatrix}}_{\mathbf{X}_1} \quad \Bigg| \quad \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{\mathbf{U}_2}$$

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underbrace{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}}_{\mathbf{A}} \quad \underbrace{\begin{bmatrix} 1 & .01 \\ -.07 & .62 \\ .04 & .78 \end{bmatrix}}_{\mathbf{X}_1} \longrightarrow \underbrace{\begin{bmatrix} 1 & 0 \\ -.02 & .65 \\ .02 & .75 \end{bmatrix}}_{\mathbf{X}_2} \quad \Bigg| \quad \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{\mathbf{U}_2}$$

UNDERSTANDING GAP DEPENDENCE

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}} \quad \underset{\mathbf{X}_2}{\begin{bmatrix} 1 & 0 \\ -.02 & .65 \\ .02 & .75 \end{bmatrix}} \rightarrow \underset{\mathbf{X}_3}{\begin{bmatrix} 1 & 0 \\ -.01 & .69 \\ .01 & .72 \end{bmatrix}} \quad \Big| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}} \quad \underset{\mathbf{X}_3}{\begin{bmatrix} 1 & 0 \\ -.01 & .69 \\ .01 & .72 \end{bmatrix}} \longrightarrow \underset{\mathbf{X}_4}{\begin{bmatrix} 1 & 0 \\ 0 & .72 \\ 0 & .68 \end{bmatrix}} \quad \Big| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underbrace{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}}_{\mathbf{A}} \quad \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & .72 \\ 0 & .68 \end{bmatrix}}_{\mathbf{X}_4} \longrightarrow \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & .76 \\ 0 & .65 \end{bmatrix}}_{\mathbf{X}_5} \quad \Bigg| \quad \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}_{\mathbf{U}_2}$$

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}} \quad \underset{\mathbf{X}_5}{\begin{bmatrix} 1 & 0 \\ 0 & .76 \\ 0 & .65 \end{bmatrix}} \longrightarrow \underset{\mathbf{X}_6}{\begin{bmatrix} 1 & 0 \\ 0 & .78 \\ 0 & .62 \end{bmatrix}} \quad \Big| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}} \quad \underset{\mathbf{X}_6}{\begin{bmatrix} 1 & 0 \\ 0 & .78 \\ 0 & .62 \end{bmatrix}} \longrightarrow \underset{\mathbf{X}_7}{\begin{bmatrix} 1 & 0 \\ 0 & .82 \\ 0 & .58 \end{bmatrix}} \quad \Bigg| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

$$\underset{\mathbf{A}}{\begin{bmatrix}3 & & \\ & 1.1 & \\ & & 1\end{bmatrix}} \underset{\mathbf{X}_7}{\begin{bmatrix}1 & 0 \\ 0 & .82 \\ 0 & .58\end{bmatrix}} \longrightarrow \underset{\mathbf{X}_8}{\begin{bmatrix}1 & 0 \\ 0 & .84 \\ 0 & .54\end{bmatrix}} \quad \Bigg| \quad \underset{\mathbf{U}_2}{\begin{bmatrix}1 & 0 \\ 0 & 1 \\ 0 & 0\end{bmatrix}}$$

UNDERSTANDING GAP DEPENDENCE

Traditional objective function: $\|u_i - \tilde{u}_i\|_2$

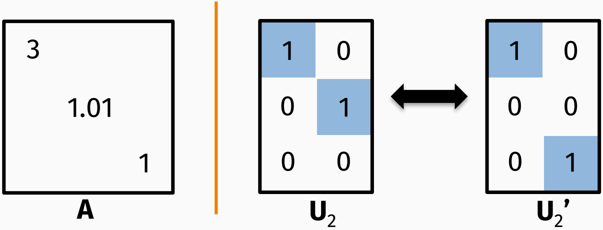
$$\underset{\mathbf{A}}{\begin{bmatrix} 3 & & \\ & 1.1 & \\ & & 1 \end{bmatrix}} \quad \underset{\mathbf{X}_8}{\begin{bmatrix} 1 & 0 \\ 0 & .84 \\ 0 & .54 \end{bmatrix}} \longrightarrow \underset{\mathbf{X}_9}{\begin{bmatrix} 1 & 0 \\ 0 & .86 \\ 0 & .51 \end{bmatrix}} \quad \Big| \quad \underset{\mathbf{U}_2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}$$

KEY INSIGHT

Convergence becomes less necessary precisely when it is difficult to achieve!

KEY INSIGHT

Convergence becomes less necessary precisely when it is difficult to achieve!



Minimizing $\|u_i - \tilde{u}_i\|_2$ is sufficient, but far from necessary.

Iterative methods viewed as denoising procedures for Random Sketching methods.

Iterative methods viewed as denoising procedures for Random Sketching methods.

Choose $\mathsf{G} \sim \mathcal{N}(0,1)^{d \times k}$. If A is rank k then:

$$span\,(\mathsf{AG}) = span(\mathsf{A})$$

Iterative methods viewed as denoising procedures for Random Sketching methods.

Choose $\mathbf{G} \sim \mathcal{N}(0,1)^{d\times k}$. If $\mathbf{A}$ is rank k then:

$$span\,(\mathbf{AG}) = span(\mathbf{A})$$

$$\tilde{\mathbf{U}}_k = span\,(\mathbf{AG}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F = \|\mathbf{A} - \mathbf{A}\|_F = 0$$

Iterative methods viewed as denoising procedures for Random Sketching methods.

Choose $\mathbf{G} \sim \mathcal{N}(0,1)^{d \times k}$. If $\mathbf{A}$ is rank k then:

$$span\,(\mathbf{AG}) = span(\mathbf{A})$$

$$\tilde{\mathbf{U}}_k = span\,(\mathbf{A}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F = \|\mathbf{A} - \mathbf{A}\|_F = 0$$

Iterative methods viewed as denoising procedures for Random Sketching methods.

Choose $\mathbf{G} \sim \mathcal{N}(0,1)^{d\times k}$. If $\mathbf{A}$ is rank k then:

$$span\,(\mathbf{AG}) = span(\mathbf{A})$$

$$\tilde{\mathbf{U}}_k = span\,(\mathbf{A}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F = \|\mathbf{A} - \mathbf{A}\|_F = 0$$

If $\mathbf{A}$ is not rank k then we have error due to $\mathbf{A} - \mathbf{A}_k$:

$$\tilde{\mathbf{U}}_k = span(\mathbf{AG}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F \leq \mathrm{poly}(d)\|\mathbf{A} - \mathbf{A}_k\|_F$$

Iterative methods viewed as denoising procedures for Random Sketching methods.

Choose $\mathbf{G} \sim \mathcal{N}(0,1)^{d\times k}$. If $\mathbf{A}$ is rank k then:

$$span\,(\mathbf{AG}) = span(\mathbf{A})$$

$$\tilde{\mathbf{U}}_k = span\,(\mathbf{A}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F = \|\mathbf{A} - \mathbf{A}\|_F = 0$$

If $\mathbf{A}$ is not rank k then we have error due to $\mathbf{A} - \mathbf{A}_k$:

$$\tilde{\mathbf{U}}_k = span(\mathbf{AG}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F \leq \text{poly}(d)\|\mathbf{A} - \mathbf{A}_k\|_F$$

- Gives an error bound for a single power method iteration.

Iterative methods viewed as denoising procedures for Random Sketching methods.

Choose $\mathbf{G} \sim \mathcal{N}(0,1)^{d\times k}$. If $\mathbf{A}$ is rank k then:

$$span\,(\mathbf{AG}) = span(\mathbf{A})$$

$$\tilde{\mathbf{U}}_k = span\,(\mathbf{A}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F = \|\mathbf{A} - \mathbf{A}\|_F = 0$$

If $\mathbf{A}$ is not rank k then we have error due to $\mathbf{A} - \mathbf{A}_k$:

$$\tilde{\mathbf{U}}_k = span(\mathbf{AG}) \implies \|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F \leq \text{poly}(d)\|\mathbf{A} - \mathbf{A}_k\|_F$$

- Gives an error bound for a single power method iteration.
- Meaningless unless $\|\mathbf{A} - \mathbf{A}_k\|_F$ (the ‘tail noise’) is very small.

How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead.

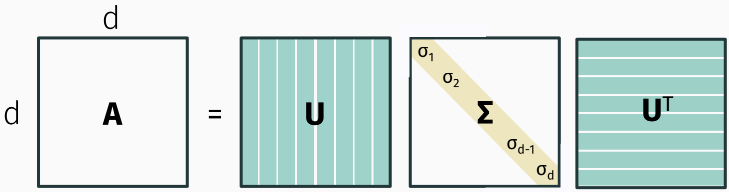
How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead.

This is exactly what Block Power Method does:

$$\mathbf{G} \to \mathbf{A}\mathbf{G} \to \mathbf{A}^2\mathbf{G} \to \ldots \to \mathbf{A}^q\mathbf{G}, \qquad \tilde{\mathbf{U}}_k = span(\mathbf{A}^q\mathbf{G})$$

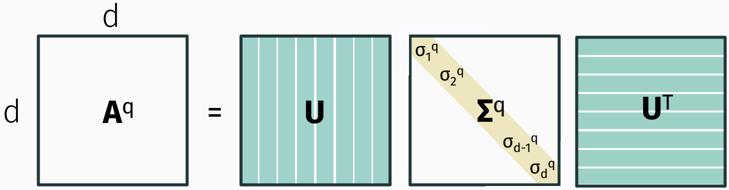
How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead.

Assuming $\mathbf{A}$ is symmetric, if $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T$ then $\mathbf{A}^q = \mathbf{U}\boldsymbol{\Sigma}^q\mathbf{U}^T$.



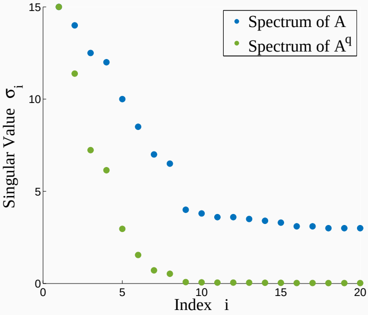
How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead.

Assuming $\mathbf{A}$ is symmetric, if $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ then $\mathbf{A}^q = \mathbf{U}\mathbf{\Sigma}^q\mathbf{U}^T$.



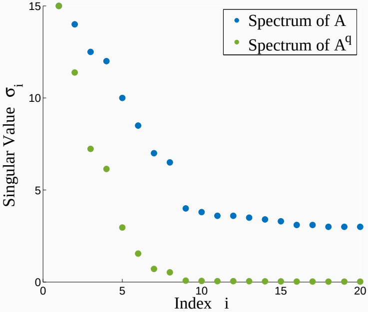
How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead.

Assuming $\mathbf{A}$ is symmetric, if $\mathbf{A} = \mathbf{U\Sigma U}^T$ then $\mathbf{A}^q = \mathbf{U\Sigma}^q\mathbf{U}^T$.



How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead.

Assuming $\mathbf{A}$ is symmetric, if $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ then $\mathbf{A}^q = \mathbf{U}\mathbf{\Sigma}^q\mathbf{U}^T$.



$\|\mathbf{A}^q - \mathbf{A}_k^q\|_F^2 = \sum_{i=k+1}^d \sigma_i^{2q}$ is extremely small.

- $q = \tilde{O}(1/\epsilon)$ ensures that any singular value below σ_{k+1} becomes extremely small in comparison to any singular value above $(1+\epsilon)\sigma_{k+1}$.

- $q = \tilde{O}(1/\epsilon)$ ensures that any singular value below σ_{k+1} becomes extremely small in comparison to any singular value above $(1+\epsilon)\sigma_{k+1}$.

$$(1-\epsilon)^{O(1/\epsilon)} << 1$$

- $q = \tilde{O}(1/\epsilon)$ ensures that any singular value below σ_{k+1} becomes extremely small in comparison to any singular value above $(1+\epsilon)\sigma_{k+1}$.

$$(1-\epsilon)^{O(1/\epsilon)} << 1$$

- $\tilde{\mathbf{U}}_k = span(\mathbf{A}^q\mathbf{G})$ must align well with large (but not the largest!) singular vectors of $\mathbf{A}^q$ to achieve even coarse Frobenius norm error:

$$\|\mathbf{A}^q - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}^q\|_F \leq \text{poly}(d)\|\mathbf{A}^q - \mathbf{A}_k^q\|_F \approx 0$$

- $q = \tilde{O}(1/\epsilon)$ ensures that any singular value below σ_{k+1} becomes extremely small in comparison to any singular value above $(1+\epsilon)\sigma_{k+1}$.

$$(1-\epsilon)^{O(1/\epsilon)} << 1$$

- $\tilde{\mathbf{U}}_k = span(\mathbf{A}^q\mathbf{G})$ must align well with large (but not the largest!) singular vectors of $\mathbf{A}^q$ to achieve even coarse Frobenius norm error:

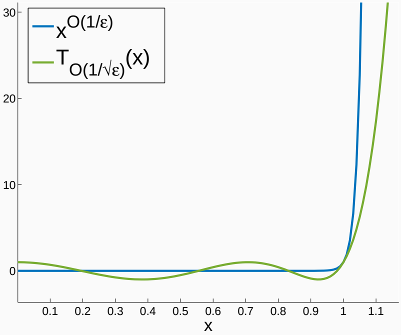
$$\|\mathbf{A}^q - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}^q\|_F \leq \text{poly}(d)\|\mathbf{A}^q - \mathbf{A}_k^q\|_F \approx 0$$

- $\mathbf{A}$ and $\mathbf{A}^q$ have the same singular vectors so $\tilde{\mathbf{U}}_k$ is good for $\mathbf{A}$.

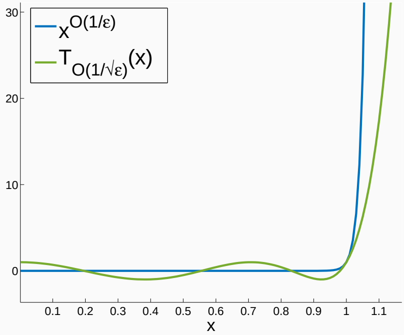
We use new tools for converting **very small** Frobenius norm low-rank approximation error to spectral norm and per vector error, without arguing about convergence of $\tilde{u}_i$ and u_i.

There are better polynomials than A^q for "denoising" A.

There are better polynomials than $\mathbf{A}^q$ for “denoising” $\mathbf{A}$.

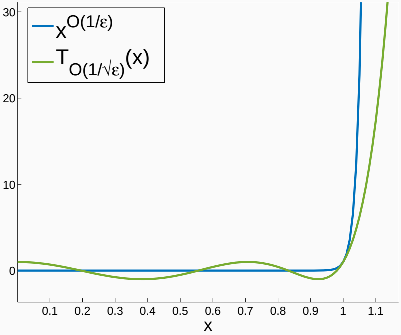


There are better polynomials than $\mathbf{A}^q$ for "denoising" $\mathbf{A}$.



With Chebyshev polynomials only need degree $q = \tilde{O}(1/\sqrt{\epsilon})$.

There are better polynomials than $\mathbf{A}^q$ for "denoising" $\mathbf{A}$.



With Chebyshev polynomials only need degree $q = \tilde{O}(1/\sqrt{\epsilon})$.

- Chebyshev polynomial $T_q(\mathtt{A})$ has a very small tail. So returning $\tilde{\mathtt{U}}_k = span(T_q(\mathtt{A})\mathtt{G})$ would suffice.

- Chebyshev polynomial $T_q(\mathbf{A})$ has a very small tail. So returning $\tilde{\mathbf{U}}_k = span(T_q(\mathbf{A})\mathbf{G})$ would suffice.
- Furthermore, block power iteration computes (at intermediate steps) all of the components needed for:

$$T_q(\mathbf{A})\mathbf{G} = c_0\mathbf{G} + c_1\mathbf{A}\mathbf{G} + \ldots + c_q\mathbf{A}^q\mathbf{G}$$

- Chebyshev polynomial $T_q(\mathbf{A})$ has a very small tail. So returning $\tilde{\mathbf{U}}_k = span(T_q(\mathbf{A})\mathbf{G})$ would suffice.
- Furthermore, block power iteration computes (at intermediate steps) all of the components needed for:

$$T_q(\mathbf{A})\mathbf{G} = c_0\mathbf{G} + c_1\mathbf{A}\mathbf{G} + ... + c_q\mathbf{A}^q\mathbf{G}$$

$$\mathbf{G} \to \mathbf{A}\mathbf{G} \to \mathbf{A}^2\mathbf{G} \to \ldots \to \mathbf{A}^q\mathbf{G}$$

- Chebyshev polynomial $T_q(\mathbf{A})$ has a very small tail. So returning $\tilde{\mathbf{U}}_k = span(T_q(\mathbf{A})\mathbf{G})$ would suffice.
- Furthermore, block power iteration computes (at intermediate steps) all of the components needed for:

$$T_q(\mathbf{A})\mathbf{G} = c_0\mathbf{G} + c_1\mathbf{A}\mathbf{G} + ... + c_q\mathbf{A}^q\mathbf{G}$$

$$\mathbf{G} \rightarrow \mathbf{A}\mathbf{G} \rightarrow \mathbf{A}^2\mathbf{G} \rightarrow \ldots \rightarrow \mathbf{A}^q\mathbf{G}$$

Block Krylov Iteration:

$$\mathcal{K} = \underbrace{\left[\mathbf{G}, \mathbf{A}\mathbf{G}, \mathbf{A}^2\mathbf{G}, \ldots, \mathbf{A}^q\mathbf{G}\right]}_{\text{Krylov subspace}}$$

But...

But... we can't explicitly compute $T_q(\mathbf{A})$, since its parameters depend on $\mathbf{A}$'s (unknown) singular values.

But... we can't explicitly compute $T_q(\mathbf{A})$, since its parameters depend on $\mathbf{A}$'s (unknown) singular values.

Solution: Returning the best $\tilde{\mathbf{U}}_k$ in the span of $\mathcal{K}$ is only better then returning $span(T_q(\mathbf{A})\mathbf{G})$.

What is the best $\tilde{\mathrm{U}}_k$?

What is the best $\tilde{\mathrm{U}}_k$? Surprisingly difficult question.

What is the best $\tilde{\mathbf{U}}_k$? Surprisingly difficult question.

- For Block Power Method, did not need to consider this – $\tilde{\mathbf{U}}_k = span(\mathbf{A}^q\mathbf{G})$ was the only option.

What is the best $\tilde{\mathbf{U}}_k$? Surprisingly difficult question.

- For Block Power Method, did not need to consider this – $\tilde{\mathbf{U}}_k = span(\mathbf{A}^q\mathbf{G})$ was the only option.
- In classical Lanczos/Krylov analysis, convergence to the true singular vectors also lets us avoid this issue. Use Rayleigh Ritz procedure.

- Project $\mathbf{A}$ to $\mathcal{K}$ and take the top k singular vectors (using an accurate classical method):

$$\tilde{\mathbf{U}}_k = span\left((\mathbf{P}_{\mathcal{K}}\mathbf{A})_k\right)$$

- Project $\mathbf{A}$ to $\mathcal{K}$ and take the top k singular vectors (using an accurate classical method):

$$\tilde{\mathbf{U}}_k = span\left((\mathbf{P}_{\mathcal{K}}\mathbf{A})_k\right)$$

Block Krylov Iteration:

$$\mathcal{K} = \underbrace{\left[\mathbf{G}, \mathbf{A}\mathbf{G}, \mathbf{A}^2\mathbf{G}, \ldots, \mathbf{A}^q\mathbf{G}\right]}_{\text{Krylov subspace}}, \; \underbrace{\tilde{\mathbf{U}}_k = span\left((\mathbf{P}_{\mathcal{K}}\mathbf{A})_k\right)}_{\text{'best' solution in Krylov subspace}}$$

- Project $\mathbf{A}$ to $\mathcal{K}$ and take the top k singular vectors (using an accurate classical method):

$$\tilde{\mathbf{U}}_k = span\left((\mathbf{P}_{\mathcal{K}}\mathbf{A})_k\right)$$

Block Krylov Iteration:

$$\mathcal{K} = \underbrace{\left[\mathbf{G}, \mathbf{A}\mathbf{G}, \mathbf{A}^2\mathbf{G}, \ldots, \mathbf{A}^q\mathbf{G}\right]}_{\text{Krylov subspace}}, \; \underbrace{\tilde{\mathbf{U}}_k = span\left((\mathbf{P}_{\mathcal{K}}\mathbf{A})_k\right)}_{\text{'best' solution in Krylov subspace}}$$

- Equivalent to the classic Block Lanczos algorithm in exact arithmetic.

This post-processing step provably gives an optimal $\tilde{U}_k$ for Frobenius norm low-rank approximation error.

This post-processing step provably gives an optimal $\tilde{\mathbf{U}}_k$ for Frobenius norm low-rank approximation error.

- Our entire analysis relied on converting very small Frobenius norm error to strong spectral norm and per vector error!

This post-processing step provably gives an optimal $\tilde{U}_k$ for Frobenius norm low-rank approximation error.

- Our entire analysis relied on converting very small Frobenius norm error to strong spectral norm and per vector error!

Take away: Modern denoising analysis gives new insight into the practical effectiveness of Rayleigh-Ritz projection.

Similar to randomized Block Power Method – extremely simple (pseudocode in paper).

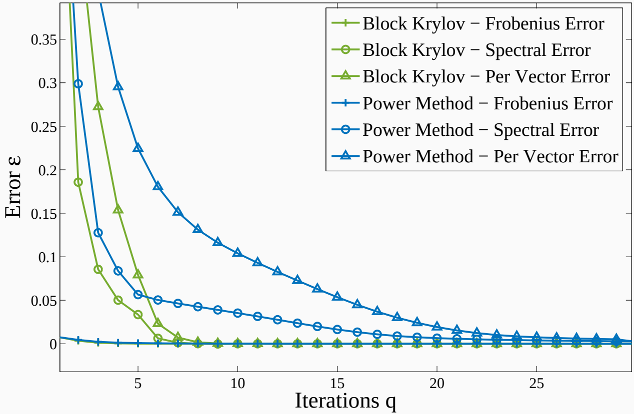
Block Power Method

```
X = randn(d,k);
for i=1:iter
    [X,R] = qr(A*X);
end
U = X;
```

Block Krylov Iteration

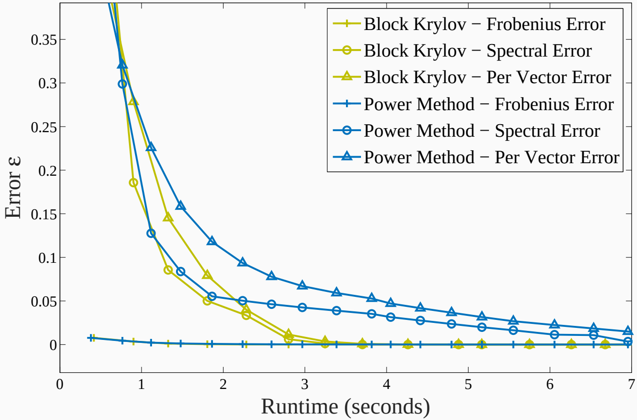
```
X = randn(d,k);
K = zeros(d,k*iter);
for i=1:iter
    [X,R] = qr(A*X);
    K(:,(i-1)*k+1:i*k) = X;
end
[Q,R] = qr(K);
[U,S] = svd(Q'*A,'econ');
U = Q*U(:,1:k);
```

Block Krylov beats Block Power Method definitively for small ϵ.



20 Newsgroups, $k = 20$

Block Krylov beats Block Power Method definitively for small ϵ.



20 Newsgroups, $k = 20$

Main Takeaway: First gap independent bound for Krylov methods.

$$O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{(\sigma_k - \sigma_{k+1}/\sigma_k}}\right) \rightarrow O\left(\text{nnz}(\mathbf{A})k \cdot \frac{\log d}{\sqrt{\epsilon}}\right)$$

Open Questions

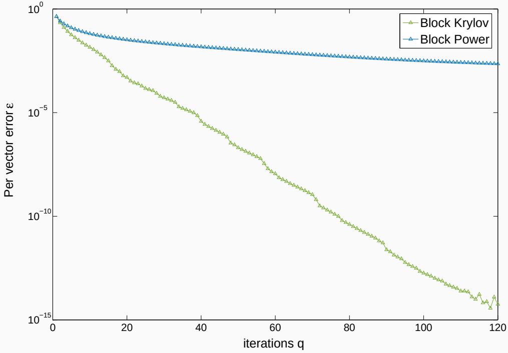
- Full stability analysis.
- 'Master' error metric for gap independent results.
- Gap independent bounds for other methods (e.g. online and stochastic PCA).
- Analysis for small space/restarted block Krylov methods?

Thank you!

Stability

- Lanczos algorithms are often considered to be unstable.
- Largely due to the fact that a recurrence is used to efficiently compute a basis for the Krylov subspace “on the fly”.
- Since our subspace is small, we do not use the recurrence. Computing the basis explicitly avoids serious stability issues.
- There is some loss of orthogonality between blocks. However it only occurs once the algorithm has converged and we can show that it is not an issue in practice.

On poorly conditioned matrices Randomized Block Krylov Iteration still significantly outperforms Block Power Method.



Per Vector Error for $k = 10$, $\kappa = 100{,}000$