FAST LOW-RANK APPROXIMATION AND PCA: BEYOND SKETCHING

Cameron Musco

June 20, 2016

Massachusetts Institute of Technology (currently at IBM Research Almaden)

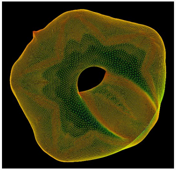
Why study low-rank approximation and PCA?

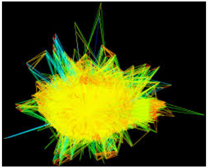
Why study low-rank approximation and PCA? Both basically boil down to singular value decomposition – aren't these solved problems?

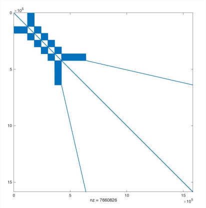
- New matrices (not just larger)

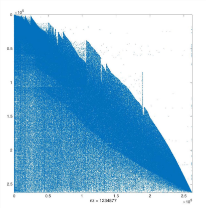
OVERVIEW

- New matrices (not just larger)









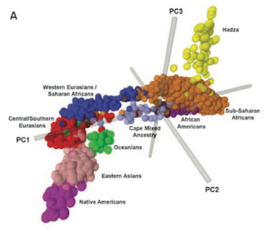
- New matrices (not just larger)
- New parameter regimes (few top principal components vs. full SVD)

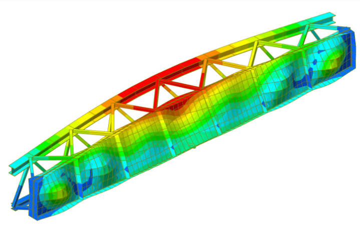
OVERVIEW

- New matrices (not just larger)
- New parameter regimes (few top principal components vs. full SVD)
- New accuracy metrics (driven by new applications)

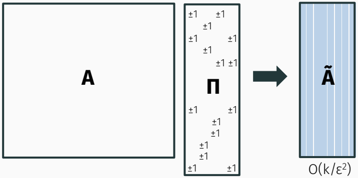
- New matrices (not just larger)
- New parameter regimes (few top principal components vs. full SVD)
- New accuracy metrics (driven by new applications)

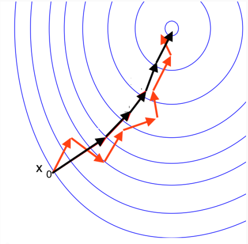
$$\epsilon >> \epsilon_{\text{MACHINE}}$$





- New matrices (not just larger)
- New parameter regimes (few top principal components vs. full SVD)
- New accuracy metrics (driven by new applications)
- New tools (randomized methods)





- Lots of room for cross-fertilization between Numerical Linear Algebra and Machine Learning.

OVERVIEW

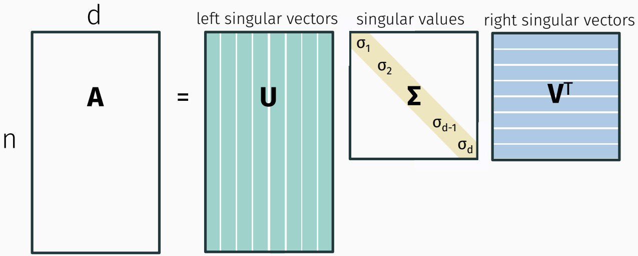
- Lots of room for cross-fertilization between Numerical Linear Algebra and Machine Learning.
- In this talk I will give three examples of this.

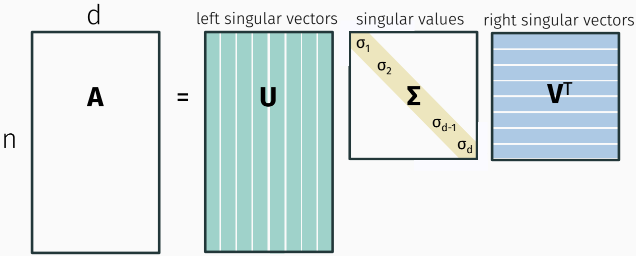
EXAMPLE 1

Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. NIPS 2016. Cameron Musco and Christopher Musco.

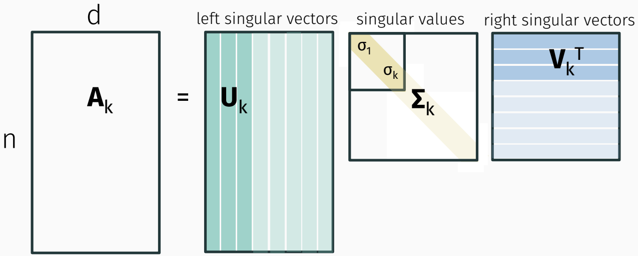
Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. NIPS 2016. Cameron Musco and Christopher Musco.

Random Sketching + Krylov Subspace Methods


d
n
A
=
left singular vectors
U
singular values
σ_1
σ_2
Σ
σ_{d-1}
σ_d
right singular vectors
V^T

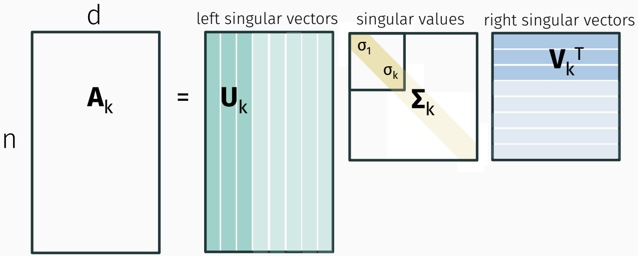


- Key primitive for dimensionality reduction, low-rank approximation, PCA, etc.



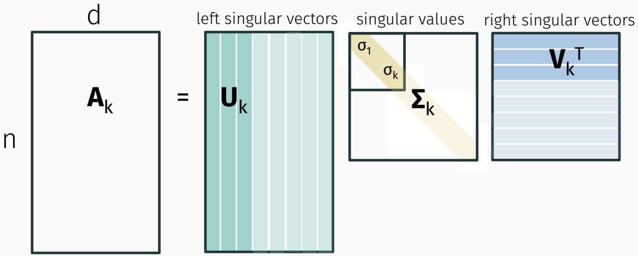
- Key primitive for dimensionality reduction, low-rank approximation, PCA, etc.

$$A_k = \arg\min_{B:rank(B)=k} \|A - B\|_F$$



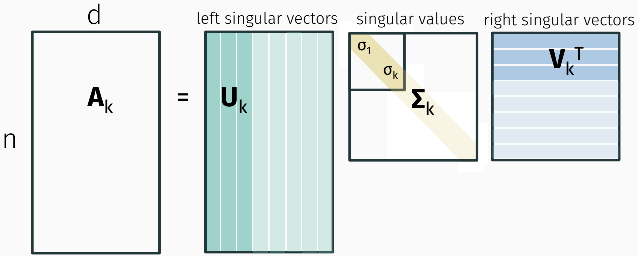
- Key primitive for dimensionality reduction, low-rank approximation, PCA, etc.

$$A_k = \arg\min_{B:rank(B)=k} \|A - B\|_2$$



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

$$U_k U_k^T A = \arg\min_{B: rank(B)=k} \|A - B\|_2$$



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

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

- Full SVD requires roughly $O(nd^2)$ time – much too slow.

Traditional Solution: Iterative methods

Compute just k top singular vectors roughly in time:

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

Traditional Solution: Iterative methods

Compute just k top singular vectors roughly in time:

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

Traditional Solution: Iterative methods

Compute just k top singular vectors roughly in time:

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

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

Traditional Solution: Iterative methods

Traditional Solution: Iterative methods

- Typical accuracy guarantees of the form

$$\|\tilde{u}_i - u_i\|_2 \leq \epsilon.$$

Traditional Solution: Iterative methods

- Typical accuracy guarantees of the form

$$\|\tilde{u}_i - u_i\|_2 \leq \epsilon.$$

- Runtime for block power method:

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

Traditional Solution: Iterative methods

- Typical accuracy guarantees of the form

$$\|\tilde{u}_i - u_i\|_2 \leq \epsilon.$$

- Runtime for block power method:

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

Traditional Solution: Iterative methods

- Typical accuracy guarantees of the form

$$\|\tilde{u}_i - u_i\|_2 \leq \epsilon.$$

- Runtime for block power method:

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

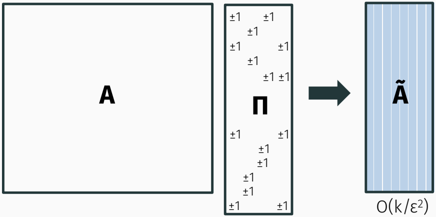
- Often the dominant factor in runtime bound.

Modern Solution: Sketching Methods

Modern Solution: Sketching Methods

- Sparse Subspace Embeddings [Clarkson, Woodruff 2013]:

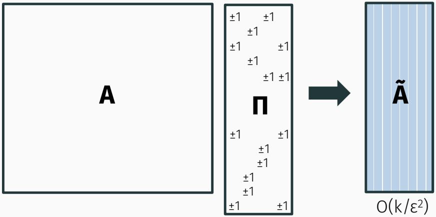
$$\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F \text{ in time } O(\text{nnz}(\mathbf{A})) + \tilde{O}\left(\frac{nk^2}{\epsilon^4}\right)$$



Modern Solution: Sketching Methods

- Sparse Subspace Embeddings [Clarkson, Woodruff 2013]:

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



FROBENIUS NORM LOW-RANK APPROXIMATION

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

FROBENIUS NORM LOW-RANK APPROXIMATION

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

- Very different from classic $\|u_i - \tilde{u}_i\| \leq \epsilon$ guarantee.

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

- Very different from classic $\|u_i - \tilde{u}_i\| \leq \epsilon$ guarantee.
- Still sufficient for many tasks (e.g. dimensionality reduction for clustering)

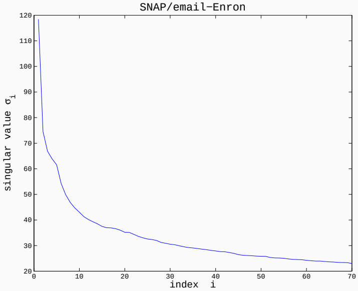
$$\|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^T\mathbf{A}\|_F$$

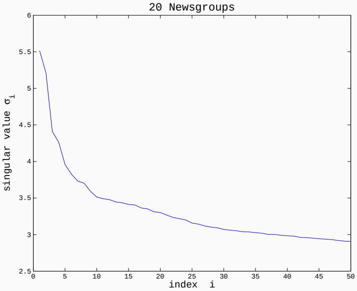
- Very different from classic $\|u_i - \tilde{u}_i\| \leq \epsilon$ guarantee.
- Still sufficient for many tasks (e.g. dimensionality reduction for clustering)
- But can be weak.

FROBENIUS NORM LOW-RANK APPROXIMATION

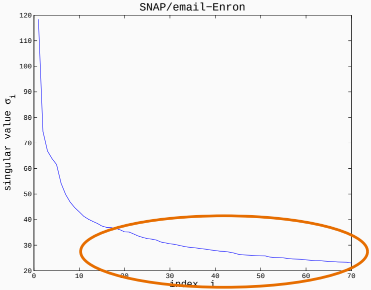
$$\|\mathbf{A} - \mathbf{U}_k \mathbf{U}_k^T \mathbf{A}\|_F^2 = \|\mathbf{A} - \mathbf{A}_k\|_F^2 = \sum_{i=k+1}^{d} \sigma_i^2$$

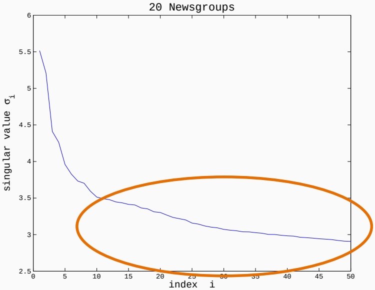
$$\|A - U_k U_k^T A\|_F^2 = \|A - A_k\|_F^2 = \sum_{i=k+1}^{d} \sigma_i^2$$



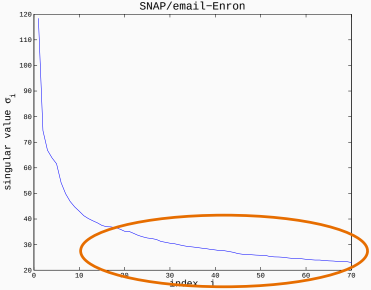


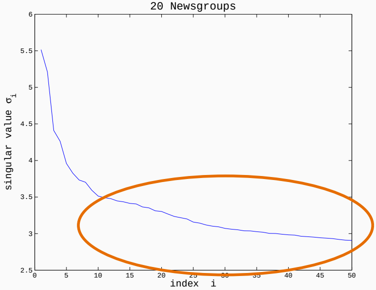
$$\|A - U_k U_k^T A\|_F^2 = \|A - A_k\|_F^2 = \sum_{i=k+1}^{d} \sigma_i^2$$





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





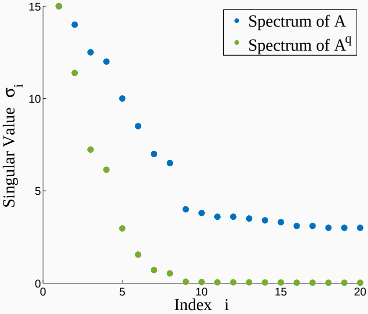
Often $\epsilon\|\mathbf{A} - \mathbf{U}_k\mathbf{U}_k^\top\mathbf{A}\|_F^2$ is bigger than even $\mathbf{A}$'s largest singular value and so guarantee isn't meaningful. Literally any $\tilde{\mathbf{U}}_k$ would work!

How to avoid tail noise?

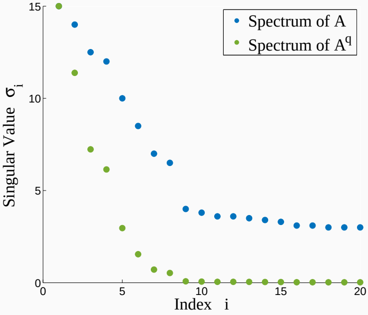
How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead.

How to avoid tail noise? Apply sketching method to $\mathbf{A}^q$ instead. 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 $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T$ 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 $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T$ then $\mathbf{A}^q = \mathbf{U}\boldsymbol{\Sigma}^q\mathbf{U}^T$.



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

- This is exactly what Block Power Method does!

$$\mathbf{\Pi} \to \mathbf{A}\mathbf{\Pi} \to \mathbf{A}^2\mathbf{\Pi} \to \ldots \to \mathbf{A}^q\mathbf{\Pi}.$$

- This is exactly what Block Power Method does!

$$\mathbf{\Pi} \rightarrow \mathbf{A}\mathbf{\Pi} \rightarrow \mathbf{A}^2\mathbf{\Pi} \rightarrow \ldots \rightarrow \mathbf{A}^q\mathbf{\Pi}.$$

- 'Denoising' analysis gives new 'gap-independent' bounds for block power method (with randomized start vectors):

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

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 is widely cited and implemented – simple algorithm with simple bounds.

redSVD

scikit learn

IBM

ScalaNLP (Breeze)

libSkylark

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

redSVD

scikit learn

IBM

ScalaNLP (Breeze)

libSkylark

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

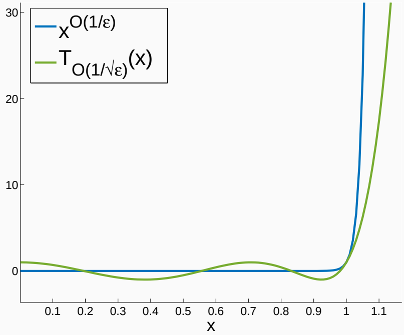
RANDOMIZED BLOCK POWER METHOD

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

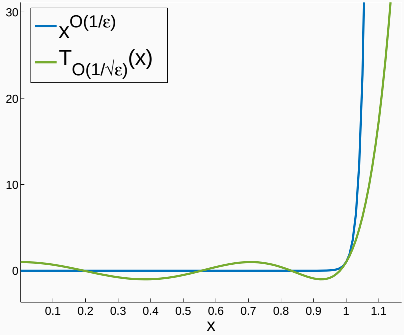
But in the numerical linear algebra community, Krylov/Lanczos methods have long been prefered over power iteration.

Key Idea: Better polynomials than A^q for "denoising" A.

Key Idea: Better polynomials than $\mathbf{A}^q$ for "denoising" $\mathbf{A}$.

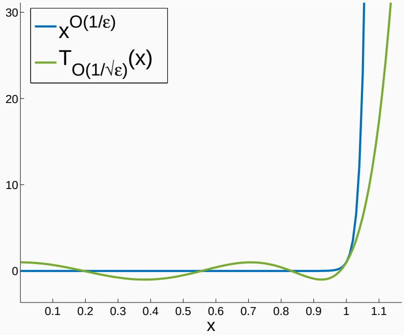


Key Idea: Better polynomials than $\mathbf{A}^q$ for "denoising" $\mathbf{A}$.



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

Key Idea: 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.

- Chebyshev polynomial $T_q(\mathbf{A})$ has a very small tail.
- But we can't compute it explicitly – parameters depend on $\mathbf{A}$'s (unknown) singular values.

- Chebyshev polynomial $T_q(\mathsf{A})$ has a very small tail.
- But we can't compute it explicitly – parameters depend on A's (unknown) singular values.

Traditional Solution: Produce a Krylov Subspace:

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

- Chebyshev polynomial $T_q(\mathsf{A})$ has a very small tail.
- But we can't compute it explicitly – parameters depend on A's (unknown) singular values.

Traditional Solution: Produce a Krylov Subspace:

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

Best solution in the span of $\mathcal{K}$ is only better than $T_q(\mathsf{A})\mathbf{\Pi}$.

What is the best solution?

What is the best solution? Traditionally, use Rayleigh-Ritz method:

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

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

What is the best solution? Traditionally, use Rayleigh-Ritz method:

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

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

- But classic Lanczos/Krylov analysis requires convergence to the true singular vectors to show the effectiveness of Rayleigh-Ritz.

OUR SOLUTION

- Rayleigh-Ritz method gives provably optimal $\tilde{\mathbf{U}}_k$ for Frobenius norm low-rank approximation error.

OUR SOLUTION

- Rayleigh-Ritz method gives provably optimal $\tilde{\mathbf{U}}_k$ for Frobenius norm low-rank approximation error.
- Our entire analysis relies on converting very small Frobenius norm error to stronger spectral norm error!

OUR SOLUTION

- Rayleigh-Ritz method gives provably optimal $\tilde{\mathbf{U}}_k$ for Frobenius norm low-rank approximation error.
- Our entire analysis relies on converting very small Frobenius norm error to stronger spectral norm error!

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

Main Takeaway: First gap independent bound for Krylov methods. $\|\mathbf{A} - \tilde{\mathbf{U}}_k \tilde{\mathbf{U}}_k^T \mathbf{A}\|_2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_2$.

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

Main Takeaway: First gap independent bound for Krylov methods. $\|\mathbf{A} - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^T\mathbf{A}\|_2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_2$.

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

Open Questions

- Full stability analysis – similar to power method analysis in [Hardt, Price 2014], [Balcan, Du, Wang, Yu 2016]
- 'Master' potential function for gap independent results.
- Analysis for small space/restarted block Krylov methods?
- $O(\text{nnz}(\mathbf{A}) + \text{poly}(k, \epsilon))$ time for spectral norm error?

EXAMPLE 2

Faster Eigenvector Computation via Shift-and-Invert Preconditioning. ICML 2016. Garber, Hazan, Jin, Kakade, Musco, Netrapalli, and Sidford.

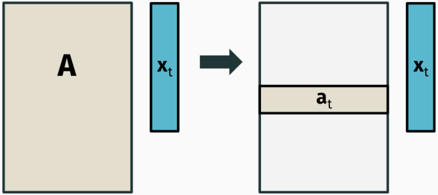
EXAMPLE 2

Faster Eigenvector Computation via Shift-and-Invert Preconditioning. ICML 2016. Garber, Hazan, Jin, Kakade, Musco, Netrapalli, and Sidford.

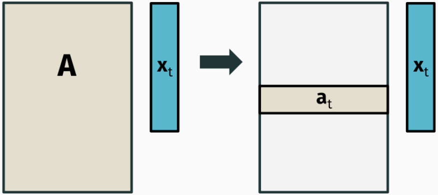
Stochastic Gradient Descent + Inverse Iteration

Key Idea: Accelerate iterative methods by replacing full matrix multiplications with single row updates.

Key Idea: Accelerate iterative methods by replacing full matrix multiplications with single row updates. Per iteration cost $\text{nnz}(\mathbf{A}) \to d$.



Key Idea: Accelerate iterative methods by replacing full matrix multiplications with single row updates. Per iteration cost $\text{nnz}(\mathbf{A}) \rightarrow d$.



- Implementable in streaming setting using just $O(d)$ space.

- Lots of recent success: [Shamir 2015, 2016], [Sa, Ré, Olukotun 2015], [Jain, Jin, Kakade, Netrapalli, Sidford 2016]

- Lots of recent success: [Shamir 2015, 2016], [Sa, Ré, Olukotun 2015], [Jain, Jin, Kakade, Netrapalli, Sidford 2016]
- Analyze stochastic convex optimization methods applied to *non-convex* top singular vector problem.

- Lots of recent success: [Shamir 2015, 2016], [Sa, Ré, Olukotun 2015], [Jain, Jin, Kakade, Netrapalli, Sidford 2016]
- Analyze stochastic convex optimization methods applied to *non-convex* top singular vector problem.
- **Alternative idea:** reduce singular vector computation to most well-studied convex problem, linear system solving.

- Lots of recent success: [Shamir 2015, 2016], [Sa, Ré, Olukotun 2015], [Jain, Jin, Kakade, Netrapalli, Sidford 2016]
- Analyze stochastic convex optimization methods applied to *non-convex* top singular vector problem.
- **Alternative idea:** reduce singular vector computation to most well-studied convex problem, linear system solving.

Shift-and-Invert Preconditioning

- **Key Idea:** Power Method on $(\sigma \mathsf{I} - \mathsf{A})^{-1}$ converges extremely quickly when $\sigma \approx \sigma_1(\mathsf{A})$.

$$\sigma_1\left((\sigma \mathsf{I} - \mathsf{A})^{-1}\right) >> \sigma_2\left((\sigma \mathsf{I} - \mathsf{A})^{-1}\right).$$

SHIFT-AND-INVERT PRECONDITIONING WITH STOCHASTIC METHODS

- **Key Idea:** Power Method on $(\sigma \mathsf{I} - \mathsf{A})^{-1}$ converges extremely quickly when $\sigma \approx \sigma_1(\mathsf{A})$.

$$\sigma_1\left((\sigma \mathsf{I} - \mathsf{A})^{-1}\right) >> \sigma_2\left((\sigma \mathsf{I} - \mathsf{A})^{-1}\right).$$

- We can apply stochastic system solvers black box (almost) to accelerate iterations and implement them in streaming/online setting.

- **Key Idea:** Power Method on $(\sigma \mathbf{I} - \mathbf{A})^{-1}$ converges extremely quickly when $\sigma \approx \sigma_1(\mathbf{A})$.

$$\sigma_1\left((\sigma \mathbf{I} - \mathbf{A})^{-1}\right) >> \sigma_2\left((\sigma \mathbf{I} - \mathbf{A})^{-1}\right).$$

- We can apply stochastic system solvers black box (almost) to accelerate iterations and implement them in streaming/online setting.
- Give a significantly more robust analysis of shift-and-invert preconditioning, which handles approximate solvers.

$$\tilde{O}\left(\mathrm{nnz}(\mathbf{A}) \cdot \frac{1}{\sqrt{\mathrm{gap}}}\right) \rightarrow \tilde{O}\left(\mathrm{nnz}(\mathbf{A}) + \frac{d^2}{\mathrm{gap}^2}\right)$$

Principal Component Projection Without Principal Component Analysis. ICML 2016. Roy Frostig, Cameron Musco, Christopher Musco, Aaron Sidford.

EXAMPLE 3

Principal Component Projection Without Principal Component Analysis. ICML 2016. Roy Frostig, Cameron Musco, Christopher Musco, Aaron Sidford.

Regularized Regression + Polynomial Approximation

Instead of returning $\mathbf{U}_k$ we often just want to compute $\mathbf{U}_k\mathbf{U}_k^T\mathbf{y}$ for some input vector.

Instead of returning $\mathbf{U}_k$ we often just want to compute $\mathbf{U}_k\mathbf{U}_k^T\mathbf{y}$ for some input vector.

- Useful in many applications like principal component regression (PCR).

Instead of returning $\mathbf{U}_k$ we often just want to compute $\mathbf{U}_k\mathbf{U}_k^T\mathbf{y}$ for some input vector.

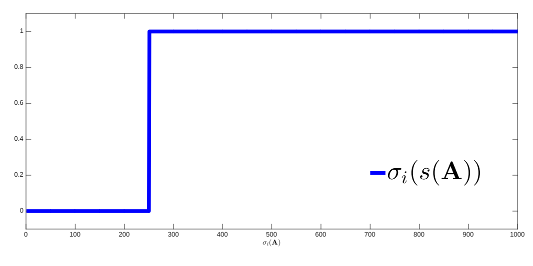
- Useful in many applications like principal component regression (PCR).
- It's very often more efficient to apply a matrix function once than compute it explicitly.

Instead of returning $\mathbf{U}_k$ we often just want to compute $\mathbf{U}_k\mathbf{U}_k^T\mathbf{y}$ for some input vector.

- Useful in many applications like principal component regression (PCR).
- It's very often more efficient to apply a matrix function once than compute it explicitly.
- $\mathbf{A}^q\mathbf{x}$, $\mathbf{A}^{-1}\mathbf{x}$, $\exp(\mathbf{A})$ ... many more.

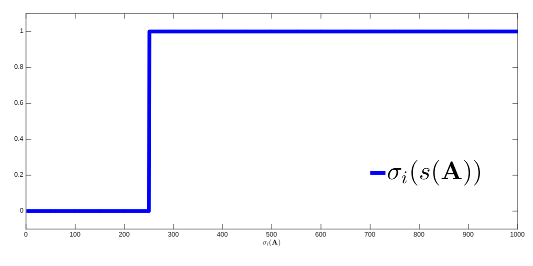
- For symmetric $\mathbf{A}$, $\mathbf{U}_k\mathbf{U}_k^T\mathbf{y} = s(\mathbf{A})\mathbf{y} = \mathbf{U}s(\boldsymbol{\Sigma})\mathbf{U}^T\mathbf{y}$ where $s(x) = 0$ for $x \leq \sigma_k$ and $s(x) = 1$ for $x \geq \sigma_k$.

- For symmetric $\mathbf{A}$, $\mathbf{U}_k\mathbf{U}_k^T\mathbf{y} = s(\mathbf{A})\mathbf{y} = \mathbf{U}s(\mathbf{\Sigma})\mathbf{U}^T\mathbf{y}$ where $s(x) = 0$ for $x \leq \sigma_k$ and $s(x) = 1$ for $x \geq \sigma_k$.



STEP FUNCTION APPROXIMATION

- For symmetric $\mathbf{A}$, $\mathbf{U}_k\mathbf{U}_k^T\mathbf{y} = s(\mathbf{A})\mathbf{y} = \mathbf{U}s(\mathbf{\Sigma})\mathbf{U}^T\mathbf{y}$ where $s(x) = 0$ for $x \leq \sigma_k$ and $s(x) = 1$ for $x \geq \sigma_k$.



- **Our Method:** Coarsely approximate the step function using ridge regression.

Main Observation: The step function removes small principal components. Ridge regression dampens them.

Main Observation: The step function removes small principal components. Ridge regression dampens them.

$$(\mathbf{A} + \sigma_k \mathbf{I})^{-1}\mathbf{A}\mathbf{y} \approx s(\mathbf{A})\mathbf{y}.$$

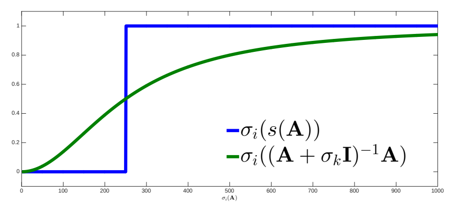
Main Observation: The step function removes small principal components. Ridge regression dampens them.

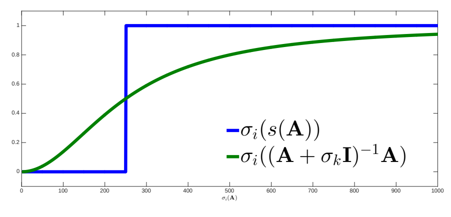
$$(\mathbf{A} + \sigma_k \mathbf{I})^{-1}\mathbf{A}\mathbf{y} \approx s(\mathbf{A})\mathbf{y}.$$

Main Observation: The step function removes small principal components. Ridge regression dampens them.

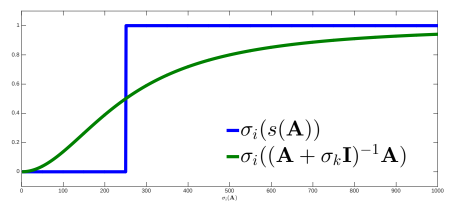
$$(\mathbf{A} + \sigma_k \mathbf{I})^{-1}\mathbf{A}\mathbf{y} \approx s(\mathbf{A})\mathbf{y}.$$

$$\frac{x}{x + \sigma_k} \approx \begin{cases} 0 \text{ for } x << \sigma_k \\ 1 \text{ for } x >> \sigma_k \end{cases}$$





- Sharpen this coarse approximation using a low-degree polynomial approximation to a symmetric step function



- Sharpen this coarse approximation using a low-degree polynomial approximation to a symmetric step function
- Symmetric step/sign function approximation is well-studied in numerical analysis, but again we give a significantly more robust analysis.

Direct method for principal component projection that doesn't require computing the top singular vectors of $\mathbf{A}$.

Direct method for principal component projection that doesn't require computing the top singular vectors of $\mathbf{A}$.

- Faster PCA by not doing PCA at all.

Thank you!

(And thanks to my collaborators!)