Finding Structure with Randomness

Joel A. Tropp

Computing + Mathematical Sciences
California Institute of Technology
jtropp@cms.caltech.edu

Thanks: Alex Gittens (eBay), Michael Mahoney (Berkeley Stat), Gunnar Martinsson (Boulder Appl. Math), Mark Tygert (Facebook)

Research supported by ONR, AFOSR, NSF, DARPA, Sloan, and Moore.

Primary Sources for Tutorial

- T. *User-Friendly Tools for Random Matrices: An Introduction*. Submitted to *FnTML*, 2014.

  tinyurl.com/pobvezn

- Halko, Martinsson, and T. "Finding structure with randomness..." *SIAM Rev.*, 2011.

  tinyurl.com/p5b5uw6

[Refs] `http://users.cms.caltech.edu/~jtropp/notes/Tro14-User-Friendly-Tools-FnTML-draft.pdf`
`http://users.cms.caltech.edu/~jtropp/papers/HMT11-Finding-Structure-SIREV.pdf`

Download the slides:

tinyurl.com/nbq2erb

[Ref] `http://users.cms.caltech.edu/~jtropp/slides/Tro14-Finding-Structure-ICML.pdf`

Matrix Decompositions & Approximations

Top 10 Scientific Algorithms

list (here, the list is in chronological order; however, the articles appear in no particular order):

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

With each of these algorithms or approaches, there is a person or group receiving credit for inventing or

[Ref] Dongarra and Sullivan 2000.

The Decompositional Approach

"The underlying principle of the decompositional approach to matrix computation is that it is not the business of the matrix algorithmicists to solve particular problems but to construct computational platforms from which a variety of problems can be solved."

- A decomposition solves not one but many problems
- Often expensive to compute but can be reused
- Shows that apparently different algorithms produce the same object
- Facilitates rounding-error analysis
- Can be updated efficiently to reflect new information
- Has led to highly effective black-box software

[Ref] Stewart 2000.

Matrix Approximations

"Matrix nearness problems arise in many areas... A common situation is where a matrix $\boldsymbol{A}$ approximates a matrix $\boldsymbol{B}$ and $\boldsymbol{B}$ is known to possess a property **P**... An intuitively appealing way of improving $\boldsymbol{A}$ is to replace it by a nearest matrix $\boldsymbol{X}$ with property **P**.

"Conversely, in some applications it is important that $\boldsymbol{A}$ does not have a certain property **P** and it useful to know how close $\boldsymbol{A}$ is to having the undesirable property."

- Approximations can purge an undesirable property (ill-conditioning)
- Can enforce a property the matrix lacks (sparsity, low rank)
- Can identify structure in a matrix
- Perform regularization, denoising, compression, ...

[Ref] Higham 1989; Dhillon & T 2006.

What's Wrong with Classical Approaches?

- Nothing... when the matrices are small and fit in core memory
- **Challenges:**
  - Medium- to large-scale data (Megabytes+)
  - New architectures (multi-core, distributed, data centers, ...)
- **Why Randomness?**
  - It works...
  - Randomized approximations can be very effective
  - Leads to multiplication-rich algorithms (low communication costs; highly optimized primitives)

Hour 1: Approximation via Random Sampling

- **Goal:** Find a structured approximation to a given matrix
- **Approach:**
  - Construct a simple unbiased estimator of the matrix
  - Average independent copies to reduce variance
- **Examples:**
  - Matrix sparsification
  - Random features
- **Analysis:** Matrix Bernstein inequality
- **Shortcomings:** Low precision; no optimality properties

Hour 2: Two-Stage Randomized Algorithms

- **Goal:** Construct near-optimal, low-rank matrix approximations
- **Approach:**
  - Use randomness to find a subspace that captures most of the action
  - Compress the matrix to this subspace, and apply classical NLA
- **Randomized Range Finder:**
  - Multiply random test vectors into the target matrix and orthogonalize
  - Apply several steps of subspace iteration to improve precision
- **Some Low-Rank Matrix Decompositions:**
  - Truncated singular value decomposition
  - Interpolative approximations, matrix skeleton, and CUR
  - Nyström approximations for psd matrices

Some Wisdom* from Scientific Computing

**"Who cares about the optimality of an approximation?
Who cares if I solve a specified computational problem?
My algorithm does great on the test set."** **—Nemo**

- **Optimality.** If your approximation is suboptimal, you could do better.
- **Validation.** If your algorithm does not fit the model reliably, you cannot attribute success to *either* the model *or* the algorithm.
- **Verification.** If your algorithm does not solve a specified problem, you cannot easily check whether it has bugs.
- **Modularity.** To build a large system, you want each component to solve a specified problem under specified conditions.
- **Reproducibility.** To use an approach for a different problem, you need the method to have consistent behavior.

Approximation by Random Sampling

Matrix Approximation via Sampling

- Let $\boldsymbol{A}$ be a fixed matrix we want to approximate
- Represent $\boldsymbol{A} = \sum_i \boldsymbol{A}_i$ as a sum (or integral) of simple matrices
- Construct a simple random matrix $\boldsymbol{Z}$ by sampling terms; e.g.,

$$\boldsymbol{Z} = p_i^{-1} \boldsymbol{A}_i \quad \text{with probability } p_i$$

- Ensures that $\boldsymbol{Z}$ is an unbiased estimator: $\mathbb{E}\, \boldsymbol{Z} = \boldsymbol{A}$
- Average independent copies to reduce variance: $\widehat{\boldsymbol{A}} = \frac{1}{\mathrm{r}} \sum_{r=1}^{\mathrm{r}} \boldsymbol{Z}_r$
- **Examples? Analysis?**

[Refs] Maurey 1970s; Carl 1985; Barron 1993; Rudelson 1999; Achlioptas & McSherry 2002, 2007; Drineas et al. 2006; Rudelson & Vershynin 2007; Rahimi & Recht 2007, 2008; Shalev-Shwartz & Srebro 2008; ...

The Matrix Bernstein Inequality

Theorem 1. [T 2012] **Assume**

- $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \ldots$ *are indep. random matrices with dimension* $m \times n$
- $\mathbb{E}\,\boldsymbol{X}_r = \mathbf{0}$ *and* $\|\boldsymbol{X}_r\| \leq L$ *for each index* r
- *Compute the variance measure*

$$v := \max\left\{ \left\| \sum\nolimits_r \mathbb{E}[\boldsymbol{X}_r \boldsymbol{X}_r^*] \right\|, \ \left\| \sum\nolimits_r \mathbb{E}[\boldsymbol{X}_r^* \boldsymbol{X}_r] \right\| \right\}$$

Then

$$\mathbb{P}\left\{ \left\| \sum\nolimits_r \boldsymbol{X}_r \right\| \geq t \right\} \leq d \cdot \exp\left\{ \frac{-t^2/2}{v + Lt/3} \right\}$$

where $d := m + n$.

$\|\cdot\|$ = spectral norm; * = conjugate transpose

[Refs] Oliveira 2009–2011; T 2010–2014. This version from T 2014, *User-Friendly Tools*, Chap. 6.

The Matrix Bernstein Inequality

Theorem 2. [T 2014] **Assume**

- $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \dots$ *are indep. random matrices with dimension* $m \times n$
- $\mathbb{E}\, \boldsymbol{X}_r = \mathbf{0}$ *and* $\|\boldsymbol{X}_r\| \le L$ *for each index* r
- *Compute the variance measure*

$$v := \max\left\{ \left\| \sum\nolimits_r \mathbb{E}[\boldsymbol{X}_r \boldsymbol{X}_r^*] \right\|, \ \left\| \sum\nolimits_r \mathbb{E}[\boldsymbol{X}_r^* \boldsymbol{X}_r] \right\| \right\}$$

Then

$$\mathbb{E} \left\| \sum\nolimits_r \boldsymbol{X}_r \right\| \le \sqrt{2v \log d} + \tfrac{1}{3} L \log d$$

where $d := m + n$.

$\|\cdot\|$ = spectral norm; * = conjugate transpose

[Refs] Chen et al. 2012; Mackey et al. 2014. This version from T 2014, *User-Friendly Tools*, Chap. 6.

Short History of Matrix Bernstein Inequality

Operator Khintchine and Noncommutative Martingales

- **Tomczak-Jaegermann 1974.** First operator Khintchine inequality; suboptimal variance.
- **Lust-Piquard 1986.** Operator Khintchine; optimal variance; suboptimal constants.
- Lust-Piquard & Pisier 1991. Operator Khintchine for trace class.
- Pisier & Xu 1997. Initiates study of noncommutative martingales.
- **Rudelson 1999.** First use of operator Khintchine for random matrix theory.
- Buchholz 2001, 2005. Optimal constants for operator Khintchine.
- Many more works in 2000s.

Matrix Concentration Inequalities

- **Ahlswede & Winter 2002.** Matrix Chernoff inequalities; suboptimal variance.
- Christofides & Markström 2007. Matrix Hoeffding; suboptimal variance.
- Gross 2011; Recht 2011. Matrix Bernstein; suboptimal variance.
- **Oliveira 2011; T 2012.** Matrix Bernstein; optimal variance. Independent works!
- Chen et al. 2012; Mackey et al. 2014; T 2014. Expectation form of matrix inequalities.
- Hsu et al. 2012. Intrinsic dimension bounds; suboptimal form.
- **Minsker 2012.** Intrinsic dimension bounds; optimal form.
- T 2014. Simplified proof of intrinsic dimension bounds.
- **Mackey et al. 2014.** New proofs and results via exchangeable pairs.

[Ref] See T 2014, *User-Friendly Tools* for more historical information.

Error Estimate for Matrix Sampling

Corollary 3. [Matrix Sampling Estimator] **Assume**

- $\boldsymbol{A}$ *is a fixed* $m \times n$ *matrix and* $d := m + n$
- $\boldsymbol{Z}$ *is a random matrix with* $\mathbb{E}\,\boldsymbol{Z} = \boldsymbol{A}$ *and* $\|\boldsymbol{Z}\| \leq L$
- $\boldsymbol{Z}_1, \dots, \boldsymbol{Z}_{\mathtt{r}}$ *are iid copies of* $\boldsymbol{Z}$
- *Compute the per-sample variance*

$$v := \max\left\{ \|\mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^*]\|\,,\ \|\mathbb{E}[\boldsymbol{Z}^*\boldsymbol{Z}]\| \right\}$$

Then *the estimator* $\widehat{\boldsymbol{A}} = \frac{1}{\mathtt{r}} \sum_{r=1}^{\mathtt{r}} \boldsymbol{Z}_r$ *satisfies*

$$\mathbb{E}\,\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\| \leq \sqrt{\frac{2v \log d}{\mathtt{r}}} + \frac{2L \log d}{3\mathtt{r}}$$

[Refs] This version is new. Variants appear in Rudelson 1999; Ahlswede & Winter 2002; Gross 2011; Recht 2011. See T 2014, *User-Friendly Tools*, Chap. 1.

Comments on Matrix Sampling Estimators

- Constructing simple random matrix $\boldsymbol{Z}$ requires **insight + cleverness**
- Approximation $\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|$ in **spectral norm** controls
  - All linear functions of the approximation (marginals)
  - All singular values and singular vectors
  - **Warning:** Frobenius-norm error bounds are usually vacuous!
- Sampling estimators are typically low precision
  - **Bottleneck:** Central Limit Theorem
  - Cost of reducing error is exorbitant (ε^{-2})
  - Error is generally not comparable with best possible approximation

[Ref] Tygert 2014.

Matrix Sparsification

Matrix Sparsification

- **Challenge:** Can we replace a dense matrix by a sparse proxy?

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \quad \rightsquigarrow \quad \begin{bmatrix} x & & x & x & x \\ & x & x & & x \\ x & & x & x & \\ & x & & & x \\ x & & x & x & \end{bmatrix}$$

- **Idea:** Achlioptas & McSherry (2002, 2007) propose random sampling

- **Goals:**
  - Accelerate spectral computations via Krylov methods
  - Compression, denoising, regularization, etc.

[Refs] Achlioptas & McSherry 2002, 2007; Arora et al. 2005; Gittens & Tropp 2009; d'Aspremont 2009; Drineas & Zouzias 2011; Achlioptas et al. 2013; Drineas & Kundu 2014. See T 2014, *User-Friendly Tools*, Chap. 6.

Writing a Matrix as a Sparse Sum

$$\boldsymbol{A} = \sum_{ij} a_{ij} \, \mathbf{E}_{ij}$$

a_{ij} denotes the (i, j) entry of $\boldsymbol{A}$

$\mathbf{E}_{ij}$ is the standard basis matrix with a one in the (i, j) position and zeroes elsewhere

Sparsification by Random Sampling

- Let $\boldsymbol{A}$ be a fixed $n \times n$ matrix
- Define sampling probabilities

$$p_{ij} = \frac{1}{2}\left[\frac{|a_{ij}|^2}{\|\boldsymbol{A}\|_{\mathrm{F}}^2} + \frac{|a_{ij}|}{\|\boldsymbol{A}\|_{\ell_1}}\right] \quad \text{for } i, j = 1, \ldots, n$$

- Let $\boldsymbol{Z} = p_{ij}^{-1} \cdot a_{ij} \cdot \mathbf{E}_{ij}$ with probability p_{ij}
- Let $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_{\mathtt{r}}$ be iid copies of the estimator $\boldsymbol{Z}$
- Thus, $\widehat{\boldsymbol{A}} = \frac{1}{\mathtt{r}} \sum_{r=1}^{\mathtt{r}} \boldsymbol{Z}_r$ is an $\mathtt{r}$-sparse unbiased estimator of $\boldsymbol{A}$

$\|\cdot\|_{\mathrm{F}}$ = Frobenius norm; $\|\cdot\|_{\ell_1}$ = elementwise ℓ_1 norm

[Refs] Kundu & Drineas 2014. T 2014, *User-Friendly Tools*, Chap. 6.

Analysis of Sparsification

- Recall: $\boldsymbol{Z} = p_{ij}^{-1} \cdot a_{ij} \cdot \mathbf{E}_{ij}$ with probability p_{ij}
- Observe: $p_{ij} \geq \dfrac{|a_{ij}|}{2 \|\boldsymbol{A}\|_{\ell_1}}$ and $p_{ij} \geq \dfrac{|a_{ij}|^2}{2 \|\boldsymbol{A}\|_{\mathrm{F}}^2}$
- Uniform bound: $\|\boldsymbol{Z}\| \leq \max_{ij} \dfrac{|a_{ij}|}{p_{ij}} \cdot \|\mathbf{E}_{ij}\| \leq 2 \|\boldsymbol{A}\|_{\ell_1}$
- Variance computation:

$$\mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^*] = \sum\nolimits_{ij} \frac{|a_{ij}|^2}{p_{ij}^2} \cdot \mathbf{E}_{ij}\mathbf{E}_{ij}^* \cdot p_{ij} \preccurlyeq \sum\nolimits_{ij} 2 \|\boldsymbol{A}\|_{\mathrm{F}}^2 \cdot \mathbf{E}_{ii} = 2n \|\boldsymbol{A}\|_{\mathrm{F}}^2 \cdot \mathbf{I}$$

$$\mathbb{E}[\boldsymbol{Z}^*\boldsymbol{Z}] = \sum\nolimits_{ij} \frac{|a_{ij}|^2}{p_{ij}^2} \cdot \mathbf{E}_{ij}^*\mathbf{E}_{ij} \cdot p_{ij} \preccurlyeq \sum\nolimits_{ij} 2 \|\boldsymbol{A}\|_{\mathrm{F}}^2 \cdot \mathbf{E}_{jj} = 2n \|\boldsymbol{A}\|_{\mathrm{F}}^2 \cdot \mathbf{I}$$

- Thus, $\max\{\|\mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^*]\|, \|\mathbb{E}[\boldsymbol{Z}^*\boldsymbol{Z}]\|\} \leq 2n \|\boldsymbol{A}\|_{\mathrm{F}}^2$

[Refs] Kundu & Drineas 2014. T 2014, *User-Friendly Tools*, Chap. 6.

Error Bound for Sparsification

- Corollary 3 provides the error bound

$$\mathbb{E}\,\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\| \leq \sqrt{\frac{2n\,\|\boldsymbol{A}\|_{\mathrm{F}}^2 \log(2n)}{\mathtt{r}}} + \frac{2\,\|\boldsymbol{A}\|_{\ell_1} \log(2n)}{3\mathtt{r}}$$

- Note that $\|\boldsymbol{A}\|_{\ell_1} \leq n\,\|\boldsymbol{A}\|_{\mathrm{F}}$, and place error on relative scale:

$$\frac{\mathbb{E}\,\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|}{\|\boldsymbol{A}\|} \leq \frac{\|\boldsymbol{A}\|_{\mathrm{F}}}{\|\boldsymbol{A}\|} \left[\sqrt{\frac{2n\log(2n)}{\mathtt{r}}} + \frac{2n\log(2n)}{3\mathtt{r}} \right]$$

- **[Q]** What does this mean?

[Refs] Kundu & Drineas 2014. T 2014, *User-Friendly Tools*, Chap. 6.

Detour: The Stable Rank

- The *stable rank* of a matrix is defined as

$$\operatorname{srank}(\boldsymbol{A}) := \frac{\|\boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}\|^2}$$

- In general, $1 \leq \operatorname{srank}(\boldsymbol{A}) \leq \operatorname{rank}(\boldsymbol{A})$
- When $\boldsymbol{A}$ has either n rows or n columns, $1 \leq \operatorname{srank}(\boldsymbol{A}) \leq n$
- **Assume** that $\boldsymbol{A}$ has n unit-norm columns, so that $\|\boldsymbol{A}\|_{\mathrm{F}}^2 = n$
- When all columns of $\boldsymbol{A}$ are the same, $\|\boldsymbol{A}\|^2 = n$ and $\operatorname{srank}(\boldsymbol{A}) = 1$
- When all columns of $\boldsymbol{A}$ are orthogonal, $\|\boldsymbol{A}\|^2 = 1$ and $\operatorname{srank}(\boldsymbol{A}) = n$

[Refs] Bourgain & Tzafriri 1987; Rudelson & Vershynin 2007; T 2009; Vershynin 2011. See T 2014, *User-Friendly Tools*, Chap. 6.

Error Bound for Sparsification II

- Fix a tolerance $\varepsilon \in (0, 1)$
- Suppose the sparsity $\mathtt{r}$ satisfies

$$\mathtt{r} \geq \varepsilon^{-2} \cdot 2n \log(2n) \cdot \operatorname{srank}(\boldsymbol{A})$$

- Then the $\mathtt{r}$-sparse matrix $\widehat{\boldsymbol{A}}$ achieves relative error

$$\frac{\mathbb{E} \, \|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|}{\|\boldsymbol{A}\|} \leq 2\varepsilon$$

- If $\operatorname{srank}(\boldsymbol{A}) \ll n$, then $\operatorname{nnz}(\widehat{\boldsymbol{A}}) \ll n^2$ suffices

[Refs] Kundu & Drineas 2014. See T 2014, *User-Friendly Tools*, Chap. 6.

Random Features

Kernel Matrices

- Consider data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathscr{X} \subset \mathbb{R}^d$
- Let $k : \mathscr{X} \times \mathscr{X} \to [-1, +1]$ be a bounded similarity measure
- Form $n \times n$ kernel matrix $\boldsymbol{K}$ with entries $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- Useful for regression, classification, feature selection, ...
- **Challenge:** Kernel is big $(n \times n)$ and expensive to evaluate $\mathcal{O}(dn^2)$
- **Opportunity:** The kernel often has low effective dimension
- **Idea:** Construct a randomized low-rank approximation of the kernel

[Refs] Schölkopf & Smola 2002; Rahimi & Recht 2007, 2008.

Random Features

- Let $\mathscr{W}$ be an auxiliary space
- Let $\varphi : \mathscr{X} \times \mathscr{W} \to [-1, +1]$ be a bounded *feature map*
- Let $\boldsymbol{w}$ be a random variable taking values in $\mathscr{W}$
- **Assume** the random feature $\varphi(\boldsymbol{x}; \boldsymbol{w})$ has the *reproducing property*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}_{\boldsymbol{w}} \left[\varphi(\boldsymbol{x}; \boldsymbol{w}) \cdot \varphi(\boldsymbol{y}; \boldsymbol{w}) \right] \quad \text{for all } \boldsymbol{x}, \boldsymbol{y} \in \mathscr{X}$$

- **Example:** Random Fourier features for shift-invariant kernels
- **Related:** Random Walsh features for inner-product kernels

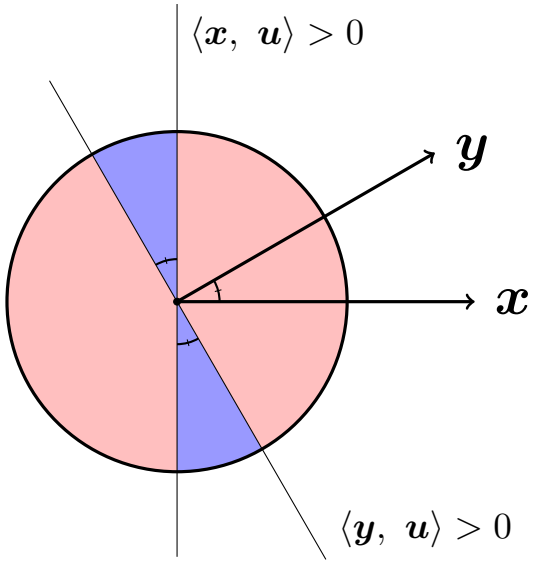
[Refs] Rahimi & Recht 2007, 2008; Maji & Berg 2009; Li et al. 2010; Vedaldi & Zisserman 2010; Vempati et al. 2010; Kar & Karnack 2012; Le et al. 2013; Pham & Pagh 2013; Hamid et al. 2014; ...

Example: Angular Similarity

- For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, consider the angular similarity

$$k(\boldsymbol{x}, \boldsymbol{y}) = \frac{2}{\pi} \arcsin \frac{\langle \boldsymbol{x},\ \boldsymbol{y} \rangle}{\|\boldsymbol{x}\|\, \|\boldsymbol{y}\|} = 1 - \frac{2 \measuredangle(\boldsymbol{x}, \boldsymbol{y})}{\pi} \in [-1, +1]$$

- Define random feature $\varphi(\boldsymbol{x}; \boldsymbol{w}) = \operatorname{sgn} \langle \boldsymbol{x},\ \boldsymbol{w} \rangle$ with $\boldsymbol{w} \sim \textsc{unif}(\mathbb{S}^{d-1})$



$$k(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}[\varphi(\boldsymbol{x}; \boldsymbol{w}) \cdot \varphi(\boldsymbol{y}; \boldsymbol{w})]$$

$\operatorname{sgn} \langle \boldsymbol{x},\ \boldsymbol{u} \rangle \langle \boldsymbol{y},\ \boldsymbol{u} \rangle > 0$

$\operatorname{sgn} \langle \boldsymbol{x},\ \boldsymbol{u} \rangle \langle \boldsymbol{y},\ \boldsymbol{u} \rangle < 0$

[Refs] Early 1900s; Grothendieck 1953; Goemans & Williamson 1996.

Low-Rank Approximation of Kernel Matrix

- Draw a random vector $\boldsymbol{w} \in \mathscr{W}$
- Define $z_i = \varphi(\boldsymbol{x}_i; \boldsymbol{w})$ for each datum $i = 1, \dots, n$
- By the reproducing property, $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{E}[z_i z_j]$
- In matrix form: $\boldsymbol{K} = \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^*]$ where $\boldsymbol{z} = [z_i] \in \mathbb{R}^n$
- That is, $\boldsymbol{Z} = \boldsymbol{z}\boldsymbol{z}^*$ is an unbiased rank-one estimator for $\boldsymbol{K}$
- Draw iid copies $\boldsymbol{Z}_1, \dots, \boldsymbol{Z}_{\mathtt{r}}$ of the estimator $\boldsymbol{Z}$
- Thus, $\widehat{\boldsymbol{K}} = \frac{1}{\mathtt{r}} \sum_{r=1}^{\mathtt{r}} \boldsymbol{Z}_r$ is an unbiased rank-$\mathtt{r}$ estimator for $\boldsymbol{K}$

[Refs] Rahimi & Recht 2007, 2008; Lopez-Paz et al. ICML 2014.

Analysis of Random Features

- Recall: $\boldsymbol{Z} = \boldsymbol{z}\boldsymbol{z}^*$ where $\mathbb{E}\,\boldsymbol{Z} = \boldsymbol{K}$ and $\|\boldsymbol{z}\|_{\ell_\infty^n} \leq 1$
- Uniform bound:

$$\|\boldsymbol{Z}\| = \|\boldsymbol{z}\boldsymbol{z}^*\| = \|\boldsymbol{z}\|^2 \leq n$$

- Variance computation:

$$0 \preccurlyeq \mathbb{E}[\boldsymbol{Z}^2] = \mathbb{E}[\|\boldsymbol{z}\|^2 \cdot \boldsymbol{z}\boldsymbol{z}^*] \preccurlyeq n \cdot \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^*] = n \cdot \boldsymbol{K}$$

- Thus, $\left\|\mathbb{E}[\boldsymbol{Z}^2]\right\| \leq n \|\boldsymbol{K}\|$

[Ref] Lopez-Paz et al. ICML 2014.

Error Bound for Random Features

- Corollary 3 implies that

$$\mathbb{E}\,\|\boldsymbol{K} - \widehat{\boldsymbol{K}}\| \leq \sqrt{\frac{2n\,\|\boldsymbol{K}\|\log(2n)}{\mathtt{r}}} + \frac{2n\log(2n)}{3\mathtt{r}}$$

- Define the *effective rank* $\rho := n/\,\|\boldsymbol{K}\|$ of the kernel to obtain

$$\frac{\mathbb{E}\,\|\boldsymbol{K} - \widehat{\boldsymbol{K}}\|}{\|\boldsymbol{K}\|} \leq \sqrt{\frac{2\rho\log(2n)}{\mathtt{r}}} + \frac{2\rho\log(2n)}{3\mathtt{r}}$$

- For relative error $\mathcal{O}(\varepsilon)$, need at most $\mathtt{r}$ random features where

$$\mathtt{r} \geq 2\varepsilon^{-2}\rho\log(2n)$$

- Random features are efficient when $\rho \ll n$

[Ref] Lopez-Paz et al., ICML 2014.

Entr'acte

Optimal Low-Rank Approximation

- Let $\boldsymbol{A}$ be an $n \times n$ matrix with SVD $\boldsymbol{A} = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*$
- **Assume** normalization $\sum_{i=1}^{n} \sigma_i = 1$
- "Optimal" low-rank sampling: $\boldsymbol{Z} = \boldsymbol{u}_i \boldsymbol{v}_i^*$ with probability σ_i
- Corollary 3 gives error bound for $\widehat{\boldsymbol{A}} = \frac{1}{\mathtt{r}} \sum_{r=1}^{\mathtt{r}} \boldsymbol{Z}_r$:

$$\mathbb{E} \, \|\boldsymbol{A} - \widehat{\boldsymbol{A}}\| \leq \sqrt{\frac{2\sigma_1 \log(2n)}{\mathtt{r}}} + \frac{2 \log(2n)}{3\mathtt{r}}$$

- **Mirsky's Theorem (1969).** The best rank-$\mathtt{r}$ approximation satisfies

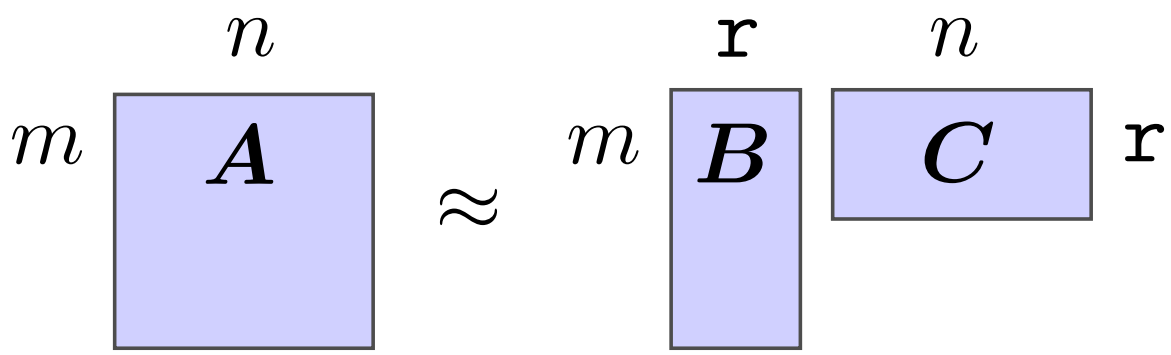
$$\min_{\mathrm{rank}(\boldsymbol{B})=\mathtt{r}} \|\boldsymbol{A} - \boldsymbol{B}\| = \sigma_{\mathtt{r}+1} \leq \frac{1}{\mathtt{r}+1}$$

- **Sampling is really (really) suboptimal!**

Two-stage Low-rank Approximations

Low-Rank Approximations

- **Goal:** Given a large matrix $\boldsymbol{A}$, find low-rank factors:

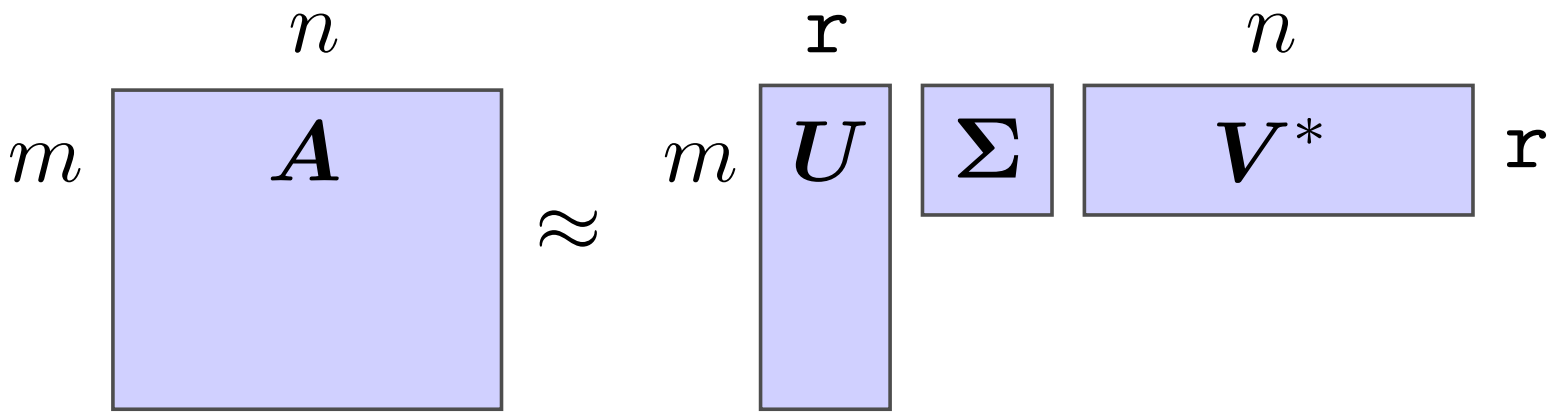


- **Solving this problem gives algorithms for computing**
  - Leading singular vectors of a general matrix
  - Leading eigenvectors of a symmetric matrix
  - Spanning sets of rows or columns

- We will focus on controlling backward error: $\|\boldsymbol{A} - \boldsymbol{BC}\| \leq \texttt{tol}$

Example: Truncated SVD

$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ where $\boldsymbol{U}, \boldsymbol{V}$ have orthonormal columns and $\boldsymbol{\Sigma}$ is diagonal:



Interpretation: r-truncated SVD = optimal rank-r approximation

Applications:

- Least-squares computations
- Principal component analysis
- Summarization and data reduction
- ...

Overview of Two-Stage Randomized SVD

Goal: Construct a truncated SVD of an input matrix $\boldsymbol{A}$

Stage A: Finding the range

- Use a *randomized algorithm* to compute a low-dimensional basis $\boldsymbol{Q}$ that captures most of the action of $\boldsymbol{A}$:

$$\boldsymbol{Q} \text{ has orthonormal columns and } \boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}.$$

Stage B: Forming the decomposition

- Use the basis $\boldsymbol{Q}$ to reduce the problem size
- Apply *classical linear algebra* to reduced problem
- Obtain truncated SVD in factored form

[Ref] Halko et al. 2011, §1.

Stage A: Finding the Range

Given:

- An $m \times n$ matrix $\boldsymbol{A}$
- Target rank $\mathtt{r}_0 \ll \min\{m, n\}$
- Actual rank $\mathtt{r} = \mathtt{r}_0 + \mathtt{s}$ where $\mathtt{s}$ is a small oversampling parameter

Construct an $m \times \mathtt{r}$ matrix $\boldsymbol{Q}$ with orthonormal columns s.t.

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \approx \min_{\mathrm{rank}(\boldsymbol{B})=\mathtt{r}_0} \|\boldsymbol{A} - \boldsymbol{B}\|,$$

- $\boldsymbol{Q}\boldsymbol{Q}^*$ is the orthogonal projector onto the range of $\boldsymbol{Q}$

Approach: Use a randomized algorithm!

Total Cost: One multiply $(m \times n \times \mathtt{r}) + \mathcal{O}(\mathtt{r}^2 n)$ flops

Stage B: Forming the SVD

Assume $\boldsymbol{A}$ is $m \times n$ and $\boldsymbol{Q}$ is $m \times \mathtt{r}$ with ON columns and $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}$

Approach: Reduce problem size; apply classical numerical linear algebra!

1. Form $\mathtt{r} \times n$ matrix $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$
2. Factor $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$
3. Conclude $\boldsymbol{A} \approx (\boldsymbol{Q}\boldsymbol{U})\boldsymbol{\Sigma}\boldsymbol{V}^*$

$$\boxed{\boldsymbol{A}} \approx \boxed{\boldsymbol{Q}}\,\boxed{\boldsymbol{Q}^*}\,\boxed{\boldsymbol{A}} = \boxed{\boldsymbol{Q}}\,\boxed{\boldsymbol{Q}^*\boldsymbol{A}}$$

$$= \boxed{\boldsymbol{Q}}\,\boxed{\boldsymbol{U}}\,\boxed{\boldsymbol{\Sigma}}\,\boxed{\boldsymbol{V}^*} = \boxed{\boldsymbol{Q}\boldsymbol{U}}\,\boxed{\boldsymbol{\Sigma}}\,\boxed{\boldsymbol{V}^*}$$

Total Cost: One multiply $(m \times n \times \mathtt{r}) + \mathcal{O}(\mathtt{r}^2 n)$ flops

Total Costs for Truncated SVD

Two-Stage Randomized Algorithm:

$$2 \text{ multiplies } (m \times n \times \mathtt{r}) \quad + \quad \mathtt{r}^2(m+n) \quad \text{flops}$$

Classical Sparse Methods (Krylov)**:**

$$\mathtt{r}_0 \text{ multiplies } (m \times n \times 1) \quad + \quad \mathtt{r}_0^2(m+n) \quad \text{flops}$$

Classical Dense Methods (RRQR + full SVD)**:**

$$\text{Not based on multiplies} \quad + \quad \mathtt{r}_0 mn \quad \text{flops}$$

Roadmap

1. How to use the range information to construct matrix approximations
2. How to find a subspace that captures the range of the matrix
3. Other types of sampling schemes
4. How to get high precision approximations by iteration

From Range to Decomposition

Interpolative Approximations

- *Interpolative approximation* represents matrix via a few rows (or cols)
- Takes the form $\boldsymbol{A} \approx \boldsymbol{W}\boldsymbol{A}_{I:}$ where
  - I is a small set of row indices
  - $\boldsymbol{W}_{I:} = \mathbf{I}$
  - All entries of $\boldsymbol{W}$ have magnitude ≤ 2
- Can be constructed efficiently with rank-revealing QR
- Useful when we must preserve meaning of rows (or cols)

Assume $\boldsymbol{A}$ is $m \times n$ and $\boldsymbol{Q}$ is $m \times \mathtt{r}$ and $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}$

1. Form interpolative decomposition $\boldsymbol{Q} = \boldsymbol{W}\boldsymbol{Q}_{I:}$ with $|I| = \mathtt{r}$
2. Conclude $\boldsymbol{A} \approx \boldsymbol{W}\boldsymbol{A}_{I:}$

Total cost: $\mathcal{O}(\mathtt{r}^2 m)$ flops

[Ref] Gu & Eisenstat 1996; Goreinov et al. 1997; Cheng et al. 2005. See Halko et al. 2011, §3.2.3 and §5.2.

Matrix Skeletons

- A *skeleton approximation* represents a matrix via a small submatrix
- Takes the form $\boldsymbol{A} \approx \boldsymbol{W} \boldsymbol{A}_{IJ} \boldsymbol{X}$ where
  - I is a small set of row indices; J is a small set of column indices
  - $\boldsymbol{W}_{I:} = \mathbf{I}$ and $\boldsymbol{X}_{:J} = \mathbf{I}$
  - All entries of $\boldsymbol{W}$ and $\boldsymbol{X}$ have magnitude ≤ 2
- Useful when we must preserve meaning of rows and columns

Assume $\boldsymbol{A}$ is $m \times n$ and $\boldsymbol{Q}$ is $m \times \mathtt{r}$ and $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}$

1. Form row interpolative decomposition $\boldsymbol{Q} = \boldsymbol{W} \boldsymbol{Q}_{I:}$ with $|I| = \mathtt{r}$
2. Form column interpolative decomposition $\boldsymbol{A}_{I:} = \boldsymbol{A}_{IJ} \boldsymbol{X}$ with $|J| = \mathtt{r}$
3. Conclude $\boldsymbol{A} \approx \boldsymbol{W} \boldsymbol{A}_{IJ} \boldsymbol{X}$

Total cost: $\mathcal{O}(\mathtt{r}^2(m+n))$ flops

[Ref] Gu & Eisenstat 1996; Goreinov et al. 1997; Cheng et al. 2005. See Halko et al. 2011, §3.2.3 and §5.2.

CUR Approximations

- A *CUR approximation* represents a matrix via a few rows and columns
- Takes the form $\boldsymbol{A} \approx \boldsymbol{A}_{:J} \boldsymbol{T} \boldsymbol{A}_{I:}$ where I and J are small
- Useful when we must preserve meaning of rows and columns

Assume $\boldsymbol{A}$ is $m \times n$ and $\boldsymbol{Q}$ is $m \times \mathtt{r}$ and $\boldsymbol{P}$ is $n \times \mathtt{r}$ and

$$\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A} \quad \text{and} \quad \boldsymbol{A} \approx \boldsymbol{A}\boldsymbol{P}\boldsymbol{P}^*$$

1. Form row interpolative decomposition $\boldsymbol{Q} = \boldsymbol{W}\boldsymbol{Q}_{I:}$ with $|I| = \mathtt{r}$
2. Form column interpolative decomposition $\boldsymbol{P} = \boldsymbol{P}_{:J}\boldsymbol{X}$ with $|J| = \mathtt{r}$
3. Compute the pseudoinverse $\boldsymbol{T} = (\boldsymbol{A}_{IJ})^\dagger$
4. Conclude $\boldsymbol{A} \approx \boldsymbol{A}_{:J} \boldsymbol{T} \boldsymbol{A}_{I:}$

Total cost: $\mathcal{O}(\mathtt{r}^2(m+n))$ flops

[Ref] Goreinov et al. 1997; Drineas et al. 2009; Mahoney & Drineas 2009; Bien et al. 2010.

Nyström Approximation for PSD Matrices

- *Nyström approximations* represent psd matrices (much) more accurately

Assume $\boldsymbol{A}$ is $n \times n$ psd and $\boldsymbol{Q}$ is $n \times \mathtt{r}$ with ON columns and $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}$

Nyström approximation:

$$\boldsymbol{A} \approx (\boldsymbol{A}\boldsymbol{Q})(\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q})^\dagger(\boldsymbol{A}\boldsymbol{Q})^*$$

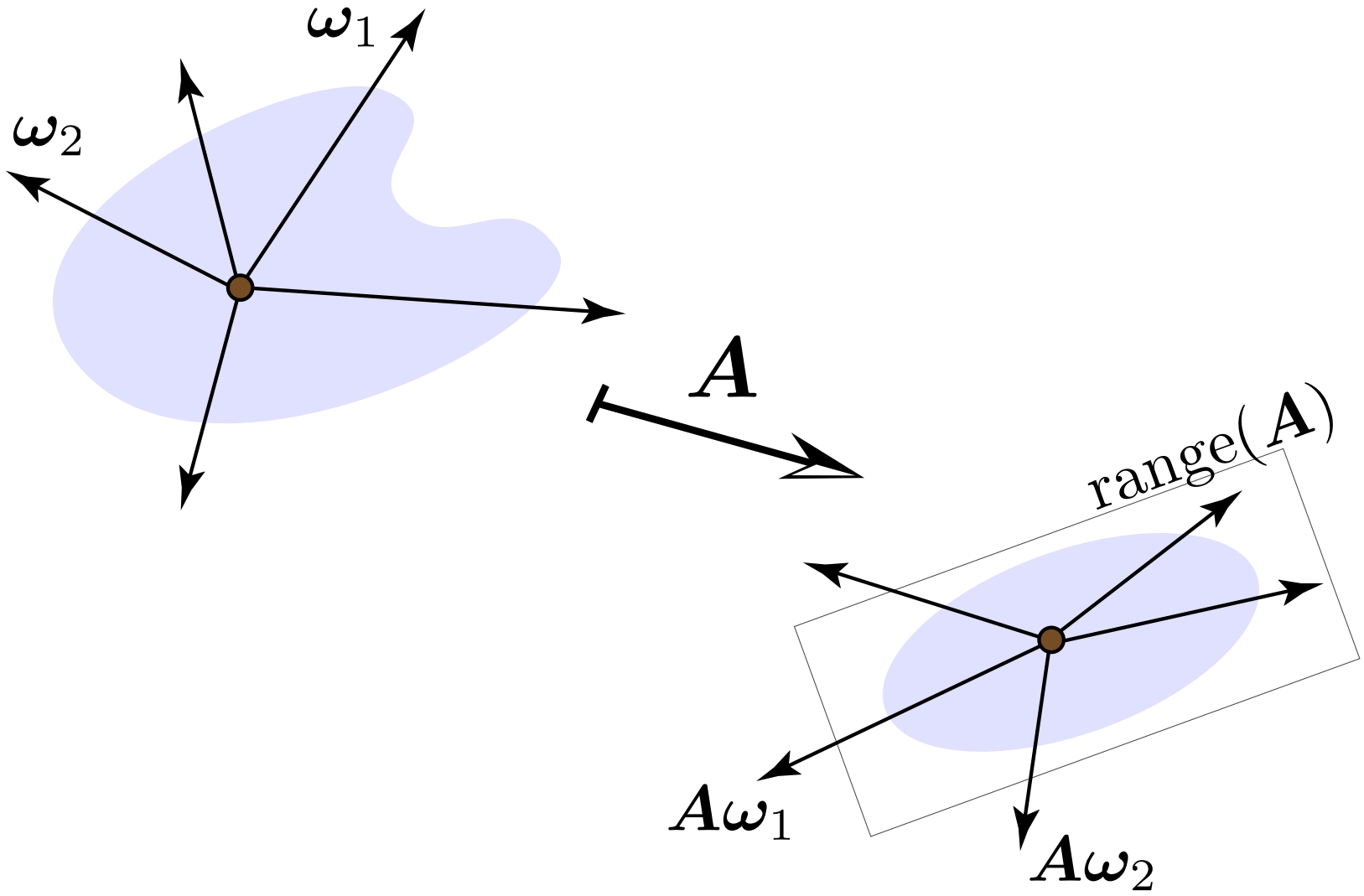
1. Form $n \times n \times \mathtt{r}$ product $\boldsymbol{F}_0 = \boldsymbol{A}\boldsymbol{Q}$
2. Form $\mathtt{r} \times n \times \mathtt{r}$ product $\boldsymbol{F} = \boldsymbol{Q}^*\boldsymbol{F}_0$
3. Compute $\mathtt{r} \times \mathtt{r}$ Cholesky decomposition $\boldsymbol{F} = \boldsymbol{T}^*\boldsymbol{T}$
4. Form $n \times \mathtt{r}$ matrix $\boldsymbol{H} = \boldsymbol{F}_0\boldsymbol{T}^{-1}$ by triangular solve
5. Conclude $\boldsymbol{A} \approx \boldsymbol{H}\boldsymbol{H}^*$

Total cost: One multiply $(n \times n \times \mathtt{r}) + \mathcal{O}(\mathtt{r}^2 n)$ flops

[Ref] Williams & Seeger 2002; Drineas & Mahoney 2005; Halko et al. 2011, §5.4; Gittens 2011; Gittens & Mahoney 2013.

Randomized Range Finder

Randomized Range Finder: Intuition



Prototype for Randomized Range Finder

Input: An $m \times n$ matrix $\boldsymbol{A}$, number $\mathtt{r}$ of samples

Output: An $m \times \mathtt{r}$ matrix $\boldsymbol{Q}$ with orthonormal columns

1. Draw an $n \times \mathtt{r}$ random matrix $\boldsymbol{\Omega}$.
2. Form the matrix product $\boldsymbol{Y} = \boldsymbol{A\Omega}$.
3. Construct an orthonormal basis $\boldsymbol{Q}$ for the range of $\boldsymbol{Y}$.

Total Cost: 1 multiply $(m \times n \times \mathtt{r}) + \mathcal{O}(\mathtt{r}^2 n)$ flops

[Refs] NLA community: Stewart (1970s). GFA: Johnson & Lindenstrauss (1984) et seq. TCS: Boutsidis, Deshpande, Drineas, Frieze, Kannan, Mahoney, Papadimitriou, Sarlós, Vempala (1998–present). SciComp: Martinsson, Rokhlin, Szlam, Tygert (2004–present). See Halko et al. 2011, §2 for more history.

Implementation Issues

Q: How do we pick the number $\mathtt{r}$ of samples?

A: Adaptively, using a randomized error estimator.

Q: How does the number $\mathtt{r}$ of samples compare with the target rank $\mathtt{r}_0$?

A: Remarkably, $\mathtt{r} = \mathtt{r}_0 + 5$ or $\mathtt{r} = \mathtt{r}_0 + 10$ is usually adequate!

Q: What random matrix $\boldsymbol{\Omega}$?

A: For many applications, standard Gaussian works brilliantly.

Q: How do we perform the matrix–matrix multiply?

A: Exploit the computational architecture.

Q: How do we compute the orthonormal basis?

A: Carefully... Double Gram–Schmidt or Householder reflectors.

[Ref] Halko et al. 2011, §4.

Simple Error Bound for Random Range Finder

Theorem 4. [HMT 2011] **Assume**

- *The matrix* $\boldsymbol{A}$ *is* $m \times n$ *with* $m \geq n$
- *The target rank is* $\mathtt{r}_0$
- *The optimal error* $\sigma_{\mathtt{r}_0+1} = \min_{\mathrm{rank}(\boldsymbol{B})=\mathtt{r}_0} \|\boldsymbol{A} - \boldsymbol{B}\|$
- *The test matrix* $\boldsymbol{\Omega}$ *is* $n \times (\mathtt{r}_0 + \mathtt{s})$ *standard Gaussian*

Then *the random range finder yields an* $(\mathtt{r}_0 + \mathtt{s})$*-dimensional basis* $\boldsymbol{Q}$ *with*

$$\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \left[1 + \frac{4\sqrt{\mathtt{r}_0 + \mathtt{s}}}{\mathtt{s} - 1} \cdot \sqrt{n}\right] \sigma_{\mathtt{r}_0+1}$$

The probability of a substantially larger error is negligible.

[Ref] Halko et al. 2011, §10.2.

Error Bound for Random Range Finder

Theorem 5. [HMT 2011] **Assume**

- *The matrix* $\boldsymbol{A}$ *is* $m \times n$
- *The target rank is* $\mathtt{r}_0$
- *The singular values of* $\boldsymbol{A}$ *are* $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots$
- *The test matrix* $\boldsymbol{\Omega}$ *is* $n \times (\mathtt{r}_0 + \mathtt{s})$ *standard Gaussian*

Then *the random range finder yields an* $(\mathtt{r}_0 + \mathtt{s})$*-dimensional basis* $\boldsymbol{Q}$ *with*

$$\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \left[1 + \sqrt{\frac{\mathtt{r}_0}{\mathtt{s}-1}}\right] \sigma_{\mathtt{r}_0+1} + \frac{\mathrm{e}\sqrt{\mathtt{r}_0+\mathtt{s}}}{\mathtt{s}} \left(\sum\nolimits_{j>\mathtt{r}_0} \sigma_j^2\right)^{1/2}$$

The probability of a substantially larger error is negligible.
This bound is essentially optimal.

[Ref] Halko et al. 2011, §10.2.

Adaptive Error Estimation

- **In practice, we do not know the target rank!**
  - A priori error bounds are motivating—but not so useful
- **Solution:** Estimate the error, and stop when we reach a target
- Let $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_{10}$ be iid standard Gaussian vectors

$$\texttt{err\_est} := \max_{1 \le j \le 10} \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{\omega}_j\|$$

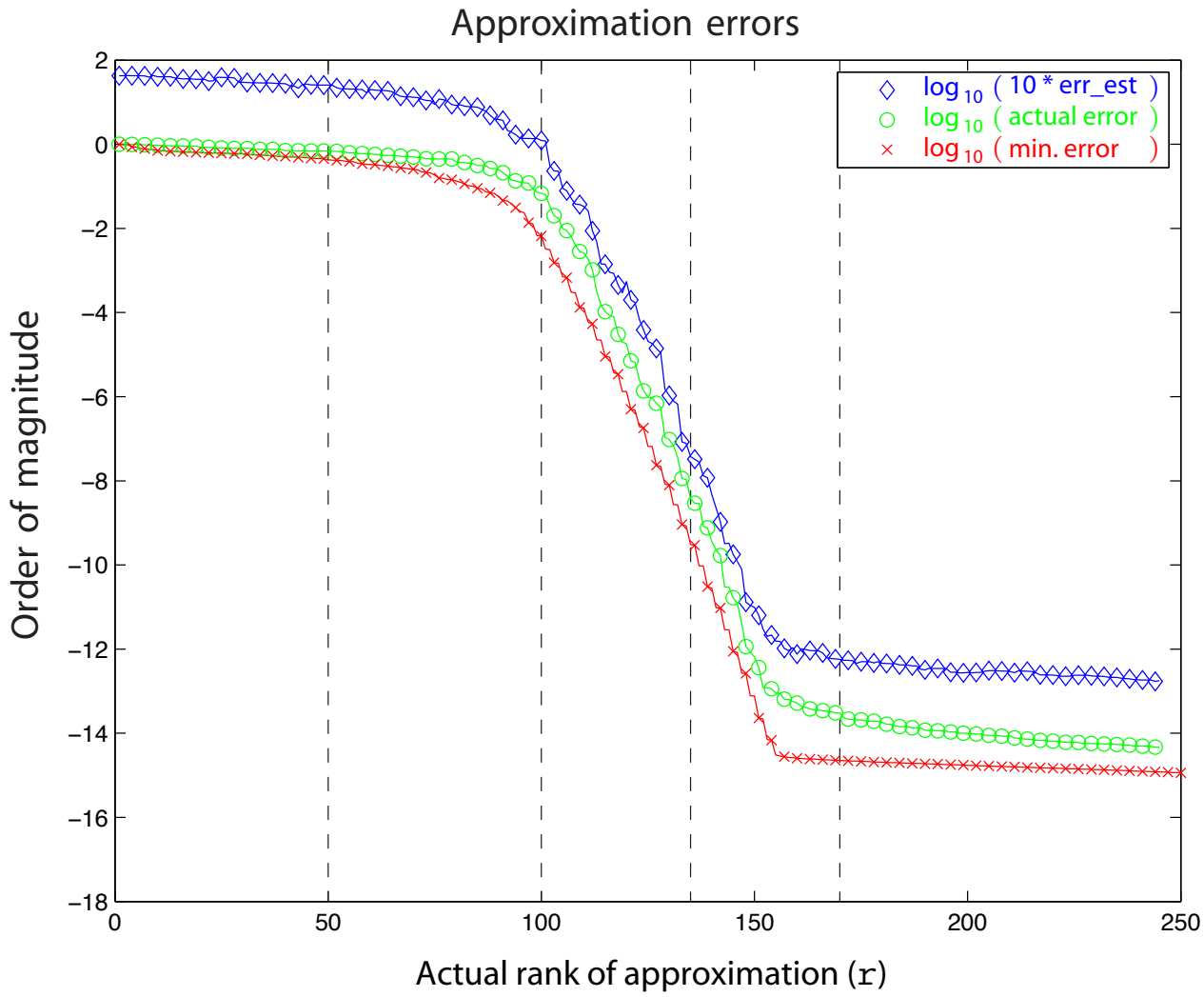
- How good is this estimate?

$$\mathbb{P}\left\{ \ \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\| \ge 10 \cdot \texttt{err\_est} \ \right\} \le 10^{-10}$$

- Can modify the random range finder to form `err_est` (almost) for free

[Refs] Woolfe et al. 2008. See Halko et al. 2011, §4.4.

Approximating a Helmholtz Integral Operator



Other Sampling Strategies

Other Sampling Techniques?

- In randomized range finder, can change distribution of test matrix $\boldsymbol{\Omega}$
- **Why?**
  - To exploit $\boldsymbol{\Omega}$ with fast multiply
  - To avoid dense $\boldsymbol{Y} = \boldsymbol{A\Omega}$ when $\boldsymbol{A}$ is sparse
- **When are these techniques worth considering?**
  - The matrix $\boldsymbol{A}$ *must* have a rapidly decaying spectrum
  - The dominant right singular vectors of the matrix $\boldsymbol{A}$ are flat
- **Executive summary:**
  - Sampling with an SRFT is effective when $\boldsymbol{A}$ has spectral decay
  - **I do not recommend the other schemes, except in special cases**

Deterministic Error Bound for Range Finder

- $\boldsymbol{A}$ is $m \times n$ with SVD partitioned as

$$\boldsymbol{A} = \boldsymbol{U} \begin{array}{c} \begin{matrix} \mathtt{r}_0 & n - \mathtt{r}_0 \end{matrix} \\ \begin{bmatrix} \boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} \mathtt{r}_0 \\ n - \mathtt{r}_0 \end{matrix} \\ \begin{bmatrix} \boldsymbol{V}_1^* \\ \boldsymbol{V}_2^* \end{bmatrix} \end{array}$$

- Let $\boldsymbol{\Omega}$ be any test matrix, decomposed as

$$\boldsymbol{\Omega}_1 = \boldsymbol{V}_1^* \boldsymbol{\Omega} \quad \text{and} \quad \boldsymbol{\Omega}_2 = \boldsymbol{V}_2^* \boldsymbol{\Omega}.$$

- Construct the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ and a basis $\boldsymbol{Q}$ for $\operatorname{range}(\boldsymbol{Y})$

Theorem 6. [BMD11; HMT11] *When $\boldsymbol{\Omega}_1$ has full row rank,*

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|^2 \leq \|\boldsymbol{\Sigma}_2\|^2 + \left\|\boldsymbol{\Sigma}_2 \boldsymbol{\Omega}_2 \boldsymbol{\Omega}_1^\dagger\right\|^2.$$

Roughly: *Range finder works when $\boldsymbol{\Omega}_1$ has well-conditioned rows.*

[Refs] Boutsidis et al. 2011. See Halko et al. 2011, §9.

Leverage Score Sampling

- Let $\boldsymbol{V}_1^*$ be the $\mathtt{r}_0 \times n$ matrix of top right singular vectors of $\boldsymbol{A}$
- The squared column norms $\ell_1, \ldots, \ell_n$ of $\boldsymbol{V}_1^*$ are called *leverage scores*
- We can obtain a sampling distribution by normalizing: $p_i = \ell_i / \mathtt{r}_0$
- Let the number $\mathtt{r}$ of samples satisfy $\mathtt{r} \sim \mathtt{r}_0 \log \mathtt{r}_0$
- Let $\boldsymbol{\omega} = \mathbf{e}_i$ with probability p_i
- Test matrix $\boldsymbol{\Omega}$ has $\mathtt{r}$ columns, each an independent copy of $\boldsymbol{\omega}$
- Then $\boldsymbol{\Omega}_1 = \boldsymbol{V}_1^* \boldsymbol{\Omega}$ is likely to have well-conditioned rows
- Leverage score sampling respects columns
- **But...** Cost of approximating leverage scores is $\mathcal{O}(mn \log m)$
- **Better method:** Use SRFT (p. 63) + interpolative approximation

$\mathbf{e}_i = i$th standard basis vector

[Ref] See Mahoney 2011 for details about leverage score sampling.

Simple Column Sampling

- **Assume** that the columns of $\boldsymbol{V}_1^*$ have comparable norms
- Then we can take a sampling distribution $p_i = 1/n$ for each i
- Let the number $\mathtt{r}$ of samples satisfy $\mathtt{r} \sim \mathtt{r}_0 \log \mathtt{r}_0$
- Let $\boldsymbol{\omega} = \mathbf{e}_i$ with probability p_i
- Test matrix $\boldsymbol{\Omega}$ has $\mathtt{r}$ columns, each an independent copy of $\boldsymbol{\omega}$
- Then $\boldsymbol{\Omega}_1 = \boldsymbol{V}_1^* \boldsymbol{\Omega}$ is likely to have well-conditioned rows
- Simple random sampling respects columns and is basically free
- Works when leverage scores are approximately uniform
- **But...** the approach fails when leverage scores are nonuniform

[Refs] See Chen & Demanet 2009; Gittens 2009; Gittens & Mahoney 2013 for analysis of simple random sampling.

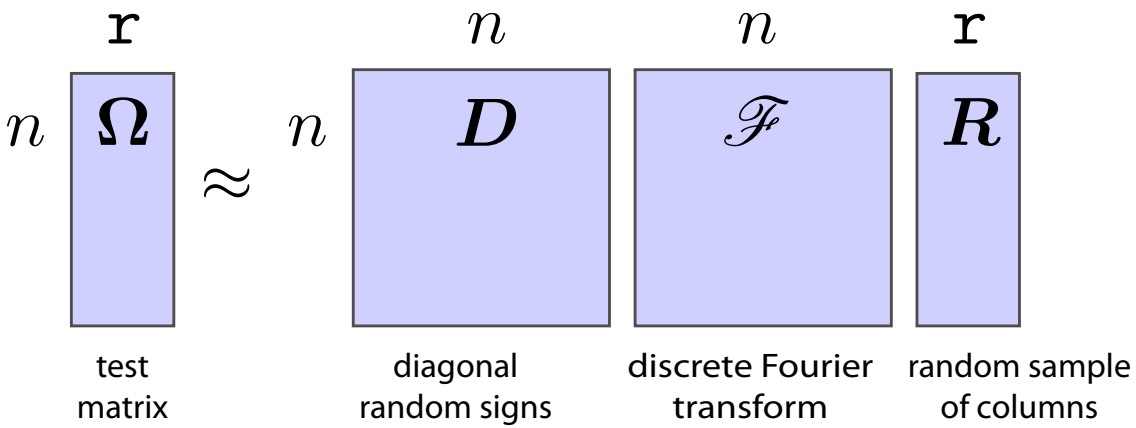
Sparse Projections

- **Assume** that the matrix $\boldsymbol{A}$ is very sparse
- May be valuable to try to preserve this sparsity
- Consider test matrix $\boldsymbol{\Omega}$ has $\log^{\mathcal{O}(1)}(n)$ random ± 1 entries per column
- The number $\mathtt{r}$ of columns in $\boldsymbol{\Omega}$ has order $\mathtt{r}_0 \log^{\mathcal{O}(1)}(n)$
- Then $\boldsymbol{\Omega}_1 = \boldsymbol{V}_1^* \boldsymbol{\Omega}$ is likely to have well-conditioned rows
- Sparse projections respect sparsity and are relatively cheap
- **But...** this approach tends not to work well for sparse matrices
- **Ironically,** may be more valuable for dense matrices

[Refs] Clarkson & Woodruff 2012; Nelson & Nguyen 2012; Meng & Mahoney 2013; Bourgain & Nelson 2013.

Structured Random Matrices

- Choose a *structured* random matrix $\boldsymbol{\Omega}$ to accelerate multiply
- **Example:** Subsampled randomized Fourier transform (SRFT)



- **Intuition:** The transform $\boldsymbol{D}\mathscr{F}$ uniformizes leverage scores
- Cost of forming $\boldsymbol{Y} = \boldsymbol{A\Omega}$ by FFT is $\mathcal{O}(mn \log n)$
- In practice, if $\boldsymbol{A}$ has spectral decay, SRFT works as well as Gaussian

[Refs] Ailon & Chazelle 2006, 2009; Rokhlin & Tygert 2008; Liberty 2009; T 2011. See Halko et al. 2011, §4.6.

An Error Bound for the SRFT

Theorem 7. [Boutsidis & Gittens 2013] **Assume**

- *The matrix* $\boldsymbol{A}$ *is* $m \times n$
- *The target rank is* $\mathtt{r}_0$
- *The singular values of* $\boldsymbol{A}$ *are* $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots$
- *The test matrix* $\boldsymbol{\Omega}$ *is an* $n \times \mathtt{r}$ *SRFT with*

$$\mathtt{r} \gtrsim (\mathtt{r}_0 + \log n) \log \mathtt{r}_0$$

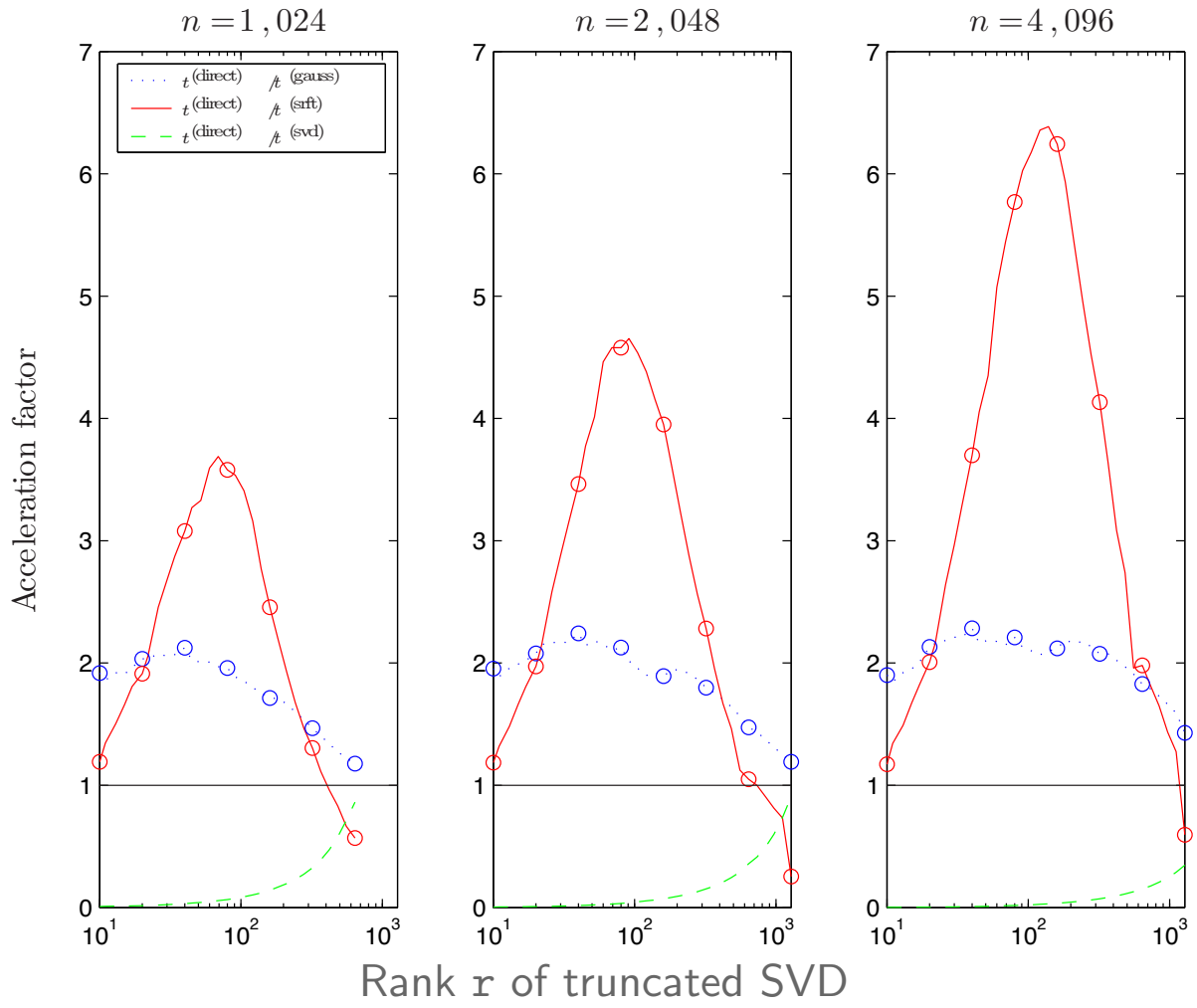
Then

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}\| \lesssim \left[1 + \frac{\log n}{\sqrt{\mathtt{r}}}\right] \sigma_{\mathtt{r}_0+1} + \sqrt{\frac{\log n}{\mathtt{r}}} \cdot \left(\sum\nolimits_{j>\mathtt{r}_0} \sigma_j^2\right)^{1/2}$$

except with probability $\mathcal{O}(\mathtt{r}_0^{-1})$

[Refs] T 2011; Halko et al. 2011; Boutsidis & Gittens 2013.

SRFT: Faster SVD when Spectrum Decays



[Ref] Halko et al. 2011, §7.4.

Randomized Power Scheme

The Role of Spectral Decay

- The random range finder works when the spectrum of $\boldsymbol{A}$ decays quickly (Helmholtz)
- **Problem:** This behavior is not common in data analysis problems
- **Examples:**
  - Matrix is contaminated with noise: $\boldsymbol{A} = \boldsymbol{A}_0 + \boldsymbol{N}$ with $\|\boldsymbol{N}\| \propto \|\boldsymbol{A}_0\|$
  - Matrix spectrum decays slowly: $\sigma_j \sim j^{-\alpha}$ for $\alpha < \frac{1}{2}$
- In these settings, the random range finder **will give unreliable results**
- **Remedy:** Use subspace iteration

Randomized Range Finder + Power Scheme

Problem: The singular values of a data matrix often decay slowly

Remedy: Apply the randomized range finder to $(\boldsymbol{A}\boldsymbol{A}^*)^{\mathtt{q}}\boldsymbol{A}$ for small $\mathtt{q}$

Input: An $m \times n$ matrix $\boldsymbol{A}$, number $\mathtt{r}$ of samples

Output: An $m \times \mathtt{r}$ matrix $\boldsymbol{Q}$ with orthonormal columns

1. Draw an $n \times \mathtt{r}$ random matrix $\boldsymbol{\Omega}$.
2. Carefully form the matrix product $\boldsymbol{Y} = (\boldsymbol{A}\boldsymbol{A}^*)^{\mathtt{q}}\boldsymbol{A}\boldsymbol{\Omega}$.
3. Construct an orthonormal basis $\boldsymbol{Q}$ for the range of $\boldsymbol{Y}$.

Total Cost: $2\mathtt{q} + 1$ multiplies $(m \times n \times \mathtt{r}) + \mathcal{O}(\mathtt{q}\mathtt{r}^2 n)$

[Refs] Stewart 1970s; Roweis 1997; Gu 2007; Rokhlin et al. 2008. See Halko et al. 2011, §4.5.

Implementation Issues

Q: How do we pick the number $\mathtt{r}$ of samples?

A: Adaptively, using a randomized error estimator.

Q: How do we pick the number $\mathtt{q}$ of power iterations?

A: Remarkably, $\mathtt{q} = 2$ or $\mathtt{q} = 3$ is usually adequate!

Q: What random matrix $\boldsymbol{\Omega}$?

A: Standard Gaussian. Other sampling distributions have limited value.

Q: How do we perform the matrix–matrix multiply?

A: Alternately multiply by $\boldsymbol{A}$ and $\boldsymbol{A}^*$.

Q: How do we compute the orthonormal basis?

A: Carefully... Perform QR factorization at each step.

[Ref] Halko et al. 2011, §4.

Simple Error Bound for Power Scheme

Theorem 8. [HMT 2011] **Assume**

- *The matrix* $\boldsymbol{A}$ *is* $m \times n$ *with* $m \geq n$
- *The target rank is* $\mathtt{r}_0$
- *The optimal error* $\sigma_{\mathtt{r}_0+1} = \min_{\mathrm{rank}(\boldsymbol{B}) \leq k} \|\boldsymbol{A} - \boldsymbol{B}\|$
- *The test matrix* $\boldsymbol{\Omega}$ *is* $n \times (\mathtt{r}_0 + \mathtt{s})$ *standard Gaussian*

Then *the basis* $\boldsymbol{Q}$ *computed by the power scheme satisfies*

$$\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \left[1 + \frac{4\sqrt{\mathtt{r}_0 + \mathtt{s}}}{\mathtt{s} - 1} \cdot \sqrt{n}\right]^{1/(2\mathtt{q}+1)} \sigma_{\mathtt{r}_0+1}$$

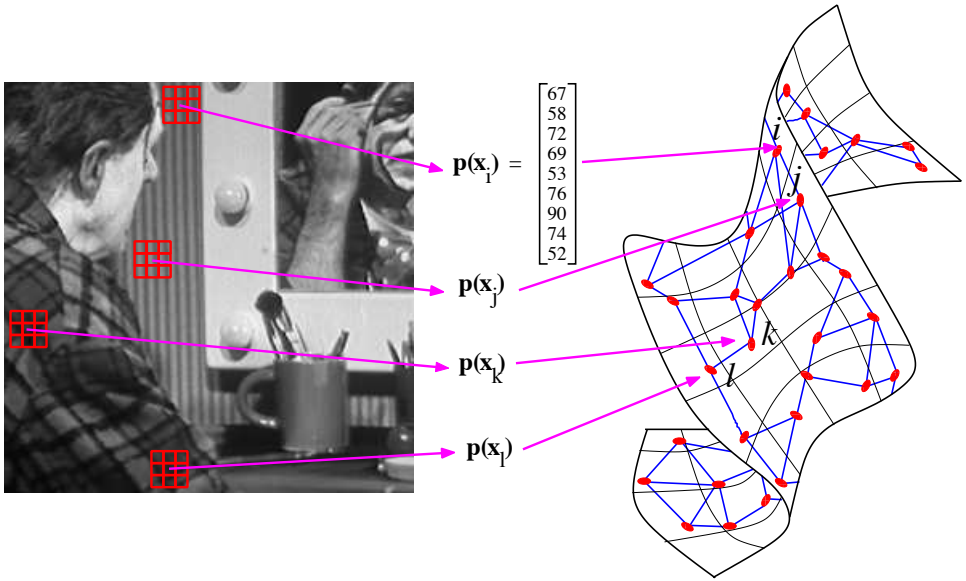
- The power scheme drives the extra factor to one exponentially fast!

[Ref] Halko et al. 2011, §10.

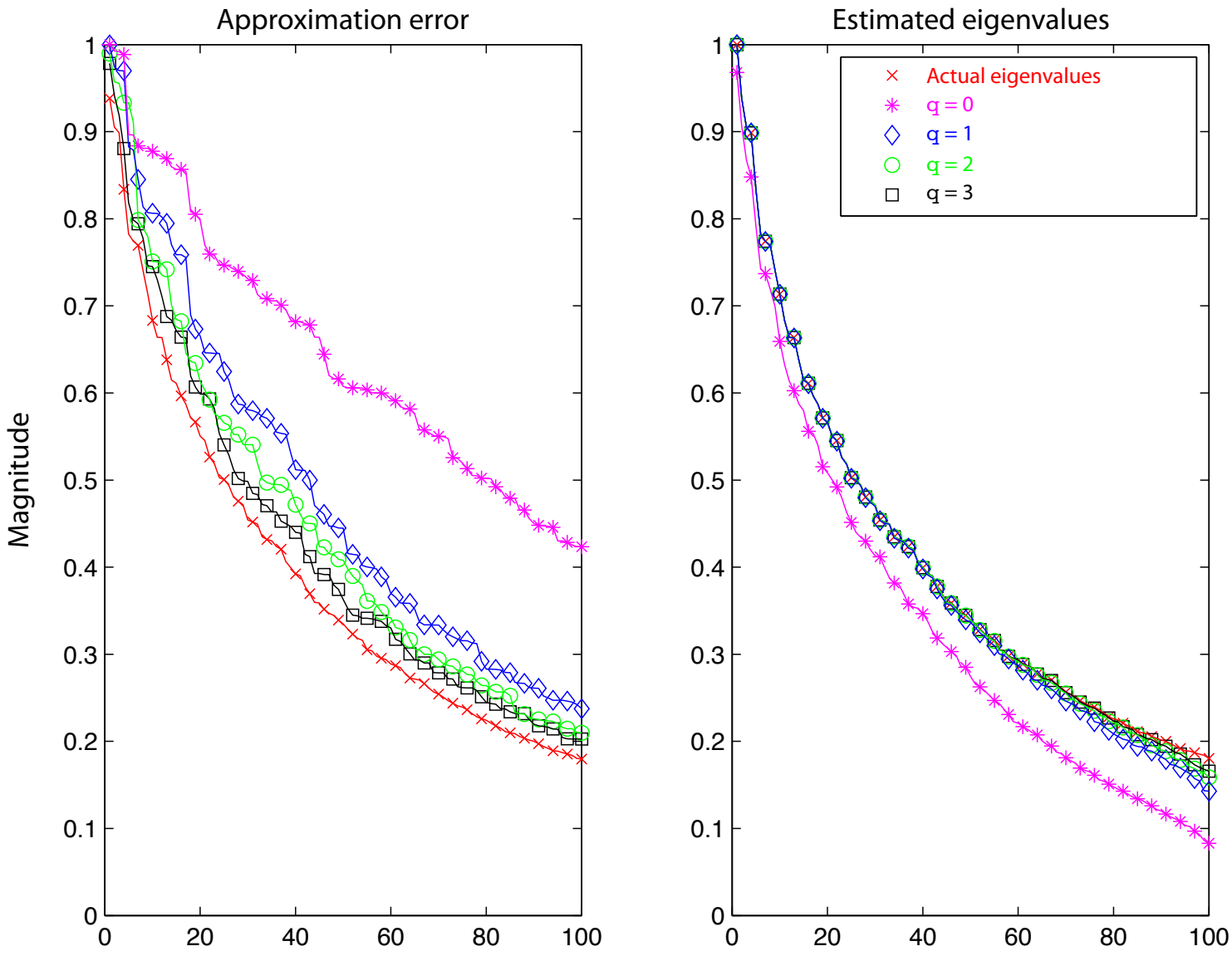
A Graph Laplacian

- Graph Laplacian on a point cloud of 9×9 image patches
- Form $9,025 \times 9,025$ symmetric data matrix $\boldsymbol{A}$
- **Total storage:** 311 Megabytes (uncompressed)

- The eigenvectors give information about graph structure
- Attempt to compute first 100 eigenvalues/vectors using power scheme



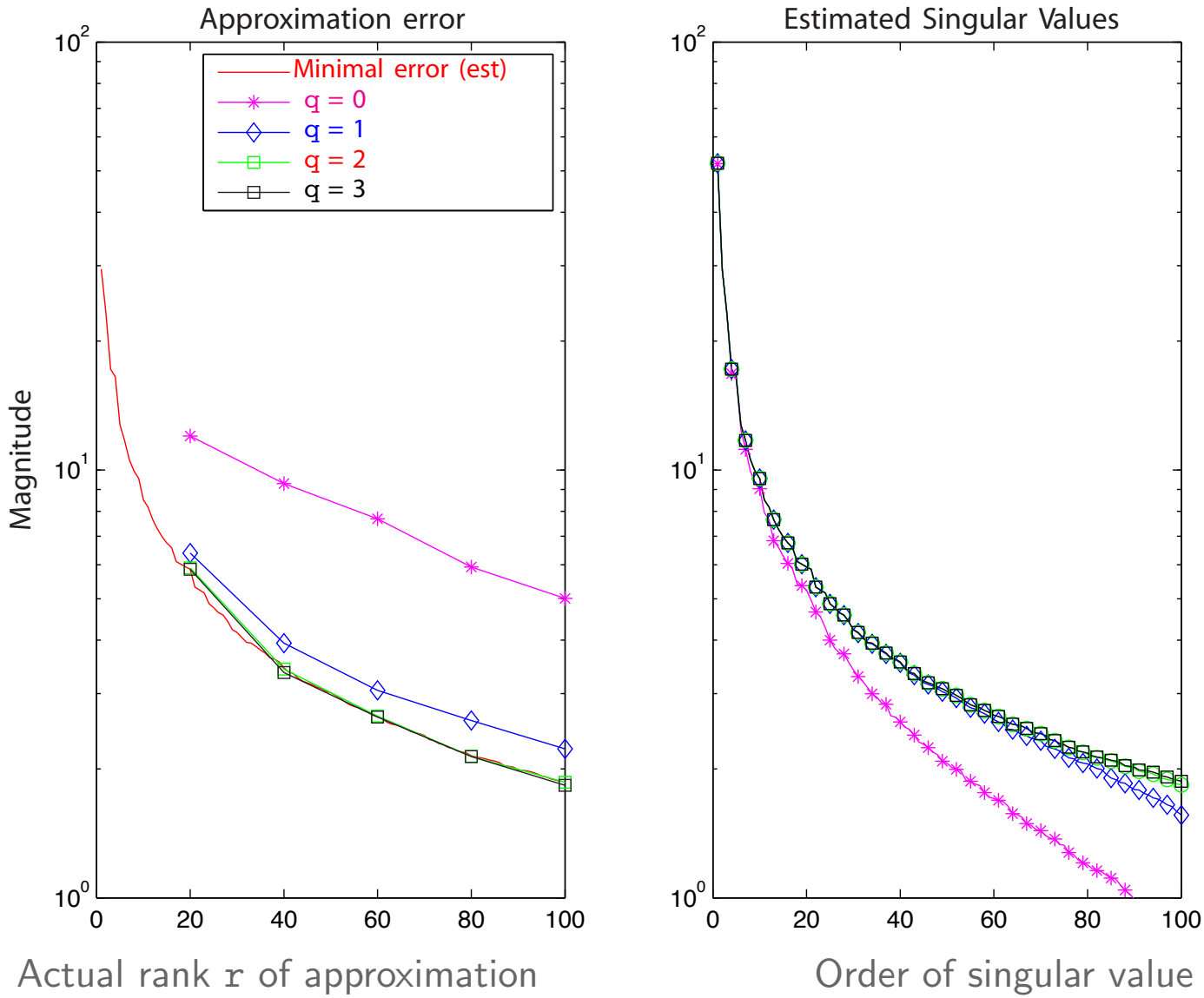
[Ref] Courtesy of Gunnar Martinsson & François Meyer.


Approximation error
Magnitude
Actual rank r of approximation
Estimated eigenvalues
Actual eigenvalues
q = 0
q = 1
q = 2
q = 3
Order of eigenvalues

Eigenfaces

- Database consists of $7,254$ photographs with $98,304$ pixels each
- Form $98,304 \times 7,254$ data matrix $\widetilde{\boldsymbol{A}}$
- **Total storage:** 5.4 Gigabytes (uncompressed)
- Center each column and scale to unit norm to obtain $\boldsymbol{A}$
- The dominant left singular vectors of $\boldsymbol{A}$ are called **eigenfaces**
- Attempt to compute first 100 eigenfaces using power scheme

[Ref] Image from Scholarpedia article "Eigenfaces," accessed 12 October 2009



[Ref] Halko et al. 2011, §7.3.

Resources

Articles with Broad Scope

- M. Mahoney, "Randomized algorithms for matrices and data," *Foundations & Trends in Machine Learning*, 2011

- N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness...," *SIAM Rev.*, 2011

- J. A. Tropp, "User-Friendly Tools for Random Matrices." Submitted to *Foundations & Trends in Machine Learning*, 2014

High-Quality Software for Randomized NLA

- Mark Tygert.
  `http://tygert.com/software.html`

- Haim Avron.
  `http://researcher.watson.ibm.com/researcher/view_group.php?id=5131`

- Xiangrui Meng.
  `http://www.stanford.edu/~mengxr/pub/lsrn.htm`

To learn more...

E-mail: jtropp@cms.caltech.edu

Web: http://users.cms.caltech.edu/~jtropp