Randomized Numerical Linear Algebra

http://bicmr.pku.edu.cn/~wenzw/bigdata2018.html

Acknowledgement: this slides is based on Prof. Petros Drineas and Prof. Michael W. Mahoney’s, Prof. Gunnar Martinsson lecture notes
Outline

1. Randomized Numerical Linear Algebra (RandNLA)
2. Approximating Matrix Multiplication
3. Approximate SVD
4. Random Sampling for SVD
5. Single View Algorithm For Matrix Approximation
Why RandNLA?

Randomization and sampling allow us to design provably accurate algorithms for problems that are:

- **Massive**
  (matrices so large that can not be stored at all, or can only be stored in slow memory devices)

- **Computationally expensive or NP-hard**
  (combinatorial optimization problems such as the Column Subset Selection Problem)
RandNLA: sampling rows/columns

Randomized algorithms

- By (carefully) sampling rows/columns of a matrix, we can construct new, smaller matrices that are close to the original matrix (w.r.t. matrix norms) with high probability.

\[
\begin{pmatrix}
A
\end{pmatrix}
\begin{pmatrix}
B
\end{pmatrix}
\approx
\begin{pmatrix}
C
\end{pmatrix}
\begin{pmatrix}
R
\end{pmatrix}
\]

- By preprocessing the matrix using random projections, we can sample rows/columns much less carefully (uniformly at random) and still get nice bounds with high probability.
RandNLA: sampling rows/columns

Matrix perturbation theory

- The resulting smaller matrices behave similarly (in terms of singular values and singular vectors) to the original matrices thanks to the norm bounds.

Structural results that “decouple” the “randomized” part from the “matrix perturbation” part are important in the analyses of such algorithms.

Interplay

- **Applications in BIG DATA**: (Data Mining, Information Retrieval, Machine Learning, Bioinformatics, etc.)
- **Numerical Linear Algebra**: Matrix computations and linear algebra (ie., perturbation theory)
- **Theoretical Computer Science**: Randomized and approximation algorithms
Issues

Computing large SVDs: computational time

- In commodity hardware (e.g., a 4GB RAM, dual-core laptop), using MatLab 7.0 (R14), the computation of the SVD of the dense 2,240-by-447,143 matrix A takes about 12 minutes.
- Computing this SVD is not a one-liner, since we cannot load the whole matrix in RAM (runs out-of-memory in MatLab).
- We compute the eigendecomposition of $AA^T$.

Obviously, running time is a concern.

Machine-precision accuracy is NOT necessary!

- Data are noisy.
- Approximate singular vectors work well in our settings.
Issues

- Selecting good columns that “capture the structure” of the top principal components
  - Combinatorial optimization problem; hard even for small matrices.
  - Often called the Column Subset Selection Problem (CSSP).
  - Not clear that such columns even exist.

The two issues:
- Fast approximation to the top k singular vectors of a matrix, and
- Selecting columns that capture the structure of the top k singular vectors

are connected and can be tackled using the same framework
SVD decomposes a matrix as

\[
\begin{pmatrix}
A
\end{pmatrix}
\approx
\begin{pmatrix}
U_k
\end{pmatrix}
\begin{pmatrix}
X
\end{pmatrix}
\]

- It is easy to see that \(X = U_k A\)
- The SVD has strong optimality properties
- \(U_k\): Top \(k\) left singular vectors. The columns of \(U_k\) are linear combinations of up to all columns of \(A\).
The CX decomposition

Mahoney & Drineas (2009) PNAS

\[
\begin{pmatrix}
m \\
\end{pmatrix} \times \begin{pmatrix}
n \\
\end{pmatrix} \quad \approx \quad \begin{pmatrix}
m \\
\end{pmatrix} \times \begin{pmatrix}
c \\
\end{pmatrix} \quad \begin{pmatrix}
c \\
\end{pmatrix} \times \begin{pmatrix}
n \\
\end{pmatrix}
A
C
X
\]

Goal: make (some norm) of \( A - CX \) small.

\( C \): \( c \) columns of \( A \), with \( c \) being as close to \( k \) as possible

Moore-Penrose pseudoinverse of \( A \):

\[
AA^\dagger A = A, \quad A^\dagger AA^\dagger = A^\dagger, \quad (AA^\dagger)^* = AA^\dagger, \quad (A^\dagger A)^* = A^\dagger A
\]
The CX decomposition

\[
\begin{pmatrix}
A
\end{pmatrix}
\approx
\begin{pmatrix}
C
\end{pmatrix}
\begin{pmatrix}
X
\end{pmatrix}
\]

- Easy to prove that optimal \( X = C^\dagger A \). (with respect to unitarily invariant norms; \( C^\dagger \) is the Moore-Penrose pseudoinverse of \( C \)). Thus, the challenging part is to find good columns of \( A \) to include in \( C \).

- From a mathematical perspective, this is a combinatorial optimization problem, closely related to the so-called Column Subset Selection Problem (CSSP); the CSSP has been heavily studied in Numerical Linear Algebra.
We would like to get theorems of the following form

Given an m-by-n matrix $A$, there exists an efficient algorithm that picks a small number of columns of $A$ such that with reasonable probability:

$$
\|A - CX\|_F = \|A - CC^\dagger A\|_F \leq (1 + \epsilon) \|A - A_k\|_F
$$

- Best rank-k approximation to $A$: $A_k = U_k U_k^\top A$
Outline

1. Randomized Numerical Linear Algebra (RandNLA)
2. Approximating Matrix Multiplication
3. Approximate SVD
4. Random Sampling for SVD
5. Single View Algorithm For Matrix Approximation
Approximating Matrix Multiplication

Problem Statement

Given an $m$-by-$n$ matrix $A$ and an $n$-by-$p$ matrix $B$, approximate the product $AB$, or equivalently, approximate the sum of $n$ rank-one matrices

$$AB = \sum_{i=1}^{n} \begin{pmatrix} A^{(i)} \end{pmatrix} \begin{pmatrix} B^{(i)} \end{pmatrix} \in \mathbb{R}^{m \times p}$$

- $A^{(i)}$ the $i$-th column of $A$
- $B^{(i)}$ the $i$-th row of $B$
- Each term in the summation is a rank-one matrix
A sampling approach

\[
AB = \sum_{i=1}^{n} \begin{pmatrix} A^{(i)} \\ B^{(i)} \end{pmatrix} \in \mathbb{R}^{m \times p}
\]

Algorithm

- Fix a set of probabilities \( p_i, i = 1, \ldots, n \), summing up to 1.

- For \( t = 1, \ldots, c \),
  - set \( j_t = i \), where \( P(j_t = i) = p_i \).
  (Pick \( c \) terms of the sum, with replacement, with respect to the \( p_i \).)

- Approximate the product \( AB \) by summing the \( c \) terms, after scaling.
Consider a discrete random variable with possible values $c_1 < \ldots < c_n$. The probability attached to $c_i$ is $p_i$. Let

$$q_0 = 0, \quad q_i = \sum_{j=1}^{i} p_j.$$ 

They are the cumulative probabilities associated with $c_i$, i.e., $q_i = F(c_i)$.

To sample this distribution

- generate a uniform $U$
- find $K \in \{1, \ldots, n\}$ such that $q_{K-1} < U < q_K$
- set $X = c_K$
With/without replacement

- **Sampling with replacement:**
  Each data unit in the population is allowed to appear in the sample more than once.
  It is easy to analyze mathematically.

- **Sampling without replacement:**
  Each data unit in the population is allowed to appear in the sample no more than once.
A sampling approach

\[
AB = \sum_{i=1}^{n} \left( A^{(i)} \right) \left( B^{(i)} \right) \\
\in \mathbb{R}^{m \times p}
\]

\[
\approx \frac{1}{c} \sum_{t=1}^{c} \frac{1}{p_{j_t}} \left( A^{(j_t)} \right) \left( B^{(j_t)} \right) \\
\in \mathbb{R}^{m \times p}
\]

Keeping the terms \( j_1, j_2, \ldots, j_c \)
The algorithm (matrix notation)

\[
\begin{pmatrix}
  m 	imes n \\
  A
\end{pmatrix}
\begin{pmatrix}
  n 	imes p \\
  B
\end{pmatrix}
\approx
\begin{pmatrix}
  m 	imes c \\
  C
\end{pmatrix}
\begin{pmatrix}
  c 	imes p \\
  R
\end{pmatrix}
\]

Algorithm:

- Pick \( c \) columns of \( A \) to form an \( m \)-by-\( c \) matrix \( C \) and the corresponding \( c \) rows of \( B \) to form a \( c \)-by-\( p \) matrix \( R \).

- Approximate \( AB \) by \( CR \).

Note

- We pick the columns and rows with non-uniform probabilities.
- We scale the columns (rows) prior to including them in \( C(R) \).
The algorithm (matrix notation)

\[
\begin{pmatrix}
  m \times n \\
  A
\end{pmatrix}
\begin{pmatrix}
  n \times p \\
  B
\end{pmatrix}
\approx
\begin{pmatrix}
  m \times c \\
  C
\end{pmatrix}
\begin{pmatrix}
  c \times p \\
  R
\end{pmatrix}
\]

Algorithm:

- Create \( C \) and \( R \) by performing \( c \) i.i.d. trials, with replacement.

- For \( t = 1, \ldots, c \), pick a column \( A^{(j_t)} \) and a row \( B^{(j_t)} \) with probability

\[
P(j_t = i) = \frac{\|A^{(i)}\|_2 \|B^{(i)}\|_2}{\sum_{j=1}^n \|A^{(j)}\|_2 \|B^{(j)}\|_2}
\]

- Include \( A^{(j_t)}/(cp_{j_t})^{1/2} \) as a column of \( C \), and \( B^{(j_t)}/(cp_{j_t})^{1/2} \) as a row of \( R \)
We can also use the sampling matrix notation:
Let $S$ be an $n$-by-$c$ matrix whose $t$-th column (for $t = 1, \ldots, c$) has a single non-zero entry, namely

$$S_{jt} = \frac{1}{\sqrt{cp_{jt}}}$$

Clearly:

$$AB \approx CR = (AS)(S^T B)$$

Note: $S$ is sparse (has exactly $c$ non-zero elements, one per column).
Simple Lemmas

- It is easy to implement this particular sampling in two passes.
- The expectation of CR (element-wise) is AB (unbiased estimator), regardless of the sampling probabilities.
- Our particular choice of sampling probabilities minimizes the variance of the estimator (w.r.t. the Frobenius norm of the error AB-CR).
A bound for the Frobenius norm

For the above algorithm,

$$ \mathbb{E}[\|AB - CR\|_F] = \mathbb{E}[\|AB - ASS^T B\|_F] \leq \frac{1}{c} \|A\|_F \|B\|_F $$

- prove using elementary manipulations of expectation
- Measure concentration follows from a martingale argument.
- The above bound also implies an upper bound for the spectral norm of the error $AB - CR$. 
Proofs

Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times p}$, $1 \leq c \leq n$, and $p_i \geq 0$, $\sum_i p_i = 1$. Then

$$
\mathbb{E}[(CR)_{ij}] = (AB)_{ij}, \quad \text{Var}[(CR)_{ij}] = \frac{1}{c} \sum_{i=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c} (AB)_{ij}^2
$$

- Define $X_t = \left( \frac{A^{(it)} B^{(it)}}{cp_{it}} \right)_{ij} = \frac{A_{it} B_{ij}}{cp_{it}}$. Then

  $$
  \mathbb{E}[X_t] = \sum_{k=1}^{n} p_k \frac{A_{ik} B_{kj}}{cp_k} = \frac{1}{c} (AB)_{ij} \quad \text{and} \quad \mathbb{E}[X_t^2] = \sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{c^2 p_k}
  $$

- $\mathbb{E}[(CR)_{ij}] = \sum_{t=1}^{c} \mathbb{E}[X_t] = (AB)_{ij}$

  $$
  \text{Var}[X_t] = \mathbb{E}[X_t^2] - \mathbb{E}[X_t]^2 = \sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{c^2 p_k} - \frac{1}{c^2} (AB)_{ij}^2
  $$
Proofs

Lemma:
\[ E[\| AB - CR \|_F^2] = \sum_{k=1}^{n} \frac{|A^{(k)}|^2 |B^{(k)}|^2}{cp_k} - \frac{1}{c} \| AB \|_F^2 \]

Proof:

\[
E[\| AB - CR \|_F^2] = \sum_{i=1}^{n} \sum_{j=1}^{p} E[(AB - CR)_{ij}^2] = \sum_{i=1}^{n} \sum_{j=1}^{p} \text{Var}[(CR)_{ij}]
\]

\[
= \frac{1}{c} \sum_{k=1}^{n} \frac{1}{p_k} \left( \sum_i A_{ik}^2 \right) \left( \sum_i B_{kj}^2 \right) - \frac{1}{c} \| AB \|_F^2
\]

\[
= \frac{1}{c} \sum_{k=1}^{n} \frac{1}{p_k} |A^{(k)}|^2 |B^{(k)}|^2 - \frac{1}{c} \| AB \|_F^2
\]
Proofs

- Find $p_k$ to minimize $\mathbb{E} [ \| AB - CR \|_F^2 ]$:

$$\min_{\sum_{k=1}^n p_k = 1} f(p_1, \ldots, p_n) = \sum_{k=1}^n \frac{1}{p_k} |A^{(k)}|^2 |B^{(k)}|^2$$

- Introduce $L = f(p_1, \ldots, p_n) + \lambda (\sum_{k=1}^n p_k - 1)$ and solve $\frac{\partial L}{\partial p_i} = 0$

- It gives $p_k = \frac{|A^{(k)}||B^{(k)}|}{\sum_{k'=1}^n |A^{(k')}||B^{(k')}|}$. Then

$$\mathbb{E} [ \| AB - CR \|_F^2 ] = \frac{1}{c} \left( \sum_{k=1}^n |A^{(k)}||B^{(k)}| \right)^2 - \frac{1}{c} \| AB \|_F^2 \leq \frac{1}{c} \| A \|_F^2 \| B \|_F^2$$
Special case: $B = A^T$

If $B = A^T$, then the sampling probabilities are

$$P(j_t = i) = \frac{\|A^{(i)}\|^2}{\|A\|^2_F}$$

Also, $R = C^T$, and the error bounds are:

$$E[\|AA^T - CC^T\|_F] = E[\|AA^T - ASS^TA^T\|_F] \leq \frac{1}{c} \|A\|^2_F$$
Special case: $B = A^T$

A better spectral norm bound via matrix Chernoff/Bernstein inequalities:

Assumptions:

- Spectral norm of $A$ is one (not important, just normalization)
- Frobenius norm of $A$ is at least 0.2 (not important, simplifies bounds).
- Important: Set
  \[ c = \Omega \left( \frac{\|A\|_F^2}{\epsilon^2} \ln \left( \frac{\|A\|_F^2}{\epsilon^2 \sqrt{\delta}} \right) \right) \]

Then: for any $0 < \epsilon < 1$ with probability at least $1 - \delta$

\[ \mathbb{E}[\|AA^T - CC^T\|_F] = \mathbb{E}[\|AA^T - ASS^T A^T\|_F] \leq \epsilon \]
Special case: $B = A^T$

Notes:

- The constants hidden in the big-Omega notation are small.
- We need a sufficiently large value of $c$, otherwise the theorem does not work.
Using a dense S

We approximated the product $AB$ as follows:

$$AB \approx CR = (AS)(S^T B)$$

- Recall that $S$ is an $n$-by-$c$ sparse matrix (one non-zero entry per column).
- Let’s replace $S$ by a dense matrix, the random sign matrix:

$$S_{ij} = \begin{cases} +1/\sqrt{c}, & \text{w.p. 1/2} \\ -1/\sqrt{c}, & \text{w.p. 1/2} \end{cases}$$

- The stable rank of $A$: $st(A) = \|A\|_F^2/\|A\|_2^2$:

$$c = \Omega(\max(st(A), st(B)) \ln(m + p)/\epsilon^2)$$

then, with high probability (see Theorem 3.1 in Magen & Zouzias SODA 2012)

$$\|AB - CR\|_2 = \|AB - ASS^T B\|_2 \leq \epsilon \|A\|_2 \|B\|_2$$
Using a dense $S$ for $B = A^T$

Approximate the product $AA^T$ (assuming that the spectral norm of $A$ is one):

$$AA^T \approx CC^T = (AS)(S^T A^T)$$

- Let $S$ be a dense matrix, the random sign matrix:

  $$S_{ij} = \begin{cases} +1/\sqrt{c}, & \text{w.p. } 1/2 \\ -1/\sqrt{c}, & \text{w.p. } 1/2 \end{cases}$$

- If

  $$c = \Omega \left( \frac{\|A\|_F^2}{\epsilon^2} \ln m \right)$$

  then, with high probability

  $$\|AA^T - CC^T\|_2 = \|AA^T - ASS^T A^T\|_2 \leq \epsilon$$
Using a dense S

- This matrix multiplication approximation is oblivious to the input matrices A and B.

- Reminiscent of random projections and the Johnson-Lindenstrauss (JL) transform.

- Bounds for the Frobenius norm are easier to prove and are very similar to the case where S is just a sampling matrix.

- We need a sufficiently large value for c, otherwise the (spectral norm) theorem does not hold.

- It holds for arbitrary A and B (not just $B = A^T$); the sampling-based approach should also be generalizable to arbitrary A and B.
Recap: approximating matrix multiplication

We approximated the product $AB$ as follows:

$$AB \approx CR = (AS)(S^T B)$$

Let $S$ be a sampling matrix (actual columns from $A$ and rows from $B$ are selected): We need to carefully sample columns of $A$ (rows of $B$) with probabilities that depend on their norms in order to get “good” bounds of the following form:

$$\mathbb{E}[\|AB - CR\|_F] = \mathbb{E}[\|AB - ASS^T B\|_F] \leq \frac{1}{\sqrt{c}} \|A\|_2 \|B\|_2$$

$$\|AA^T - CC^T\|_2 = \|AA^T - ASS^T A^T\|_2 \leq \epsilon$$
Outline

1. Randomized Numerical Linear Algebra (RandNLA)
2. Approximating Matrix Multiplication
3. Approximate SVD
4. Random Sampling for SVD
5. Single View Algorithm For Matrix Approximation
**Problem Statement:**
Given: mxn matrix $A$, and $0 < k < \min(m,n) = n$.
Goal: Compute a rank-$k$ approximation to $A$.

- Fast low-rank matrix approximation is key to efficiency of superfast direct solvers for integral equations and many large sparse linear systems.

- Indispensable tool in mining large data sets.

- Randomized algorithms compute accurate truncated SVD.

- Minimum work and communication/Exceptionally high success rate.
Low-rank Approximation

seek to compute a rank-$k$ approximation with $k \ll n$

$$A \approx U_k X$$

- Eigenvectors corresponding to leading eigenvalues.
- Singular Value Decomposition (SVD) / Principal Component Analysis (PCA).
- Spanning columns or rows.

The problem being addressed is ubiquitous in applications.
SVD - Properties

**Theorem: SVD**

If $A$ is a real $m$-by-$n$ matrix, then there exits

$$U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m} \text{ and } V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times n}$$

such that $U^T U = I$, $V^T V = I$ and

$$U^T A V = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}, \quad p = \min(m, n),$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$.

**Eckart & Young, 1936**

Let the SVD of $A \in \mathbb{R}^{m \times n}$ be given in Theorem: SVD. If $k < r = \text{rank}(A)$ and $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$, then

$$\min_{\text{rank}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}.$$
Applications:

- Semidefinite programming
- Low-rank matrix completion
- Robust principal component analysis
- Sparse principal component analysis
- Sparse inverse covariance matrix estimation
- Nearest correlation matrix estimation
- High dimensional data reduction
- Density functional theory for electronic structure calculation
- Nonlinear eigenvalue problems
- Fast algorithms for elliptic PDEs: more efficient Fast Multipole Methods, fast direct solvers, construction of special quadratures for corners and edges, etc.
Review of existing methods: dense matrix

For a dense $n \times n$ matrix that fits in RAM, excellent algorithms are already part of LAPACK (and incorporated into Matlab, Mathematica, etc).

- Double precision accuracy.
- Very stable.
- $O(n^3)$ asymptotic complexity. Reasonably small constants.
- Require extensive random access to the matrix.
- When the target rank $k$ is much smaller than $n$, there also exist $O(n^2k)$ methods with similar characteristics (the well-known Golub-Businger method, RRQR by Gu and Eisentstat, etc).
- For small matrices, the state-of-the-art is quite satisfactory. (By “small” we mean something like $n \leq 10000$ on today’s computers.)
Review of existing methods: structured matrix

If the matrix is large, but can rapidly be applied to a vector (if it is sparse, or sparse in Fourier space, or amenable to the FMM, etc.), so called Krylov subspace methods often yield excellent accuracy and speed.

**Lanczos-based methods:**

1. From $v \in \mathbb{R}^n$, computes orthonormal basis $V$ for

   $$\mathcal{K}(A, v) = \text{span} \{v, Av, A^2v, \ldots, A^{k-1}v\}$$

2. Rayleigh-Ritz: $\text{eig}(V^TAV) \Rightarrow$ Ritz pairs $\approx$ eigenpairs

3. If “not converged”, update $v$ and go to Step 1.

**Strength and weakness:**

- Most efficient in terms of the number of $Av$ (or SpMv)
- Fast and reliable for computing “not too many” eigenpairs
- Lower concurrency and unable to be warm-started
“New” challenges in algorithmic design

The existing state-of-the-art methods of numerical linear algebra that we have very briefly outlined were designed for an environment where the matrix fits in RAM and the key to performance was to minimize the number of floating point operations required. Currently, communication is becoming the real bottleneck:

- While clock speed is hardly improving at all anymore, the cost of a flop keeps going down rapidly. (Multi-core processors, GPUs, cloud computing, etc.)
- The cost of slow storage (hard drives, flash memory, etc.) is also going down rapidly.
- Communication costs are decreasing, but not rapidly. Moving data from a hard-drive. Moving data between nodes of a parallel machine. (Or cloud computer ... ) The amount of fast cache memory close to a processor is not improving much. (In fact, it could be said to be shrinking — GPUs, multi-core, etc.)
- “Deluge of data”. Driven by ever cheaper storage and acquisition techniques. Web search, data mining in archives of documents
Review of existing randomized methods

- Random column/row selection
  Draw at random some columns and suppose that they span the entire column space. If rows are drawn as well, then spectral properties can be estimated. Crude sampling leads to less than $O(mn)$ complexity, but is very dangerous.

- Sparsification
  Zero out the vast majority of the entries of the matrix. Keep a random subset of entries, and boost their magnitude to preserve “something”.

- Quantization and sparsification
  Restrict the entries of the matrix to a small set of values (-1/0/1 for instance).

- Randomized Subspace iteration
  Random sampling + Rayleigh Ritz procedure
Linear Time SVD Algorithm

- Input: m-by-n matrix A, $1 \leq k \leq c \leq n$, $\{p_i\}_{i=1}^n$ such that $p_i \geq 0$ and $\sum_i p_i = 1$

- Sampling:
  - For $t = 1$ to $c$
    - pick $i_t \in \{1, \ldots, n\}$ with $P(i_t = \alpha) = p_\alpha$
    - Set $C^{(t)} = \frac{A(i_t)}{\sqrt{cp_{i_t}}}$
  - Compute $C^T C$ and its SVD, say $C^T C = \sum_{t=1}^c \sigma_t(C)^2 y_t (y_t)^T$

- Compute $h^t = \frac{C y^t}{\sigma_t(C)}$ for $t = 1, \ldots, k$
  - (Note: $A = U \Sigma V^T$ and $C = H \Sigma C Y^T$)

- Return $H_k$ where $H_k^{(t)} = h^t$ and $\sigma_t(C)$ for $t = 1, \ldots, k$

The left singular vectors of C are with high probability approximations to the left singular vectors of A
Extract approximate SVD

Given \( A \). Let \( X \) be an approximation of the left singular vectors of \( A \) corresponding to \( k \) largest singular values

\[
\begin{align*}
\text{method} & = 2; \quad \% = 1 \text{ or } 2 \\
Y & = (X' \ast A)'; \quad \% \ Y = A' \ast X; \\
\text{switch method}
\quad \text{case 1;}
\quad \quad [V, S, W] & = \text{svd}(Y, 0); \\
\quad \quad U & = X \ast W; \\
\quad \text{case 2;}
\quad \quad [V, R] & = \text{qr}(Y, 0); \\
\quad \quad [W, S, Z] & = \text{svd}(R'); \\
\quad \quad U & = X \ast W; \quad V = V \ast Z;
\end{align*}
\]

\text{end}

The pair \((U, S, V)\) is an approximation of the \( k \)-dominant SVD
Main theoretical results

Let $H_k$ be constructed the linear Time SVD

$$E[\|A - H_k H_k^T A\|_F^2] \leq \|A - A_k\|_F^2 + \epsilon\|A\|_F^2$$

- Exact SVD of $A = U \Sigma V^T$, $A_k = U_k \Sigma_k V_k^T = U_k U_k^T A = AV_k V_k^T$. 
  - $\min_{\text{rank}(B) \leq k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}(A)$
  - $\min_{\text{rank}(B) \leq k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = \sum_{t=k+1}^{\infty} \sigma_t^2(A)$

- Perturbation theory of matrices

  $$\max_{1 \leq t \leq n} |\sigma_t(A + E) - \sigma_t(A)| \leq \|E\|_2, \quad \sum_{k=1}^{n} (\sigma_k(A + E) - \sigma_k(A))^2 \leq \|E\|_F^2$$

  the latter is known as Hoffman-Wielandt inequality

- Exact SVD of $C = H \Sigma C Y^T$
Proofs

Lemma:

\[
\|A - H_k H_k^T A\|_F^2 \leq \|A - A_k\|_F^2 + 2\sqrt{k}\|AA^T - CC^T\|_F
\]

\[
\|A - H_k H_k^T A\|_2^2 \leq \|A - A_k\|_2^2 + 2\|AA^T - CC^T\|_2
\]

Proof of the first inequality

- \(\|X\|_F^2 = \text{Tr}(X^T X)\) and \(\text{Tr}(X + Y) = \text{Tr}(X) + \text{Tr}(Y)\)

\[
\|A - H_k H_k^T A\|_F^2 = \text{Tr}((A - H_k H_k^T A)^T (A - H_k H_k^T A))
\]

\[
= \text{Tr}(A^T A) - \text{Tr}(A^T H_k H_k^T A) = \|A\|_F^2 - \|A^T H_k\|_F^2
\]

- Using Cauchy-Schwartz inequality:

\[
\left|\|A^T H_k\|_F^2 - \sum_{t=1}^k \sigma_t^2(C)\right| \leq \sqrt{k} \left(\sum_{t=1}^k (|A^T h^t|^2 - \sigma_t^2(C))^2\right)^{1/2}
\]

\[
= \sqrt{k} \left(\sum_{t=1}^k (|A^T h^t|^2 - |C^T h^t|^2)^2\right)^{1/2} = \sqrt{k} \left(\sum_{t=1}^k ((h^t)^T (AA^T - CC^T) h^t)^2\right)^{1/2}
\]

\[
\leq \sqrt{k}\|AA^T - CC^T\|_F
\]
Proofs

- by Hoffman-Wielandt inequality

\[
\left| \sum_{t=1}^{k} \sigma_i^2(C) - \sum_{t=1}^{k} \sigma_i^2(A) \right| \leq \sqrt{k} \left( \sum_{t=1}^{k} (\sigma_i^2(C) - \sigma_i^2(A))^2 \right)^{1/2}
\]

\[
= \sqrt{k} \left( \sum_{t=1}^{k} (\sigma_t(C C^T) - \sigma_t(A A^T))^2 \right)^2
\]

\[
\leq \sqrt{k} \left( \sum_{t=1}^{m} (\sigma_t(C C^T) - \sigma_t(A A^T))^2 \right)^2 \leq \sqrt{k} \|C C^T - A A^T\|_F
\]

- Therefore

\[
\left| \|A^T H_k\|_F^2 - \sum_{t=1}^{k} \sigma_i^2(A) \right| \leq 2 \sqrt{k} \|A A^T - C C^T\|_F
\]
Proofs

• matrix approximation gives

$$\mathbb{E}[\|AB - CR\|_F^2] \leq \frac{1}{c} \|A\|_F^2 \|B\|_F^2$$

which yields

$$2\sqrt{k}\mathbb{E}[\|AA^T - CC^T\|_F] \leq \left(\frac{4k}{c}\right)^{1/2} \|A\|_F^2$$

•

$$\|A^T H_k\|_F^2 \geq \sum_{t=1}^{k} \sigma_t^2(A) - 2\sqrt{k}\|AA^T - CC^T\|_F$$

• If $c \geq 4k/\epsilon^2$, then

$$\mathbb{E}[\|A - H_k H_k^T A\|_F^2] \leq \|A\|_F^2 - \sum_{t=1}^{k} \sigma_t^2(A) + 2\sqrt{k}\mathbb{E}[\|AA^T - CC^T\|_F]$$

$$\leq \|A - A_k\|_F^2 + \epsilon \|A\|_F^2$$
Back to the CX decomposition

We would like to get theorems of the following form

Given an m-by-n matrix A, there exists an efficient algorithm that picks a small number of columns of A such that with reasonable probability:

\[
\|A - CX\|_F = \|A - CC^\dagger A\|_F \leq (1 + \epsilon) \|A - A_k\|_F
\]

Let’s start with a simpler, weaker result, connecting the spectral norm of A-CX to matrix multiplication.
Approximating singular vectors

Sample $c (=140)$ columns of the original matrix $A$ and rescale them appropriately to form a 512-by-$c$ matrix $C$.

Show that $A - CX$ is “small”.

($C^\dagger$ is the pseudoinverse of $C$ and $X = C^\dagger A$)
Approximating singular vectors

Sample c (=140) columns of the original matrix A and rescale them appropriately to form a 512-by-c matrix C.

Show that $A - CX$ is “small”.

$(C^\dagger$ is the pseudoinverse of C and $X = C^\dagger A$)
The fact that $AA^T - CC^T$ is small will imply that $A - CX$ is small as well.
Proof (spectral norm)

Using the triangle inequality and properties of norms,

\[
\|A - CC^\dagger A\|_2^2 = \|(I - CC^\dagger)A\|_2^2 \\
= \|(I - CC^\dagger)AA^T(I - CC^\dagger)^T\|_2 \\
= \|(I - CC^\dagger)(AA^T - CC^\dagger)(I - CC^\dagger)^T\|_2 \\
\leq \|AA^T - CC^\dagger\|_2
\]

- \(I - CC^\dagger\) is a projection matrices
- \((I - CC^\dagger)CC^\dagger = 0\)
Proof (spectral norm)

Assume that our sampling is done in \( c \) i.i.d. trials and the sampling probabilities are:

\[
P(j_t = i) = \frac{\|A(i)\|^2_2}{\|A\|^2_F}
\]

We can use our matrix multiplication result: (We will upper bound the spectral norm by the Frobenius norm to avoid concerns about \( c \), namely whether \( c \) exceeds the threshold necessitated by the theory.)

\[
E[\|A - CC^\dagger A\|_2] \leq E[\|AA^T - CC^T\|_2] \leq \frac{1}{c^{1/4}} \|A\|_F
\]
Is this a good bound?

\[ \mathbb{E}[\|A - CC^\dagger A\|_2] \leq \mathbb{E}[\|AA^T - CC^T\|_2] \leq \frac{1}{c^{1/4}} \|A\|_F \]

- **Problem 1:** If \( c = n \) we do not get zero error. That’s because of sampling with replacement. (We know how to analyze uniform sampling without replacement, but we have no bounds on non-uniform sampling without replacement.)

- **Problem 2:** If \( A \) had rank exactly \( k \), we would like a column selection procedure that drives the error down to zero when \( c = k \). This can be done deterministically simply by selecting \( k \) linearly independent columns.

- **Problem 3:** If \( A \) had numerical rank \( k \), we would like a bound that depends on the norm of \( A - A_k \) and not on the norm of \( A \). Such deterministic bounds exist when \( c = k \) and depend on \( (k(n - k))^{1/2} \|A - A_k\|_2 \)
Relative-error Frobenius norm bounds

Given an m-by-n matrix $A$, there exists an $O(mn^2)$ algorithm that picks $O\left(\left(\frac{k}{\epsilon^2}\right) \ln\left(\frac{k}{\epsilon^2}\right)\right)$ columns of $A$ such that with probability at least 0.9

$$\|A - CX\|_F = \|A - CC^\dagger A\|_F \leq (1 + \epsilon)\|A - A_k\|_F$$
The algorithm

- Input: m-by-n matrix $A$, $0 < \epsilon < 0.5$, the desired accuracy $C$
- $C$, the matrix consisting of the selected columns

**Sampling algorithm**

- Compute probabilities $p_j$ summing to 1
- Let $c = O\left((k/\epsilon^2) \ln(k/\epsilon^2)\right)$.
- In $c$ i.i.d. trials pick columns of $A$, where in each trial the $j$-th column of $A$ is picked with probability $p_j$.
- Let $C$ be the matrix consisting of the chosen columns.

**Note**: there is no rescaling of the columns of $C$ in this algorithm; however, since our error matrix is $A - CX = A - CC^\dagger A$, rescaling the columns of $C$ (as we did in our matrix multiplication algorithms), does not change $A - CX = A - CC^\dagger A$. 
Towards a relative error bound

Structural result (deterministic):

\[ \|A - CX\|_F = \|A - CC^\dagger A\|_F \leq \|A - A_k\|_F + \|(A - A_k)S(V_k^TS)^\dagger\|_F \]

- This holds for any n-by-c matrix S such that \( C = AS \) as long as the k-by-c matrix \( V_k^TS \) has full rank (equal to k).

- The proof of the structural result critically uses the fact that with \( X = C^\dagger A \) is the argmin for any unitarily invariant norm of the error \( A - CX \).

- Variants of this structural result have appeared in various papers.
The rank of $V^T_k S$

Structural result (deterministic):

$$\|A - CX\|_F = \|A - CC^\dagger A\|_F \leq \|A - A_k\|_F + \|(A - A_k)S(V^T_k S)^\dagger\|_F$$

- This holds for any n-by-c matrix S such that $C = AS$ as long as the k-by-c matrix $V^T_k S$ has full rank (equal to k).

- S be a sampling and rescaling matrix, where the sampling probabilities are the leverage scores: our matrix multiplication results (and the fact that the square of the Frobenius norm of $V_k$ if equal to k) guarantee that, for our choice of c (with constant probability):

$$\|V^T_k V_k - V^T_k SS^T V_k\|_2 = \|I_k - V^T_k SS^T V_k\|_2 \leq \epsilon$$
The rank of $V_k^T S$

From matrix perturbation theory, if

$$\|V_k^TV_k - V_k^TSS^TV_k\|_2 = \|I_k - V_k^TSS^TV_k\|_2 \leq \epsilon$$

it follows that all singular values ($\sigma_i$) of $V_k^T S$ satisfy

$$\sqrt{1 - \epsilon} \leq \sigma_i(V_k^T S) \leq \sqrt{1 + \epsilon}$$

By choosing $\epsilon$ small enough, we can guarantee that $V_k^T S$ has full rank (with constant probability).

$$\|(A - A_k)S(V_k^T S)^\dagger\|_F \leq \|(A - A_k)S\|_F \|(V_k^T S)^\dagger\|_2$$

$$= \sigma_{\min}^{-1}(V_k^T S) \|(A - A_k)S\|_F$$
Bounding the second term

To conclude:

\[
\| \left( A - A_k \right) S (V_k^T S)^\dagger \|_F \leq \| \left( A - A_k \right) S \|_F \| (V_k^T S)^\dagger \|_2 \\
= \sigma_{\min}^{-1} (V_k^T S) \| \left( A - A_k \right) S \|_F
\]

- We already have a bound for all singular values of $V_k^T S$ (go back two slides).

- Prove, using our sampling and rescaling,

\[
\mathbb{E} \left[ \| \left( A - A_k \right) S \|_F^2 \right] = \| A - A_k \|_F^2
\]

Collecting, we get a $(2 + \epsilon)$ constant-factor approximation.

A more careful (albeit, longer) analysis can improve the result to a $(1 + \epsilon)$ relative error approximation.
Using a dense $S$

Our proof would also work if instead of the sampling matrix $S$, we used, for example, the dense random sign matrix $S$:

$$S_{ij} = \begin{cases} 
    +1/\sqrt{c}, & \text{w.p. } 1/2 \\ 
    -1/\sqrt{c}, & \text{w.p. } 1/2 
\end{cases}$$

The intuition is clear: the most critical part of the proof is based on approximate matrix multiplication to bound the singular values of $V_k^T S$. This also works when $S$ is a dense matrix.
Selecting fewer columns

**Problem:**
How many columns do we need to include in the matrix C in order to get relative-error approximations?

**Recall:** with $O((k/\epsilon^2)\log(k/\epsilon^2))$ columns, we get (subject to a failure probability)

$$
\|A - CC^\dagger A\|_F \leq (1 + \epsilon)\|A - A_k\|_F
$$

Deshpande & Rademacher (FOCS’10): with exactly k columns, we get

$$
\|A - CC^\dagger A\|_F \leq \sqrt{k}\|A - A_k\|_F
$$

What about the range between $k$ and $O(k \log k)$?
Selecting fewer columns

(Boutsidis, Drineas, & Magdon-Ismail, FOCS 2011)

**Question:**
What about the range between $k$ and $O(k \log k)$?

**Answer:**
A relative-error bound is possible by selecting $c = 3k/\epsilon$ columns

**Technical breakthrough:**
A combination of sampling strategies with a novel approach on column selection, inspired by the work of Batson, Spielman, & Srivastava (STOC ’09) on graph sparsifiers.

- The running time is $O((mnk + nk^3)\epsilon^{-1})$.
- Simplicity is gone . . .
Outline

1. Randomized Numerical Linear Algebra (RandNLA)
2. Approximating Matrix Multiplication
3. Approximate SVD
4. Random Sampling for SVD
5. Single View Algorithm For Matrix Approximation
Given an $m \times n$ matrix $A$ and an integer $k < \min(m, n)$, find an orthonormal $m \times k$ matrix $Q$ such that

$$A \approx QQ^T A$$

Solving the primitive problem via randomized sampling — intuition

- Draw random vectors $r_1, r_2, \ldots, r_k \in \mathbb{R}^n$.
- Form “sample” vectors $y_1 = Ar_1, y_2 = Ar_2, \ldots, y_k = Ar_k \in \mathbb{R}^m$.
- Form orthonormal vectors $q_1, q_2, \ldots, q_k \in \mathbb{R}^m$ such that

$$\text{span}\{q_1, q_2, \ldots, q_k\} = \text{span}\{y_1, y_2, \ldots, y_k\}$$

Almost always correct if $A$ has exact rank $k$
Low-Rank Approximation: Randomized Sampling

Algorithm RandSam0

Input: mxn matrix A, int k, p.

- Draw a random $n \times (k + p)$ matrix $\Omega$
- Compute $QR = A\Omega$
- and SVD: $Q^T A = \hat{U}\hat{\Sigma}\hat{V}^T$
- Truncate SVD: $\hat{U}_k\hat{\Sigma}_k\hat{V}_k^T$

Output: $B = (Q\hat{U}_k)\hat{\Sigma}_k\hat{V}_k^T$

- Easy to implement.
- Very efficient computation.
- Minimum communication.
error for Gaussian test matrices


- Let $A$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.
- Let $k$ denote a target rank and let $p$ denote an over-sampling parameter.
- Let $\Omega$ denote an $n \times (k+p)$ Gaussian matrix.
- Let $Q$ denote the $m \times (k+p)$ matrix $Q = \text{orth}(A\Omega)$.

If $p \geq 4$, then

$$\|A - QQ^*A\|_2 \leq \left( 1 + 6\sqrt{(k+p)p \log p} \right) \sigma_{k+1} + 3\sqrt{k+p} \left( \sum_{j>k} \sigma_j^2 \right)^{1/2}$$

except with probability at most $3p^{-p}$.
Algorithm RandSam1

Input: mxn matrix A, int k, p, c.
- Draw a random \( n \times (k + p + c) \) matrix \( \Omega \)
- Compute \( QR = A\Omega \)
- and SVD: \( Q^TA = \hat{U}\hat{\Sigma}\hat{V}^T \)
- Truncate SVD: \( \hat{U}_k\hat{\Sigma}_k\hat{V}_k^T \)

Output: \( B = (Q\hat{U}_k)\hat{\Sigma}_k\hat{V}_k^T \)

- Only change from RandSam0: p becomes p + c
- Smallest modification of any algorithm.
- c allows a drastically different error bound, controls accuracy.
- p remains in control of failure chance.
Randomized Power Method

Algorithm RandSam2

Input: mxn matrix A, int k, p, c, q

- Draw a random $n \times (k + p + c)$ matrix $\Omega$
- Compute $QR = (AA^T)^qA\Omega$
- and SVD: $Q^TA = \hat{U}\hat{\Sigma}\hat{V}^T$
- Truncate SVD: $\hat{U}_k\hat{\Sigma}_k\hat{V}_k^T$

Output: $B = (Q\hat{U}_k)\hat{\Sigma}_k\hat{V}_k^T$

- QR needs done carefully for numerical accuracy.
- Algorithm is old one when $q = 0$; but $q = 1$ far more accurate.
- Should converge faster when singular values do not decay very fast.
Example 1

We consider a $1000 \times 1000$ matrix $A$ whose singular values are shown below:

A is a discrete approximation of a certain compact integral operator normalized so that $\|A\| = 1$. Curiously, the nature of $A$ is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$. 

The red line indicates the singular values $\sigma_{k+1}$ of $A$. These indicate the theoretically minimal approximation error.

The blue lines indicate the actual errors $e_k$ incurred by 20 instantiations of the proposed method.
Example 2

We consider a $1000 \times 1000$ matrix $A$ whose singular values are shown below:

A is a discrete approximation of a certain compact integral operator normalized so that $\|A\| = 1$. Curiously, the nature of $A$ is in a strong sense irrelevant: the error distribution depends only on $\left\{ \sigma_j \right\}_{j=1}^{\min(m, n)}$. 

The red line indicates the singular values $\sigma_{k+1}$ of $A$. These indicate the theoretically minimal approximation error.

The blue lines indicate the actual errors $e_k$ incurred by 20 instantiations of the proposed method.

A is a discrete approximation of a certain compact integral operator normalized so that $\|A\| = 1$. Curiously, the nature of $A$ is in a strong sense irrelevant: the error distribution depends only on $\left\{ \sigma_j \right\}_{j=1}^{\min(m, n)}$. 

Example 3

The matrix $A$ being analyzed is a $9025 \times 9025$ matrix arising in a diffusion geometry approach to image processing.

To be precise, $A$ is a graph Laplacian on the manifold of $3 \times 3$ patches.
The pink lines illustrate the performance of the basic random sampling scheme. The errors for $q = 0$ are huge, and the estimated eigenvalues are much too small.
Outline

1. Randomized Numerical Linear Algebra (RandNLA)
2. Approximating Matrix Multiplication
3. Approximate SVD
4. Random Sampling for SVD
5. Single View Algorithm For Matrix Approximation
Low-rank reconstruction

Given $A \in \mathbb{R}^{m \times n}$ and a target rank $r$. Select $k$ and $\ell$. Given random matrices $\Omega \in \mathbb{R}^{n \times k}$ and $\Psi \in \mathbb{R}^{\ell \times m}$. Compute

$$Y = A \Omega, \quad W = \Psi A,$$

Then an approximation $\hat{A}$ is computed:

- Form an orthogonal-triangular factorization $Y = QR$ where $Q \in \mathbb{R}^{m \times k}$.
- Solve a least-squares problem to obtain $X = (\Psi Q)^\dagger W \in \mathbb{R}^{k \times n}$
- Construct the rank-$k$ approximation $\hat{A} = QX$

Suppose $k = 2r + 1$ and $\ell = 4r + 2$, then

$$\mathbb{E}\|A - \hat{A}\|_F \leq 2 \min_{\text{rank}(Z) \leq r} \|A - Z\|_F$$
Linear update of $A$

Suppose that $A$ is sent as a sequence of additive updates:

$$A = H_1 + H_2 + H_3 + \cdots$$

Then one compute

$$Y \leftarrow Y + H\Omega, \quad W = W + \Psi H$$

Suppose that $A$ is sent as a sequence of additive updates:

$$A = \theta A + \eta H$$

Then one compute

$$Y \leftarrow \theta Y + \eta H\Omega, \quad W = \theta W + \eta \Psi H$$
Suppose

\[ A \approx QQ^*A. \]

We want to form the rank-k approximation \( Q(Q^*A) \), but we cannot compute the factor \( Q^*A \) without revisiting the target matrix \( A \).

Note

\[ W = \Psi(QQ^*A) + \Psi(A - QQ^*A) \approx (\Psi Q)(Q^*A) \]

The construction of \( X \):

\[ X = (\Psi Q)^\dagger W \approx Q^*A \]

Hence

\[ \hat{A} = QX \approx QQ^*A \approx A \]
Projection onto a Convex Set.

Let $C$ be a closed and convex set. Define the projection:

$$\Pi_C(M) = \arg \min_X \|X - M\|_F^2, \text{ s.t. } X \in C$$

Suppose $A \in C$. Let $\hat{A}_{in}$ be an initial approximation of $A$,

$$\|A - \Pi_C(\hat{A}_{in})\|_F \leq \|A - \hat{A}_{in}\|_F$$

Conjugate Symmetric Approximation

$$H^n = \{X \in \mathbb{C}^{n \times n} | X = X^* \}$$

The projection

$$\Pi_{H^n}(M) = \frac{1}{2}(M + M^*)$$
Conjugate Symmetric Approximation.

Let \( A \in H^n \). Let \( \hat{A} = QX \).

\[
\Pi_{H^n}(\hat{A}) = \frac{1}{2} (QX + X^*Q^*) = \frac{1}{2} [Q, X^*] \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} [Q, X^*]^*
\]

Let \([Q, X^*] = U[T_1, T_2]\). Then

\[
S = \frac{1}{2} (T_1T_2^* + T_2T_1^*)
\]

Construct

\[
\hat{A}_{sym} = USU^*
\]
Let $A$ be positive semidefinite (PSD). Let $\hat{A} = QX$.

- Form eigenvalue decomposition

\[ S = VDV^* \]

- Compute

\[ \hat{A}_{sym} = (UV)D(UV)^* \]

- Construct

\[ \hat{A}_+ = \Pi_{H^+}(\hat{A}) = (UV)D_+(UV)^* \]