

Randomized Linear Algebra: Martinsson Ch 3,4

Nikola Janjušević, Paul Beckman

Sept. 16, 2020

Table of Contents

CH3: Matrix Factorizations and Low-Rank Appx.

3.2: Low-Rank Approximation

3.3: SVD

3.4: QR Factorization

3.5: Interpolative Decomposition

CH4: Randomized Methods for Low-Rank Approximation

4.2: Two-Stage Approach

4.3: The Range Finding Problem

4.4: SRFT Range Finder

4.5: Theoretical Performance Bounds

4.6: Power Iteration Range Finder

4.8: Randomized ID and SVD

3.1: Definitions and Notation

- ▶ Default vector-norm is Euclidean: $\|x\| = \|x\|_2 = \sqrt{\sum_i x_i^2}$
- ▶ Default matrix-norm is Spectral: $\|A\| = \sup_{\|x\|=1} \|Ax\|$
- ▶ **Def:**
 - ▶ Column-space of $A = \text{range}(A)$
 - ▶ Row-space of $A = \text{range}(A^t)$
 - ▶ Kernel of $A = \text{null}(A)$
 - ▶ $\text{rank}(A) = \dim(\text{range}(A))$
 - ▶ $\text{nullity}(A) = \dim(\text{null}(A))$
- ▶ Denote Conjugate-Transpose, A^*
- ▶ **Thm:** (Dimension Theorem) Let $A \in \mathbb{R}^{m \times n}$.
 $\text{rank}(A) + \text{nullity}(A) = \min(m, n)$

3.2: Low Rank Approximation

Definition: ϵ -rank k

Let $A \in \mathbb{R}^{m \times n}$, $\epsilon > 0$. We say A has ϵ -rank k if

- (a) $\exists B \in \mathbb{R}^{m \times n}$ s.t. $\mathbf{rank}(B) = k$ and $\|A - B\| \leq \epsilon$.
- (b) $\nexists B \in \mathbb{R}^{m \times n}$ s.t. $\mathbf{rank}(B) < k$ and $\|A - B\| \leq \epsilon$.

► Lazy definition: A has ϵ -rank k if $\inf\{\|A - B\| : \mathbf{rank}(B) = k\} \leq \epsilon$.

3.3: Singular Value Decomposition

For every $m \times n$ matrix A , there exists a unique¹ decomposition,

$$A = \begin{matrix} & U & D & V^* \\ m \times n & m \times p & p \times p & p \times n \end{matrix},$$

where $p = \min(m, n)$, U, V are orthonormal and their columns ($\{u_j, v_j\}_{j=1}^p$) called the left and right singular-vectors of A respectively, $D = \mathbf{diag}(\{\sigma_i\}_{i=1}^p)$, and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$ are called the singular values of A .

Best Approximation (Eckart-Young)

For A with SVD $A = UDV^*$, let $A_k = \sum_{j=1}^k \sigma_j u_j v_j^*$. Then,

$$\|A - A_k\| = \inf\{\|A - B\| : \mathbf{rank}(B) = k\} = \sigma_{k+1}$$

$$\|A - A_k\|_F = \inf\{\|A - B\|_F : \mathbf{rank}(B) = k\} = \left(\sum_{j=k+1}^p \sigma_j^2\right)^{1/2}$$

¹up to complex constants

3.3: Computing SVD

- ▶ SVD is equivalent to eigen-value problem which is equivalent to root-finding – it requires iterative solvers.
- ▶ In practice, algorithms are $O(mnp)$.
- ▶ Not easily parallelizable, although \exists tricks
- ▶ Example for $A \in \mathbb{R}^{m \times n}$, $m \gg n$:
 1. Compute QR Factorization, $A = QR$.
 2. Compute SVD of $R \in \mathbb{R}^{n \times n}$ factor, $R = \hat{U}DV^*$.
 3. $A = Q\hat{U}DV^* = UDV^*$

Off-load more expensive factorization to smaller matrix.

3.4: Column-Pivoted QR Decomposition (CPQR)

Let $A \in \mathbb{R}^{m \times n}$, $p = \min(m, n)$, then A admits a factorization,

$$\underset{m \times n}{A} \underset{n \times n}{P} = \underset{m \times p}{Q} \underset{p \times n}{R},$$

where P is a permutation matrix, Q orthonormal, R upper-triangular (UT).
i.e. in MATLAB, $AP = A(:, J)$, J , index vector.

- ▶ Same as normal QR except more numerically stable
- ▶ QR concept: Gram-Schmidt the columns of A .
- ▶ CPQR concept: Gram-Schmidt the columns of A choosing largest norm col.² next.

3.4: Example CPQR Algorithm

Algorithm 1: CPQR

Input: $A \in \mathbb{R}^{m \times n}$

Initialize: $Q_0 = [], R_0 = [], E_0 = A; p = \min(m, n)$

for $k = 1 : p$

$j_k = \operatorname{argmax}\{\|E_{k-1}(:, \ell)\| : \ell = 1, 2, \dots, n\}$ % select

$q = E_{k-1}(:, j_k)$

$q = q / \|q\|$ % normalize

$r = q^* E_{k-1}$ % compute coeffs

$Q_k = [Q_{k-1} \quad q]$ % store

$R_k = \begin{bmatrix} R_{k-1} \\ r \end{bmatrix}$

$E_k = E_{k-1} - qr$ % project

end

$P = \mathcal{I}(:, [j_1, j_2, \dots, j_p])$

$R = RP$

Outputs: Q, R, P

Note: for-loop forms $\hat{R} = RP^*$, permuted UT matrix R .

3.4: Low-rank Approximation via QR

At each step k of above algorithm,

$$A_{m \times n} = Q_k_{m \times k} R_k_{k \times n} + E_k_{m \times n}$$

- ▶ $Q_k R_k$ has rank k .
- ▶ Stop at k th step for rank- k appx.

-or- evaluate $\|E\|_k \leq \epsilon$ if seeking a certain precision.

Compute partial SVD via partial QR:

$$A = Q_k \underbrace{R_k}_{=\hat{U}DV^*} + E_k = \underbrace{Q_k \hat{U}}_{=U} DV^* + E_k = UD V^* + E_k$$

3.4: Aside on Blocking and Execution Speed

- ▶ Mat-Mat operations are better than looping Mat-Vec operations.
- ▶ For A square $n \times n$, QR, CPQR, and SVD are all $O(n^3)$, however with different constants.

Algorithm	QR	CPQR	SVD
Speed	Fast	Slow	Slowest
Ease of parallelization	Fairly Easy	Difficult	Difficult
Low-rank approximation?	No	Yes	Excellent
Partial factorization?	Yes but useless	Yes	Not easily

- ▶ CPQR and SVD are mostly Mat-Vec ops.
- ▶ Coming soon: random matrices to allow more Mat-Mat ops.

3.5: Interpolative Decomposition (ID)

Consider $A \in \mathbb{R}^{m \times n}$ with $\mathbf{rank}(A) = k < \min(m, n)$. Then A admits a factorization,

$$A = \begin{matrix} & C & Z \\ \begin{matrix} m \times n \\ m \times k \\ k \times n \end{matrix} & & \end{matrix},$$

with C subset of cols. of A , Z “well-conditioned”.

- ▶ C preserves sparsity and definiteness of A .
- ▶ Requires $k(m + n)$ words storage vs. mn or $p(m + n)$.
- ▶ Often “physics preserving”.

3.5: ID Computation from QR

Consider CPQR for A ,

$$\underset{m \times n}{A} \underset{n \times n}{P} = \underset{m \times p}{Q} \underset{p \times n}{S}.$$

with

$$Q = [Q_1 \quad Q_2] \quad \text{and} \quad S = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}$$

$Q_1 \in \mathbb{R}^{m \times k}$, $S_{11} \in \mathbb{R}^{k \times k}$. So,

$$AP = [Q_1 | Q_2] \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} = [Q_1 S_{11} | Q_1 S_{12} + Q_2 S_{22}] \quad (1)$$

$$= Q_1 [S_{11} | S_{12}] + Q_2 [0 | S_{22}] \quad (2)$$

$$= Q_1 S_{11} [\mathbf{I}_k | S_{11}^{-1} S_{12}] + Q_2 [0 | S_{22}] \quad (3)$$

$$A = \underbrace{Q_1 S_{11}}_{=C} \underbrace{[\mathbf{I}_k | S_{11}^{-1} S_{12}]}_{=Z} P^* + Q_2 [0 | S_{22}] P^* \quad (4)$$

$$A = CZ + Q_2 [0 | S_{22}] P^* \quad (5)$$

Note: for $\text{rank}(A) = k$, $S_{22} = 0$.

3.5: Column, Row, and Double-Sided ID

Given function for computing Column ID, $[J_s, Z] = \text{ID_col}(A, k)$ (where $A \approx A(:, J_s)Z$, rank k), we can easily form corresponding decompositions:

Row ID: $[I_s, X^t] = \text{ID_col}(A^t, k)$

$$A^t \approx A^t(:, I_s)X^t$$

$$A \approx XA(I_s, :)$$

Double-Sided ID: $[J_s, Z] = \text{ID_col}(A, k)$,
 $[I_s, X] = \text{ID_row}(A(:, J_s), k)$

$$A \approx A(:, J_s)Z \approx XA(I_s, J_s)Z.$$

- ▶ Clearly, only partial factorization needed in computing ID.
- ▶ Can augment to take tolerance ϵ rather than rank k .

3.6: Moore-Penrose Pseudoinverse

Generalized notion of matrix inverse for non-square (non-singular) matrices based on SVD. Let $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = k \leq \min(m, n)$. Let $A = UDV^*$ be the SVD of A . Then,

$$A = \sum_{j=1}^k \sigma_j u_j v_j^* = \begin{matrix} U_k & D_k & V_k^* \\ m \times k & k \times k & k \times n \end{matrix}$$

Then the **pseudoinverse** of A is the $n \times m$ matrix,

$$A^\dagger := V^k D_k^{-1} U_k^*,$$

and so,

$$AA^\dagger = U_k U_k^*, \quad A^\dagger A = V_k V_k^*$$

- ▶ A square nonsingular $\rightarrow A^\dagger = A^{-1}$.
- ▶ $x = A^\dagger b$ is LLS solution to $Ax = b$.

Table of Contents

CH3: Matrix Factorizations and Low-Rank Appx.

3.2: Low-Rank Approximation

3.3: SVD

3.4: QR Factorization

3.5: Interpolative Decomposition

CH4: Randomized Methods for Low-Rank Approximation

4.2: Two-Stage Approach

4.3: The Range Finding Problem

4.4: SRFT Range Finder

4.5: Theoretical Performance Bounds

4.6: Power Iteration Range Finder

4.8: Randomized ID and SVD

4.1: Introduction

Goal: Efficiently compute (good) rank- k approximations of $A \in \mathbb{R}^{m \times n}$.
Should take into account complexity *and* blocking³.


Starting point:

Draw $G \in \mathbb{R}^{n \times k}$, $G_{ij} \sim \mathcal{N}(0, 1)$

$$Y = AG \in \mathbb{R}^{m \times k}$$

$$A_k = YY^\dagger A \quad (\text{Orthogonal Proj.})$$

- ▶ Above is provably close to optimal for rank- $(k - 5)$ approximation
- ▶ Slightly more sophisticated approach with random matrix theory can bring us:
 - ▶ $O(mnk) \rightarrow O(mn \log k + k^2(m + n))$ appx. complexity
 - ▶ Less communication for distributed computing
 - ▶ *single-pass* factorization

³previously mentioned methods are $O(n^3)$ and not blocked 

4.2: Two-Stage Approach (for SVD)

Algorithm 2: Prototype Rank-k SVD

Input: $A \in \mathbb{R}^{m \times n}$, target rank k , oversampling parameter p

Output: Rank- $(k + p)$ approximate SVD of $A \approx UDV^*$

Stage A: Find approximate range.

Obtain $Q \in \mathbb{R}^{m \times (k+p)}$ orthonormal s.t. $A \approx QQ^*A$

Stage B: Factorize.

Form the matrix $B = Q^*A \in \mathbb{R}^{(k+p) \times n}$

Form the SVD of $B = \hat{U}D\hat{V}^*$

Form $U = Q\hat{U}$

- ▶ As $k \ll \min(m, n)$, Stage B is cheap.
- ▶ Stage B is exact (up to double precision).
- ▶ \rightarrow all errors result from Stage A: $\|A - UDV^*\| = \|A - QQ^*A\|$.

4.3: The Range Finding Problem

Algorithm 3: Range finding algorithm

Input: $A \in \mathbb{R}^{m \times n}$, target rank k , oversampling parameter p

Output: Q orthonormal spanning $\text{range}(A)$.

Form a Gaussian random matrix $G \in \mathbb{R}^{n \times (k+p)}$

Form the sample matrix $Y = AG \in \mathbb{R}^{m \times (k+p)}$

Orthonormalize the columns $Q = \text{orth}(Y)$

- ▶ If A has **exact rank- k** , above works with $p = 0$ and probability 1.
- ▶ Numerical rank deficiency causes problems \rightarrow fix with oversampling factor p .
- ▶ Martinsson (paraphrase): “ $p=10$ is nice”.
- ▶ Error analysis gives **expected error**, where singular-value decay is important.

4.4: The randomized SVD

MATLAB RSVD code:

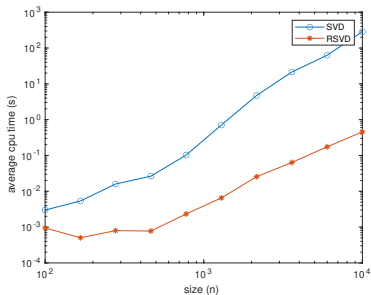
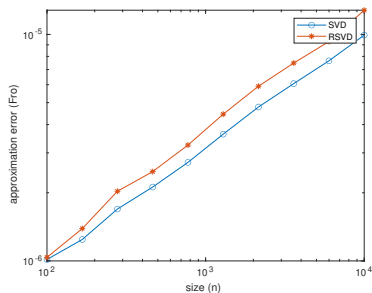
```
function [U,D,V] = rsvd(A,k,p)
    [m,n] = size(A);
    G = randn(n,k+p);
    Y = A*G;
    [Q,~] = qr(Y,0);
    B = Q'*A;
    [Uhat,D,V] = svd(B,'econ');
    U = Q*Uhat;
end
```

- ▶ This algorithm can be faster than deterministic methods depending on code optimization
- ▶ Has the same $\mathcal{O}(mnk)$ asymptotic complexity due to computing $Y = AG$ and $B = Q^*A$

Numerical Example

SVD vs. RSVD algorithm presented.

Rank-5 approximation from $A \in \mathbb{R}^{n \times n}$, with A numerically full rank but 5 dominant singular values.



4.4: Faster range finding using the FFT

To improve on $\mathcal{O}(mnk)$, we can use a non-Gaussian random matrix

$$\Omega = \begin{matrix} & D & F & S \\ \begin{matrix} n \times \ell \\ n \times n \\ n \times n \\ n \times \ell \end{matrix} & & & \end{matrix}$$

where we define the following matrices:

- ▶ D is a diagonal matrix with entries drawn uniformly at random from the unit circle in the complex plane
- ▶ F is the discrete Fourier transform

$$F_{pq} = n^{-1/2} e^{-2\pi i(p-1)(q-1)/n} \quad p, q = 1, 2, \dots, n$$

- ▶ S is a random subset of ℓ columns of the identity matrix

This is the *subsampled randomized Fourier transform* (SRFT)

$A\Omega$ can be evaluated in $\mathcal{O}(mn \log k)$ time complexity using the FFT

Ω is sufficiently random that $A\Omega$ accurately spans the range of A

4.5: Theoretical performance bounds

Since we have

$$A - \underbrace{U}_{=Q\hat{U}} DV^* = A - Q \underbrace{\hat{U}DV^*}_{=\text{svd}(B)} = A - Q \underbrace{B}_{=Q^*A} = A - QQ^*A$$

we can study $\|A - QQ^*A\| = \|A - UDV^*\|$ as the approximation error.

4.5: Theoretical performance bounds

Theorem 4.1: (Theorem 10.6 from [HMT11])

Let $A \in \mathbb{R}^{m \times n}$ have singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Take target rank k and oversampling parameter p with $p \geq 2$ and $k + p \leq \min(m, n)$.

Let $G \in \mathbb{R}^{m \times (k+p)}$ be a Gaussian random matrix, and define $Q = \text{orth}(AG) \in \mathbb{R}^{m \times (k+p)}$. Then we have

$$\begin{aligned}\mathbb{E}[\|A - QQ^*A\|_F] &\leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2} \\ \mathbb{E}[\|A - QQ^*A\|] &\leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2} \\ &\leq \left[1 + \sqrt{\frac{k}{p-1}} + \frac{e\sqrt{k+p}}{p} \sqrt{\min(m,n) - k}\right] \sigma_{k+1}\end{aligned}$$

4.6: Improved accuracy using power iteration

Recall the power iteration:

$$x_{q+1} \leftarrow Ax_q$$

x_q converges to the dominant eigenvector of A

Dominant left singular vectors of A are dominant eigenvectors of AA^*

Therefore by taking q power iterations, we can compute

$$Y = (AA^*)^q AG$$

in place of

$$Y = AG.$$

This works well because for $A = UDV^*$ we see that

$$(AA^*)^q A = UD^{2q+1}V^*$$

i.e. $(AA^*)^q A$ has the same left singular vectors as A but its singular values decay much more quickly

4.6: Improved accuracy using power iteration

Algorithm 4: Power iteration range finder

Input: $A \in \mathbb{R}^{m \times n}$, target rank k , oversampling parameter p , number of power iterations q

Output: Q orthonormal spanning $\text{range}(A)$.

Form Gaussian matrix $G \in \mathbb{R}^{n \times (k+p)}$

$Y = AG$

for $j = 1 : q$ **do**

$Z = A^*Y$

$Y = AZ$

Return $Q = \text{orth}(Y)$

4.6: Theoretical performance bounds

Theorem 4.2: (Corollary 10.10 from [HMT11])

Let $A \in \mathbb{R}^{m \times n}$ have singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Take target rank k , oversampling parameter p with $p \geq 2$ and $k + p \leq \min(m, n)$, and number of power iterations q .

Let $G \in \mathbb{R}^{m \times (k+p)}$ be a Gaussian random matrix, and define $Q = \text{orth} \left((AA^*)^q AG \right) \in \mathbb{R}^{m \times (k+p)}$. Then we have

$$\begin{aligned} \mathbb{E}[\|A - QQ^*A\|] &\leq \left[\left(1 + \sqrt{\frac{k}{p-1}} \right) \sigma_{k+1}^{2q+1} \right. \\ &\quad \left. + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^{2(2q+1)} \right)^{1/2} \right]^{1/(2q+1)} \\ &\leq \left[1 + \sqrt{\frac{k}{p-1}} + \frac{e\sqrt{k+p}}{p} \sqrt{\min(m,n) - k} \right]^{1/(2q+1)} \sigma_{k+1} \end{aligned}$$

4.7: Adaptive rank determination

So far we assume k to be fixed in advance

We can instead iteratively increase the rank until a norm error tolerance

$$\|A - QQ^*A\| \leq \varepsilon$$

is met.

See the text for details.

4.8: Randomized ID

- ▶ Our current RSVD algorithm relies on computing $\text{svd}(Q^*A)$
- ▶ We would like to avoid the $\mathcal{O}(mnk)$ cost of forming Q^*A

After computing $Y = AG$, if we compute the row ID

$$Y \approx XY[I_s, :]$$

there exists some $F \in \mathbb{R}^{k \times n}$ such that

$$A \approx YF$$

because the columns of Y form an approximate basis for the columns of A . Inserting the ID of Y , we obtain

$$A \approx XY[I_s, :]F.$$

Looking at rows I_s and using the fact that $X[I_s, :] = I$, we see that

$$A[I_s, :] \approx X[I_s, :]Y[I_s, :]F = Y[I_s, :]F$$

and thus

$$A \approx XA[I_s, :],$$

i.e. we automatically obtain the ID of A from the ID of Y .

4.8: From ID to SVD

Algorithm 5: RSVD

Input: $A \in \mathbb{R}^{m \times n}$, target rank k , oversampling parameter p

Output: Rank- $(k + p)$ approximate SVD of $A \approx UDV^*$

Form a SRFT random matrix $\Omega \in \mathbb{R}^{n \times (k+p)}$

Form the sample matrix $Y = A\Omega \in \mathbb{R}^{m \times (k+p)}$

Compute the ID of the sample matrix $Y = XY[I_s, :]$

Compute the QR decomposition of the interpolation matrix $X = QR$

Form the matrix $F = RA[I_s, :] \in \mathbb{R}^{(k+p) \times n}$

Compute the SVD of the matrix $F = \hat{U}DV^*$

Form $U = Q\hat{U}$

- ▶ We now have a fully $\mathcal{O}(mn \log k)$ algorithm for the SVD!
- ▶ Summary: Obtain $[I_s, X] = \text{rID_row}(Y, k)$ s.t. $A \approx XA(I_s, :)$. Then,

$$A \approx \underbrace{X}_{=QR} A(I_s, :) = Q \underbrace{RA(I_s, :)}_{=F} = Q \underbrace{F}_{=\hat{U}DV^*} = \underbrace{Q\hat{U}}_U DV^* = UDV^*$$

Summary

- ▶ Two-stage approach:
 - A. Construct orthonormal Q s.t. $A \approx QQ^*A$
 - B. Compute $\hat{U}DV^* = \text{svd}(Q^*A)$ and $U = Q\hat{U}$
- ▶ The range finding problem
 - ▶ Compute $Q = \text{orth}(Y)$ for sample matrix Y
 - ▶ $Y = AG$ for G Gaussian $\implies \mathcal{O}(mnk)$
 - ▶ $Y = A\Omega$ for Ω SRFT $\implies \mathcal{O}(mn \log k)$
 - ▶ $Y = (AA^*)^q AG$ or $(AA^*)^q A\Omega \implies$ improved accuracy
- ▶ Using the interpolative decomposition
 - ▶ Evaluating Q^*A is $\mathcal{O}(mnk)$
 - ▶ The ID of Y gives the ID of A for free
 - ▶ Can compute the SVD from this ID $\implies \mathcal{O}(mn \log k)$
- ▶ Advantages over deterministic methods
 - ▶ Reduce communication as well as flop counts
 - ▶ Can be adapted to use only a single pass over A

References



Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review **53** (2011), no. 2, 217–288.