Matrix factorization

Suraj Kumar

Inria & ENS Lyon

Email:*suraj.kumar@inria.fr*

CR12: September 2023

https://surakuma.github.io/courses/daamtc.html

Matrix factorizations

- Useful to solve systems of linear equations $Ax = b$

- Popular factorizations
  - LU factorization
  - QR factorization
  - Singular value decomposition

Important definitions

Vector norm for $x \in \mathbb{R}^n$

The Euclidean norm of x is represented as $||x||$ or $||x||_2$ and defined as $||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$

Matrix norm for $A \in \mathbb{R}^{n \times n}$

$$\text{Frobenius norm, } ||A||_F = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{n} {A_{ij}}^2} = \sqrt{trace(AA^T)}$$

Spectral norm, $||A||_2$ = largest singular value of A

Orthogonal matrix

An orthogonal matrix Q satisfies $Q^T Q = QQ^T = I$ (the identity matrix)

- Q's rows are orthogonal to each other and have unit norm
- Q's columns are orthogonal to each other and have unit norm

Table of Contents

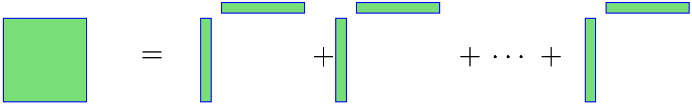
1. Singular value decomposition
2. LU factorization
3. QR factorization

Singular Value Decomposition (SVD)

- It decomposes a matrix $A \in \mathbb{R}^{m \times n}$ to the form $U \Sigma V^T$
  - U is an $m \times m$ orthogonal matrix
  - V is an $n \times n$ orthogonal matrix
  - Σ is an $m \times n$ rectangular diagonal matrix
- The diagonal entries $\sigma_i = \Sigma_{ii}$ of Σ are called singular values
  - $\sigma_i \geq 0$ and $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_{\min(m,n)}$
- Columns of U and V are known as left and right singular vectors respectively
- If u_i, v_i are the *ith* vector of U and V, then $A = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^T$
- The largest r such that $\sigma_r \neq 0$ is called the rank of the matrix

SVD and rank of a matrix

- SVD represents a matrix as the sum of r rank one matrices



- $||A||_F^2 = \sum_{i=1}^{\min(m,n)} \sigma_i^2 = \sum_{i=1}^{r} \sigma_i^2$
- If $r' \leq r$ and $\tilde{A} = \sum_{i=1}^{r'} \sigma_i u_i v_i^T$, then

$$||A - \tilde{A}||_F^2 = \sum_{i=r'+1}^{\min(m,n)} \sigma_i^2 = \sum_{i=r'+1}^{r} \sigma_i^2$$

- Useful for compression, dimension reduction and low-rank approximation
- Expensive to compute and hard to parallelize

Table of Contents

1. Singular value decomposition
2. LU factorization
3. QR factorization

Algebra of LU factorization with an example

Given the matrix $A = \begin{pmatrix} 2 & 6 & 5 \\ 4 & 15 & 11 \\ 6 & 30 & 23 \end{pmatrix}$

- Let $L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}$, $L_1 A = \begin{pmatrix} 2 & 6 & 5 \\ 0 & 3 & 1 \\ 0 & 12 & 8 \end{pmatrix}$
- Let $L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{pmatrix}$, $L_2 L_1 A = \begin{pmatrix} 2 & 6 & 5 \\ 0 & 3 & 1 \\ 0 & 0 & 4 \end{pmatrix}$
- Let $U = \begin{pmatrix} 2 & 6 & 5 \\ 0 & 3 & 1 \\ 0 & 0 & 4 \end{pmatrix}$, $L_2 L_1 A = U$

Algebra of LU factorization

$$L_2 L_1 A = U \implies A = (L_2 L_1)^{-1} U = L_1^{-1} L_2^{-1} U$$

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix},\ L_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix},\ L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{pmatrix},\ L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{pmatrix}$$

$$L_1^{-1} L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} 2 & 6 & 5 \\ 4 & 15 & 11 \\ 6 & 30 & 23 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix} \begin{pmatrix} 2 & 6 & 5 \\ 0 & 3 & 1 \\ 0 & 0 & 4 \end{pmatrix} = LU, \text{ where } L = L_1^{-1} L_2^{-1}$$

The need of pivoting (or row exchanges): $PA = LU$

- To avoid division by 0 or small diagonal elements (for stability)
- $A = \begin{pmatrix} 0 & 2 & 4 \\ 3 & 1 & 2 \\ 6 & 8 & 7 \end{pmatrix}$ has an LU factorization if we permute the rows of the matrix A

$$PA = \begin{pmatrix} 3 & 1 & 2 \\ 0 & 2 & 4 \\ 6 & 8 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 3 & 1 \end{pmatrix} \begin{pmatrix} 3 & 1 & 2 \\ 0 & 2 & 4 \\ 0 & 0 & -9 \end{pmatrix}$$

$$\text{Here } P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Communication lower bounds

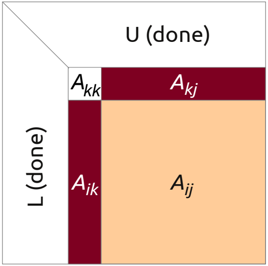
- Matrix multiplication lower bounds apply to LU factorization using reduction [Ballard et. al., 09]

$$\begin{pmatrix} I & & -B \\ A & I & \\ & & I \end{pmatrix} = \begin{pmatrix} I & & \\ A & I & \\ & & I \end{pmatrix} \begin{pmatrix} I & & -B \\ & I & AB \\ & & I \end{pmatrix}$$

Lower bounds

- Sequential lower bound on bandwidth $= \Omega(\frac{n^3}{\sqrt{M}})$
- Memory-dependent parallel lower bound on bandwidth $= \Omega(\frac{n^3}{P\sqrt{M}})$
- Memory-independent parallel lower bound on bandwidth $= \Omega\left(\frac{n^3}{P^{\frac{2}{3}}}\right)$

LU factorization



LU factorization (Gaussian elimination):

- Convert a matrix A into product $L \times U$
- L is lower triangular with diagonal 1
- U is upper triangular
- L and U stored in place with A

LU Algorithm

For $k = 1 \ldots n-1$:

- For $i = k+1 \ldots n$,
  $A_{i,k} \leftarrow A_{i,k}/A_{k,k}$ (column/panel preparation)
- For $i = k+1 \ldots n$,
  For $j = k+1 \ldots n$,
  $A_{i,j} \leftarrow A_{i,j} - A_{i,k}A_{k,j}$ (update)

Block LU factorization

Partition of a $n \times n$ matrix A

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

Here A_{11} is of size $b \times b$, A_{21} is of size $(n-b) \times b$, A_{12} is of size $b \times (n-b)$ and A_{22} is of size $(n-b) \times (n-b)$.

Structure of LU factorization algorithm

- The first iteration computes the factorization:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & I_{n-b} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & A' \end{pmatrix}$$

- The algorithm continues recursively on the trailing matrix A'.

Block LU factorization

1. Compute the LU factorization of the first block column

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} U_{11}$$

2. Solve the triangular system

$$L_{11} U_{12} = A_{12}$$

3. Update the trailing matrix

$$A' = A_{22} - L_{21} U_{12}$$

4. Compute recursively the block LU factorization of A'

Table of Contents

1. Singular value decomposition
2. LU factorization
3. QR factorization

Terminology related to QR factorization

An **orthogonal** matrix Q satisfies $Q^T Q = QQ^T = I$ (the identity matrix)

- Q must be square
- Q's rows are orthogonal to each other and have unit norm
- Q's columns are orthogonal to each other and have unit norm

Terminology related to QR factorization

An **orthogonal** matrix Q satisfies $Q^TQ = QQ^T = I$ (the identity matrix)

- Q must be square
- Q's rows are orthogonal to each other and have unit norm
- Q's columns are orthogonal to each other and have unit norm

A matrix U has **orthonormal columns** if $U^TU = I$ (the identity matrix)

- U's columns are orthogonal to each other and have unit norm
- U can have more rows than columns, in which case $UU^T \neq I$

Terminology related to QR factorization

An **orthogonal** matrix Q satisfies $Q^T Q = QQ^T = I$ (the identity matrix)

- Q must be square
- Q's rows are orthogonal to each other and have unit norm
- Q's columns are orthogonal to each other and have unit norm

A matrix U has **orthonormal columns** if $U^T U = I$ (the identity matrix)

- U's columns are orthogonal to each other and have unit norm
- U can have more rows than columns, in which case $UU^T \neq I$

Given a matrix A, we can **orthogonalize** its columns by finding a matrix Q such that

- Q's columns span the same space as A's columns
- Q has orthonormal columns
- there exists a nonsingular matrix Z such that $A = QZ$

QR factorization

The **QR factorization** is a fundamental matrix factorization:

$$A = QR = \begin{bmatrix} \hat{Q} & \tilde{Q} \end{bmatrix} \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} = \hat{Q}\hat{R}$$

- if A is $m \times n$, $m \geq n$, then Q is $m \times m$, R is $m \times n$, $\hat{Q}$ is $m \times n$, and $\hat{R}$ is $n \times n$
- Q is orthogonal, $\hat{Q}$ has orthonormal columns, and R is upper triangular
- $\hat{Q}$ is an orthogonalization of A

Classical algorithms for QR factorization

1. Gram-Schmidt process
   - intuitive: each vector is orthogonalized against previous ones by subtracting out components of the vector in previous directions
   - has numerical problems (vectors aren't always numerically orthonormal)
   - two variants "classical" and "modified" are mathematically identical

2. Householder QR
   - uses orthogonal matrices to transform input to triangular form
   - numerically stable

Gram-Schmidt

Classical Gram-Schmidt (CGS) process

Require: $A = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$

for $i = 1$ to n **do**

$v_i = x_i$

for $j = 1$ to $i-1$ **do**

$r_{ji} = q_j^T x_i$ ▷ compute size of projection of ith col of A onto q_j

$v_i = v_i - r_{ji} q_j$ ▷ remove this component from vector v_i

end for

$r_{ii} = \|v_i\|_2$

$q_i = v_i / r_{ii}$ ▷ normalize vector

end for

Ensure: $Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}$ has orthonormal columns

Ensure: R is upper triangular and $A = QR$

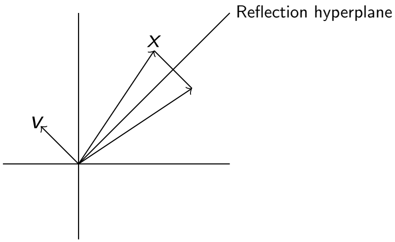
Gram-Schmidt

Modified Gram-Schmidt (MGS) process

Require: $A = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$
  for $i = 1$ to n **do**
    $v_i = x_i$
    for $j = 1$ to $i - 1$ **do**
      $r_{ji} = q_j^T v_i$ ▷ compute size of projection of current vector onto q_j
      $v_i = v_i - r_{ji} q_j$ ▷ remove this component from vector v_i
    end for
    $r_{ii} = \|v_i\|_2$
    $q_i = v_i / r_{ii}$ ▷ normalize vector
  end for
Ensure: $Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}$ has orthonormal columns
Ensure: R is upper triangular and $A = QR$

Householder transformation

- v is a unit vector
- The reflection hyperplane can be defined by its normal vector v
- $(I - 2vv^T)x$ is the reflection of point x with the hyperplane



- $P = I - 2vv^T$ matrix is known as the Householder matrix
- P is symmetric and orthogonal, $P^2 = I$

Main idea of Householder QR factorization

Look for a Householder matrix that annihilates the elements of a vector x, except first one:

$$Px = y, ||x||_2 = ||y||_2, y = \sigma e_1, \sigma = \pm||x||_2$$

The choice of sign is made to avoid cancellation or small numerical values while computing $v_1 = x_1 - \sigma$. Here v_1, x_1 are the first elements of vectors v, x respectively.

$$v = x - y = x - \sigma e_1$$

$$\sigma = -sign(x1)||x||_2, v = x - \sigma e_1$$

$$u = \frac{v}{||v||_2}, P = I - 2uu^T$$

Householder QR algorithm

Given vector x, a **Householder transformation** $I - 2uu^T$ maps x to σe_1

- u is called the **Householder vector**

Require: $A = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$
  for $i = 1$ to n **do**
    Compute Householder vector u_i from x_i
    $A = (I - 2u_i u_i^T)A$ ▷ apply Householder transformation
  end for
  $R = A$
Ensure: $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix}$ is lower triangular
Ensure: R is upper triangular and $A = (I - 2u_1 u_1^T) \cdots (I - 2u_n u_n^T)R$

Householder QR computational complexity

Let $A \in \mathbb{R}^{m\times n}$, we count the number of operation to update A ($A = (I - 2u_iu_i^T)A = A - 2u_iu_i^TA$) in each iteration i.

Operations per iteration

- Dot product $w = u_i^TA(i:m, i:n) : 2(m-i)(n-i)$
- Outer product $u_iw : (m-i)(n-i)$
- Subtraction $A(i:m, i:n) = A(i:m, i:n) - 2u_iw : (m-i)(n-i)$

The number of operations to multiply 2 with w is $(n-i)$, however it is a lower order term. Hence we do not consider it explicitly.

Operations in Householder QR factorization

$$\sum_{i=1}^{n} = 4(m-i)(n-i) = 4\sum_{i=1}^{n} = 4(mn - (m+n)i + i^2)$$

$$\approx 4mn^2 - 4(m+n)\frac{n^2}{2} + 4\frac{n^3}{3} = 2mn^2 - 2\frac{n^3}{3}$$

QR factorization

- Q can be stored in compact representation
- Structure of block QR algorithm is similar to the block LU algorithm
- Matrix communication lower bounds are also valid for the Householder/CGS/MGS QR factorization

QR factorization algorithms

Algorithm	**# flops**	**# words**	**stability**
CGS	$2mn^2 + O(n^3)$	$O(mn^2)$	Bad
MGS		$O(mn^2)$	Okay
HouseholderQR		$O(mn^2)$	Good

Table of Contents

1. Singular value decomposition
2. LU factorization
3. QR factorization
   - Tall Skinny QR (TSQR) factorization

TSQR: QR factorization of a tall skinny matrix

QR factorization of a $m \times n$ matrix with $m >> n$

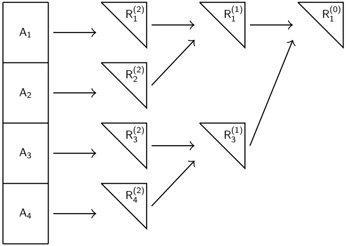
The goal and process of Householder QR:

- annihilate entries below diagonal to obtain upper triangular form
- work column-by-column, left-to-right

Tall-Skinny QR idea (Demmel, Grigori, Hoemmen, Langou '12):

- change the order of annihilation to minimize communication
- work row-by-row, top to bottom

Algebra of TSQR



$$A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} = \begin{pmatrix} Q_1^{(2)} R_1^{(2)} \\ Q_2^{(2)} R_2^{(2)} \\ Q_3^{(2)} R_3^{(2)} \\ Q_4^{(2)} R_4^{(2)} \end{pmatrix} = \begin{pmatrix} Q_1^{(2)} & & & \\ & Q_2^{(2)} & & \\ & & Q_3^{(2)} & \\ & & & Q_4^{(2)} \end{pmatrix} \begin{pmatrix} R_1^{(2)} \\ R_2^{(2)} \\ R_3^{(2)} \\ R_4^{(2)} \end{pmatrix}$$

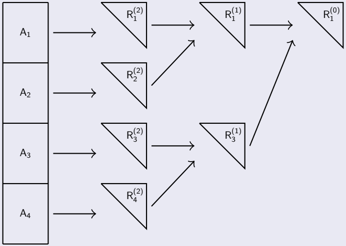
$$\begin{pmatrix} R_1^{(2)} \\ R_2^{(2)} \\ R_3^{(2)} \\ R_4^{(2)} \end{pmatrix} = \begin{pmatrix} Q_1^{(1)} R_1^{(1)} \\ Q_2^{(1)} R_2^{(1)} \end{pmatrix} = \begin{pmatrix} Q_1^{(1)} & \\ & Q_2^{(1)} \end{pmatrix} \begin{pmatrix} R_1^{(1)} \\ R_2^{(1)} \end{pmatrix}, \quad \begin{pmatrix} R_1^{(1)} \\ R_2^{(1)} \end{pmatrix} = Q_1^{(0)} R_1^{(0)}$$

Q is represented implicitly as a product.

Flexibility of TSQR

Parallel TSQR

- Assuming block row layout on P processors
- Communication cost is that of binomial-tree reduction: $\beta \cdot O(n^2 \log P) + \alpha \cdot O(\log P)$



Sequential TSQR

- Assuming cache size is $\Omega(n^2)$
- It streams through matrix once achieving $O(mn)$ amount of data transfers