Training and Workshop on Python Programming and Numerical Methods

An Introduction to Numerical Linear Algebra and Differential Equations

RHUDAINA MOHAMMAD

Numerical Analysis & Scientific Computing Group
Institute of Mathematics, UP Diliman
rzmohammad@up.edu.ph

– BREAK –

10:00 – 10:30

MATRIX FACTORIZATION

Matrix Factorization

Matrix factorization (decomposition) writes a matrix A as product of matrices

$$A = BCD\cdots$$

where matrices $B, C, D, \ldots$ have some special structure.

- Diagonal matrices
- Triangular matrices
- Orthogonal matrices
- Permutation matrices

Matrix Factorization

Diagonal matrix $D = [d_{ij}]$ is a matrix in which all off-diagonal entries are zero, i.e., for $i \neq j$, $d_{ij} = 0$.

$$\begin{bmatrix} * & & & \\ & * & & \\ & & \ddots & \\ & & & * \end{bmatrix} \quad \begin{bmatrix} * & & & \\ & * & & \\ & & \ddots & \\ & & & * \\ & & & \\ & & & \end{bmatrix} \quad \begin{bmatrix} * & & & & \\ & * & & & \\ & & \ddots & & \\ & & & * & \end{bmatrix}$$

Matrix Factorization

Lower triangular matrix $A = [a_{ij}]$: a square matrix in which all entries above the main diagonal are zero, i.e., for $i < j$, $a_{ij} = 0$.

$$\begin{bmatrix} * & & & \\ * & * & & \\ \vdots & \vdots & \ddots & \\ * & * & \cdots & * \end{bmatrix}$$

Upper triangular matrix $A = [a_{ij}]$ is a square matrix in which all entries below the main diagonal are zero, i.e., for $i > j$, $a_{ij} = 0$.

$$\begin{bmatrix} * & * & \cdots & * \\ & * & \cdots & * \\ & & \ddots & \vdots \\ & & & * \end{bmatrix}$$

Matrix Factorization

First step of Gauss elimination (without row swapping):

$$-m_{j1}R_1 + R_j \to R_j, \qquad m_{j1} = \frac{a_{j1}}{a_{11}}$$

for $j = 2, 3, \ldots, n$.

- This is equivalent to: $L^{(1)}Ax = L^{(1)}y =: y^{(2)}$ where

$$L^{(1)} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -m_{21} & 1 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -m_{n1} & 0 & \cdots & 0 & 1 \end{bmatrix},$$

  which is a lower triangular matrix.
- Denote $A^{(2)} := L^{(1)}A$.

Matrix Factorization

At kth step with $A^{(k)}x = y^{(k)}$, we obtain

$$A^{(k+1)}x \;=\; L^{(k)}A^{(k)}x \;=\; L^{(k)}L^{(k-1)}\cdots L^{(1)}Ax \;=:\; y^{(k+1)}$$

where

$$L^{(k)} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & \ddots & \ddots & & & \vdots \\ \vdots & & \vdots & -m_{k+1,k} & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -m_{nk} & 0 & \cdots & 0 & 1 \end{bmatrix},$$

which is lower triangular.

Matrix Factorization

- The process ends with:

$$A^{(n)}x = y^{(n)}$$

where

$$A^{(n)} = L^{(n-1)}L^{(n-2)}\cdots L^{(1)}A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n}^{(n-1)} \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{bmatrix} =: U,$$

an upper triangular matrix.

Matrix Factorization

$$\underbrace{L^{(n-1)}L^{(n-2)}\cdots L^{(1)}}_{\widetilde{L},\ \text{lower triangular}}A = U$$

$$\widetilde{L}A = U$$

$$A = \underbrace{\widetilde{L}^{-1}}_{L}U$$

$$A = LU$$

LU Factorization

If Gauss elimination can be performed on $Ax = y$ without row swaps, then A can be factored into a product of lower tringular matrix L and upper triangular matrix U, that is,

$$A = LU$$

where

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ m_{n1} & \cdots & m_{n,n-1} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n}^{(n-1)} \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{bmatrix}$$

LU Factorization

- However, not every nonsingular (invertible) matrix allows an LU factorization.

  Example.
  $$A = \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix}$$
  has no LU factorization.

LU Factorization

- For each nonsingular (invertible) $n \times n$ matrix A, there exists a permutation matrix P such that

$$PA = LU$$

  - Note that $P = (e_{p(1)}, \dots, e_{p(n)})$ where

    $e_1, \dots, e_n$ are the columns of the identity matrix I_n

    $p(1), \dots, p(n)$ is a permutation of $1, \dots, n$

  Example.

$$PA = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix},$$

  which can now be factored into LU.

LU Factorization

To solve linear system $Ax = y$:

1. In Python, decompose $A = PLU$. This means that

$$P^T A = LU \quad \Rightarrow \quad P^T \underbrace{Ax}_{y} = L \underbrace{Ux}_{z}$$

```
from scipy.linalg import lu

P, L, U = lu(A)
print(P);    # prints permutation matrix P
print(L);    # prints lower triangular matrix L
print(U);    # prints upper triangular matrix U
```

2. solve $Lz = P^T y$ via forward substitution
3. solve $Ux = z$ via backward substitution

Orthogonality

- We say $u, v \in \mathbb{R}^n$ are orthogonal vectors if their inner product

$$(u, v) = 0$$

- Orthogonal basis for a subspace: every pair of basis vectors are orthogonal.
- Orthonormal basis: an orthogonal basis of unit vectors.
  - The norm of vector $z \in \mathbb{R}^n$ is given by

$$\|z\|^2 = (z, z) = z_1^2 + z_2^2 + \cdots + z_n^2.$$

Orthogonality

- Tall thin matrix Q with orthonomal columns:

$$Q^T Q = I$$

- Orthogonal matrices are square matrices with orthonormal columns:

$$Q^T = Q^{-1}$$

  - Columns of an $n \times n$ orthogonal matrix form an orthonormal basis for $\mathbb{R}^n$.
  - Rows of an $n \times n$ orthogonal matrix also form an orthonormal basis for $\mathbb{R}^n$.
  - The name ”orthogonal matrix” should really be ”orthonormal matrix”

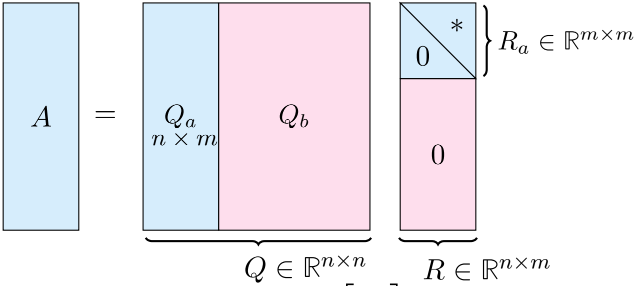
QR Factorization

- **Gram-Schmidt Process** produces orthonormal vectors from linearly independent vectors of A.

$$
\begin{aligned}
A \quad &:= \begin{bmatrix} \vdots & \vdots & & \vdots \\ a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & & \vdots \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \vdots & \vdots & & \vdots \\ q_1 & q_2 & \cdots & q_n \\ \vdots & \vdots & & \vdots \end{bmatrix}}_{Q,\ \text{orthogonal}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}}_{R,\ \text{right (upper) triangular}} , \qquad r_{ij} = q_i^T a_j \\
&= QR
\end{aligned}
$$

QR Factorization

Consider a matrix $A \in \mathbb{R}^{n\times m}$ with rank $m < n$.



$$
\begin{aligned}
&= QR = \begin{bmatrix} Q_a & Q_b \end{bmatrix} \begin{bmatrix} R_a \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \vdots & & \vdots & \vdots & & \vdots \\ q_1 & \cdots & q_m & q_{m+1} & \cdots & q_n \\ \vdots & & \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} R_a \\ 0 \end{bmatrix} \\
&= Q_a R_a
\end{aligned}
$$

QR Factorization

- Using `numpy`:

```
from numpy.linalg import qr

Q,R = qr(A)                #default: reduced  factorization
Q,R = qr(A, mode="complete") #complete factorization
```

- Using `scipy`:

```
from scipy.linalg import qr

Q,R = qr(A)               #default: complete decomposition
Q,R = qr(A, mode="economic") #reduced decomposition
```

Linear Least Squares Problem

- *Algorithm.* Linear Least Squares via QR Factorization

  To solve

$$\min_x \|y - Ax\|^2 ,$$

  we perform the following:

  1. Factor

$$A = QR$$

     where $Q = [Q_a, Q_b] \in \mathbb{R}^{n \times n}$ with $Q_a \in \mathbb{R}^{n \times m}$
  2. Compute $z_a = Q_a^T y$.
  3. Solve $R_a x = z_a$ by back substitution.

Linear Least Squares Problem

- Suppose $A = QR$. Define

$$z = Q^T y = \begin{bmatrix} z_a \\ z_b \end{bmatrix}$$

where $z_a \in \mathbb{R}^m$, $z_b \in \mathbb{R}^{n-m}$. Then,

$$\begin{aligned} \|y - Ax\|^2 &= \left\|Q^T(y - Ax)\right\|^2 \\ &= \|z - Rx\|^2, \qquad\qquad \text{since } R = Q^T A \\ &= \|z_a - R_a x\|^2 + \|z_b\|^2 \end{aligned}$$

- To minimize this norm, we make the first term zero by taking x to be the solution of the linear system

$$R_a x = z_a.$$

The norm of the residual can then be computed as $\|z_b\| = \left\|Q_b^T y\right\|$.

Linear Least Squares Problem

Example. For a number of Portland cements of varied composition, the heat evolved during setting and hardening has been determined after 180 days.

The cement compunds are:

C_1. tricalcium aluminate ($3CaO \cdot Al_2O_3$)

C_2. tricalcium silicate ($3CaO \cdot SiO_2$)

C_3. tetracalcium aluminoferrite ($4CaO \cdot Al_2O_3 \cdot Fe_2O_3$)

C_4. β-dicalcium silicate ($2CaO \cdot SiO_2$)

Determine the contribution of each percent of each compound to the total heat evolution on the assumption that there exists a linear relationship between the compound composition of a cement and its heat evolution.

Cement	C_1	C_2	C_3	C_4	heat
2122	7	26	6	60	78.5
2123	1	29	15	52	74.3
2092	11	56	8	20	104.3
2088	11	31	8	47	87.6
2096	7	52	6	33	95.9
2085	11	55	9	22	109.2
2094	3	71	17	6	102.7
2124	1	31	22	44	72.5
2089	2	54	18	22	93.1
2090	21	47	4	26	115.9
2125	1	40	23	34	83.8
2095	11	66	9	12	113.3
2091	10	68	8	12	109.4

Eigendecomposition

Consider matrix A with a full set of n independent eigenvectors $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$.

- Denote eigenvector matrix

$$X = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix},$$

  which is nonsingular (invertible). Hence,

$$AX = \begin{bmatrix} Ax_1 & \cdots & Ax_n \end{bmatrix} = \begin{bmatrix} \lambda_1 x_1 & \cdots & \lambda_n x_n \end{bmatrix} = X \underbrace{\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}}_{\Lambda}$$

- This gives the factorization: $A = X \Lambda X^{-1}$.

Symmetric Positive Definite Matrices

- For a symmetric matrix $S = S^T$, we have the following properties:
  - All n eigenvalues are real numbers.
  - The n eigenvectors $q_1, \ldots, q_n$ can be chosen to be orthonormal, i.e. inner product

$$(q_i, q_j) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

- Hence, the eigenvector matrix for S is orthogonal, i.e.,

$$Q^T Q = I$$

with orthonormal columns

Symmetric Positive Definite Matrices

- Spectral Theorem. Every real symmetric matrix S has the factorization

$$S = Q\Lambda Q^{-1}$$

- A positive definite matrix has all positive eigenvalues.
- A positive semidefinite matrix has all nonnegative eigenvalues.

Singular Values and Singular Vectors

Consider matrix $A \in \mathbb{R}^{n\times m}$.

- Note that $A^TA \in \mathbb{R}^{m\times m}$ is a symmetric positive semidefinite matrix
- The singular values of A, denoted by $\sigma_1, \ldots, \sigma_n$ are the nonnegative square roots of the eigenvalues of A^TA.
  - If rank of A is r, then A has exactly r positive singular values, i.e., in descending order

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$$

  - It can be shown that there exists two sets of orthonormal vectors, called singular vectors $u_1, \ldots, u_n \in \mathbb{R}^n$ and $v_1, \ldots, v_m \in \mathbb{R}^m$ such that

$$Av_1 = \sigma_1 u_1 \quad \cdots \quad Av_r = \sigma_i u_r \qquad Av_{r+1} = 0 \quad \cdots \quad Av_m = 0$$

Singular Value Decomposition

- Denote orthogonal matrices

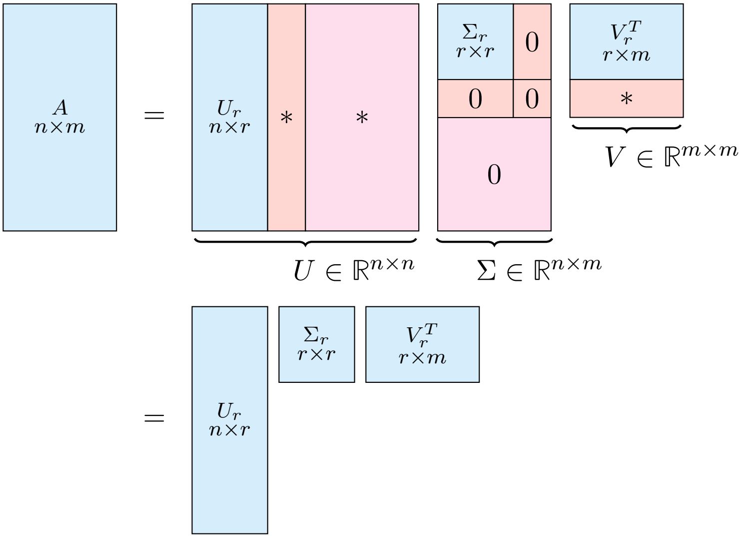
$$U = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \qquad V = \begin{bmatrix} v_1 & \cdots & v_m \end{bmatrix}$$

Then, we can write

$$AV = A \begin{bmatrix} v_1 & \cdots & v_m \end{bmatrix} = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \underbrace{\left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_r & \\ \hline & 0 & & 0 \end{array}\right]}_{\Sigma} = U\Sigma$$

- This gives the factorization: $A = U\Sigma V^T$, called Singular Value Decomposition

Singular Value Decomposition



Singular Value Decomposition

- Full SVD:

```python
from numpy.linalg import svd
U, S, Vh = svd(A);
```

- Reduced SVD:

```python
from numpy.linalg import svd
U, S, Vh = svd(A, full_matrices=False);  # reduced SVD
```

- If $A = U\Sigma V^T = U_r \Sigma_r V_r^T$, then a solution to linear system $Ax = y$ is given by

$$x_* = \underbrace{V_r \Sigma_r^{-1} U_r^T}_{A^\dagger} y, \qquad \text{where } \Sigma_r^{1} = \begin{bmatrix} \lambda_1^{-1} & & \\ & \ddots & \\ & & \lambda_r^{-1} \end{bmatrix},$$

$A^\dagger$ is called pseudo-inverse or Moore-Penrose inverse of A

Hands-on Activity

1. Generate a random $n \times n$ matrix A and a random n-vector x_{true} for $n = 2^3, \ldots, 2^{10}$. For each n, define $y = Ax_{\text{true}}$ and compare runtimes in solving for x in linear system

$$Ax = y$$

using LU, QR, eigendecomposition, SVD, and built-in linear system solver ‘np.linalg.solve‘. Which method is the fastest?

– LUNCH –

12:00 – 13:00