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

29-31 January 2025 · Visayas State University, Baybay City, Leyte

– LUNCH –
12:00 – 13:00

Hands-on Activity

2. Generate a set of $n = 100$ data points (x_i, y_i) for $i = 1, \ldots, n$ with

$$\frac{1}{2} = x_1 < x_2 < \cdots < x_{n-1} < x_n = 1$$

$$y_i = ae^{b(x_i+1)} + \delta_i, \quad i = 1, 2, \ldots, n$$

where $a = b = 2$. Assume that δ_i's are independent and normally distributed with mean 0 and variance 1.
Although the data points exhibit a nonlinear relationship, we can transform the problem into a linear form by taking the natural logarithm as follows.

$$\ln y = \ln a + b(x + 1)$$

Hands-on Activity

This leads to the following linear system

$$\begin{bmatrix} 1 & x_1 + 1 \\ 1 & x_2 + 1 \\ \vdots & \\ 1 & x_n + 1 \end{bmatrix} \begin{bmatrix} \ln a \\ b \end{bmatrix} = \begin{bmatrix} \ln y_1 \\ \ln y_2 \\ \vdots \\ \ln y_n \end{bmatrix}$$

Determine the best exponential curve that fits the given data points by solving the corresponding linear least squares problem via singular value decomposition. In one figure, plot the true and fitted nonlinear curves with the given data points. Discuss your results.

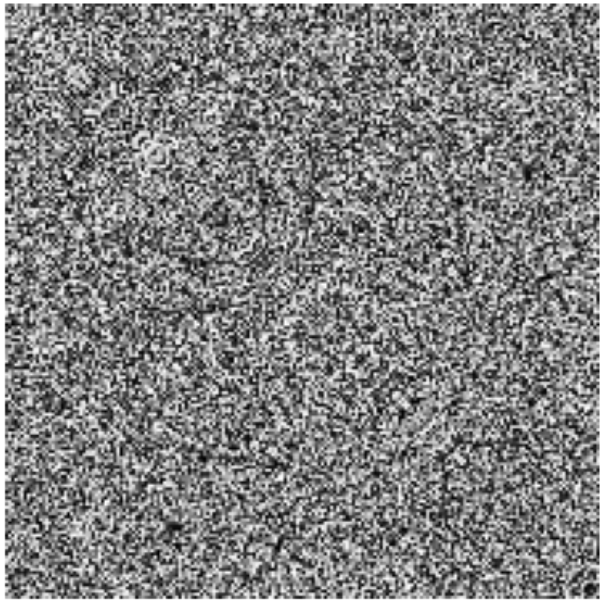
MORE ON SVD

Dimensionality Reduction

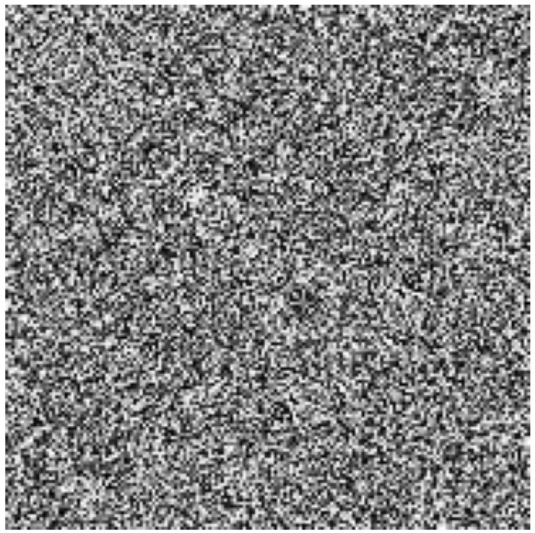
High dimensionality is a common challenge in processing data from complex systems.

Dimensionality Reduction

$A \in \mathbb{R}^{192 \times 168}$



$AA^T \in \mathbb{R}^{192 \times 192}$



$A^TA \in \mathbb{R}^{168 \times 168}$

- Both correlation matrices AA^T and A^TA are symmetric.
  - AA^T is formed by taking inner product of rows of A
  - A^TA is formed by taking inner product of columns of A

Dimensionality Reduction

- Note that

$$A = U\Sigma V^T = U \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix} V^T$$

Then,

$$\begin{aligned} AA^T &= U \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix} \underbrace{V^T V}_{I} \begin{bmatrix} \hat{\Sigma} & 0 \end{bmatrix} U^T = U \begin{bmatrix} \hat{\Sigma} & 0 \\ 0 & 0 \end{bmatrix} U^T \\ A^T A &= V \begin{bmatrix} \hat{\Sigma} & 0 \end{bmatrix} \underbrace{U^T U}_{I} \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix} V^T = V \hat{\Sigma}^2 V^T \end{aligned}$$

- To get the singular values, it is better to compute the eigendecomposition of $A^T A$, which is smaller and more manageable than AA^T.
- This gives V and $\hat{\Sigma}$. From $\hat{\Sigma}$, we only keep p nonzero singular values in Σ_p; and their corresponding columns V_p of V. From these matrices, we can construct $U_p = AV_p\Sigma_p^{-1}$

Dimensionality Reduction

- Since all nonzeros entries of Σ are in the diagonal, then we can write

$$A = U\Sigma V^T = \sum_{i=1}^{m} \sigma_i u_i v_i^T = \underbrace{\sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_m u_m v_m^T}_{\text{sum of rank-1 matrices}}$$

  where singular values σ_i is the ith diagonal entry of Σ such that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$$

  Here, u_i and v_i are the ith columns of singular vectors U and V, respectively.

- For many systems, singular values decrease rapidly, and so, we can get a good approximation of A by truncating at some rank p:

$$A \approx A_p = \sum_{i=1}^{p} \sigma_i u_i v_i^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_p u_p v_p^T$$

Truncated SVD

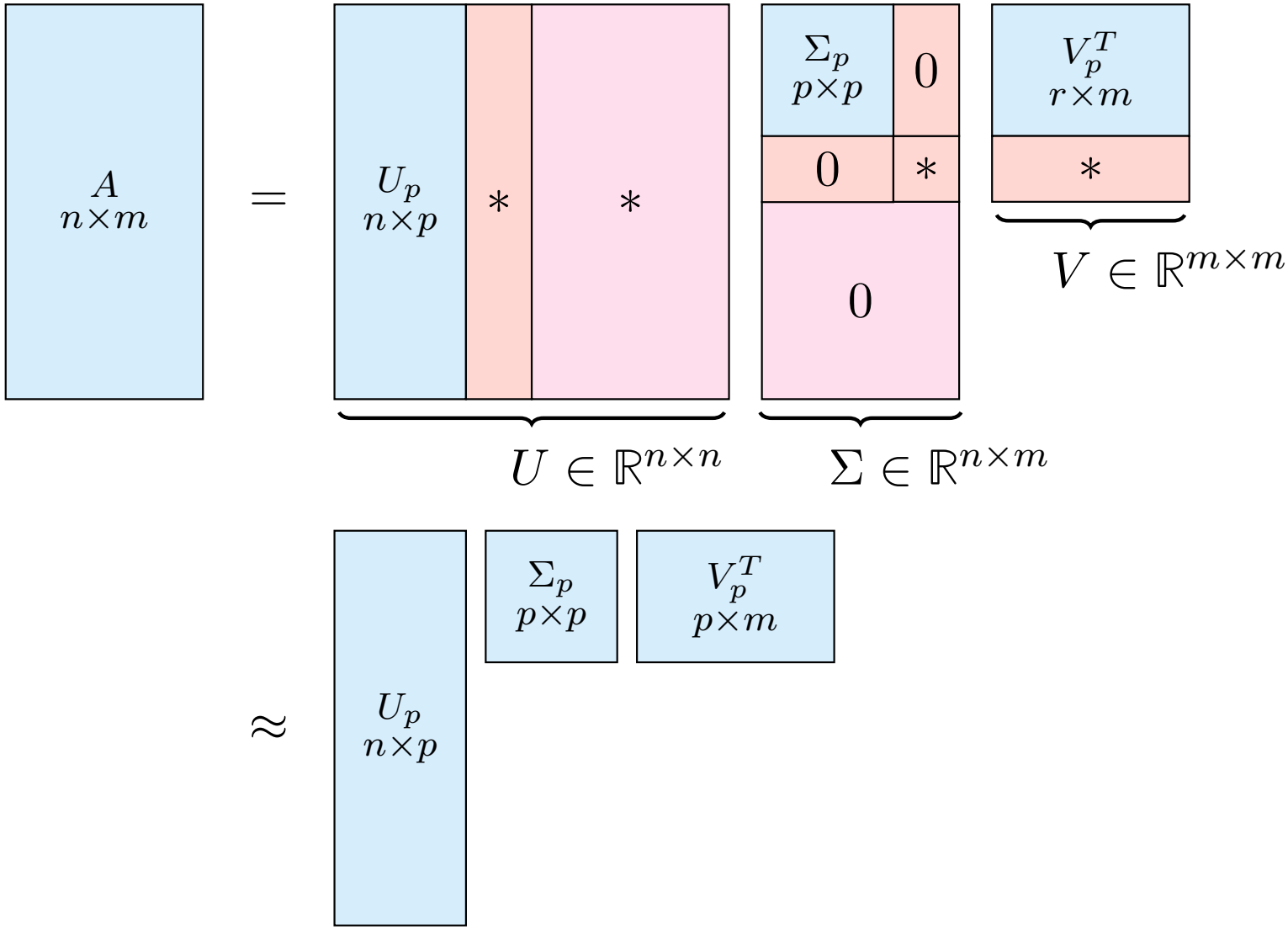


Image Compression via Truncated SVD

Consider a grayscale image with n pixels in the vertical direction and m pixels in the horizontal direction

- Let A be an $n \times m$ matrix representation of a 8-bit grayscale image
- pixel values denote light intensities from 0 (black) to 255 (white)

(UP Diliman, 2021)

$A \in \mathbb{R}^{4000 \times 3000}$

$p = 20$

about 50% image variance

$p = 200$

almost 80% image variance

Hands-on Activity

Take a picture with a new friend from this workshop. Convert your colored digital image to blue B, green G, and red R intensity matrices. For each intensity matrix (channel), take its singular value decomposition and plot the singular values and corresponding image variance. Determine the lowest possible p so that the truncated SVDs yield at least 80% image variance in all color channels. Plot the corresponding compressed colored image.

– BREAK –
15:00 – 15:30

LINEAR SPARSE SYSTEMS AND ITERATIVE METHODS

Linear Sparse Systems

- A linear system is said to be sparse if the matrix $A \in \mathbb{R}^{n \times n}$ has a number of nonzero entries of order of n (and not n^2).
  - For example, matrix

$$A = \begin{bmatrix} * & * & * & * & * \\ * & * & & & \\ * & & * & & \\ * & & & * & \\ * & & & & * \end{bmatrix}$$

    has $3n - 2$ nonzero entries. After the first step of LU decomposition, the matrix becomes dense with $n(n-1)+1$ nonzero entries.
  - However, reordering rows and columns of matrix

$$A = \begin{bmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \\ * & * & * & * & * \end{bmatrix}$$

    minimizes the number of nonzero entries in its decomposition.

Sparse Systems

Storage formats for sparse matrices can be divided into two groups:

- for efficient modification, such as coordinate (COO) format, which are typically used to contruct matrices
  - COO stores a list of (row, column, value) tuples.
- for efficient access and matrix operations, such as Compressed Sparse Row/Column (CSR/CSC) Format
  - It is similar to COO, but compresses the row/column indices, hence the name
  - CSR stores a sparse $m \times n$ matrix in row form using three 1D arrays: (1) matrix entries, (2) corresponding column indices, and (3) total number of nonzeros above each row

Iterative Methods for Linear Systems

Solve a system of n linear equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= y_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_2 &= y_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= y_n \end{aligned}$$

Given a matrix $A = [a_{ij}] \in \mathbb{C}^{n\times n}$ and $y \in \mathbb{C}^n$, find $x \in \mathbb{C}^n$ such that

$$Ax = y$$

- take an initial guess $x_0 \in \mathbb{C}^n$
- construct a sequence of iterates $\{\mathbf{x}_k\} \subset \mathbb{C}^n$ such that

$$x_k \to x, \quad \text{as } k \to \infty$$

Iterative Methods for Linear Systems

- Transform $Ax = y$ into an equivalent fixed-point form.
- Decompose

$$A = D + A_L + A_U$$

into diagonal matrix $D = \text{diag}(a_{11}, \ldots, a_{nn})$ and proper lower and upper triangular matrices

$$A_L = \begin{bmatrix} 0 & & & \\ a_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix} \qquad A_U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix}$$

respectively.

Iterative Methods for Linear Systems

- Jacobi Method (Simultaneous Displacements). Suppose D is nonsingular, then linear system $Ax = y$ is transformed into an equivalent form

$$x = -D^{-1}(A_L + A_U)x + D^{-1}y$$

This is solved by successive approximations

$$x_{k+1} = -D^{-1}(A_L + A_U)x_k + D^{-1}y, \qquad k = 0, 1, \ldots$$

with arbitrarily chosen initial x_0

Written in components,

$$x_{(k+1),i} = -\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_{k,j} + \frac{y_i}{a_{ii}}, \qquad i = 1, \ldots, n$$

Iterative Methods for Linear Systems

- Gauss-Seidel Method (Successive Displacements). The linear system $Ax = y$ is transformed into an equivalent form

$$x = -(D + A_L)^{-1} A_U x + (D + A_L)^{-1} y,$$

which is solved by successive approximations

$$x_{k+1} = -(D + A_L)^{-1} A_U x_k + (D + A_L)^{-1} y, \qquad k = 0, 1, \ldots$$

with arbitrarily chosen initial x_0

In actual computations, we solve linear system

$$(D + A_L) x_{k+1} = -A_U x_k + y$$

Written in components,

$$x_{(k+1),i} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_{(k+1),j} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_{k,j} + \frac{y_i}{a_{ii}}, \qquad i = 1, \ldots, n$$

Try this!

Write a code that implements Gauss-Seidel Method. Generate a ”noisy” tridiagonal matrix A of your choice, and a random vector y. Use your code, to solve linear system $Ax = y$, and plot your solution.

– END OF DAY 2 –

THANK YOU SO MU–