

Linear Algebra

Singular Value Decomposition

Automotive Intelligence Lab.

*Dt
Cs | Rd.*



HANYANG UNIVERSITY



Contents

- Definition of singular value decomposition
- Geometric Implications of SVD
- SVD with different input and output dimensions
- Calculate method of SVD
- Purpose of SVD
- Application of SVD

Definition of Singular Value Decomposition



Singular Value Decomposition

- For a set of orthogonal vectors,
 - ▶ Orthogonal set whose size changes after a linear transformation but still Orthogonal.
- Call singular value decomposition as SVD.

SVD: One of the Matrix Decomposition Methods

- SVD allows to decompose random $m \times n$ matrix A as:

$$A = U\Sigma V^T$$

A : $m \times n$ rectangular matrix

U : $m \times m$ orthogonal matrix

Σ : $m \times n$ diagonal matrix

V : $n \times n$ orthogonal matrix

Four matrix's size and properties

Supplementary Explanation about Previous Page

■ Property of orthogonal matrix U .

- ▶ $U^T U = U U^T = \boxed{I}$.
- ▶ $U^T = \boxed{U^{-1}}$.

■ Property of diagonal matrix Σ .

- ▶ Form of a matrix of size $m \times n$.

$$\begin{pmatrix} p_1 & 0 \\ 0 & p_2 \end{pmatrix}$$

$$\begin{pmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & p_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

$m = n = 2$

$m > n$

$m < n$

$$\begin{pmatrix} p_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 & 0 & \cdots & 0 \\ & & \ddots & & & & \\ 0 & 0 & \cdots & p_m & 0 & \cdots & 0 \end{pmatrix}$$

Geometric Implications of SVD



HANYANG UNIVERSITY



Example In a Two-Dimensional Vector Space

■ Can always find Orthogonal Vector to a given vector.

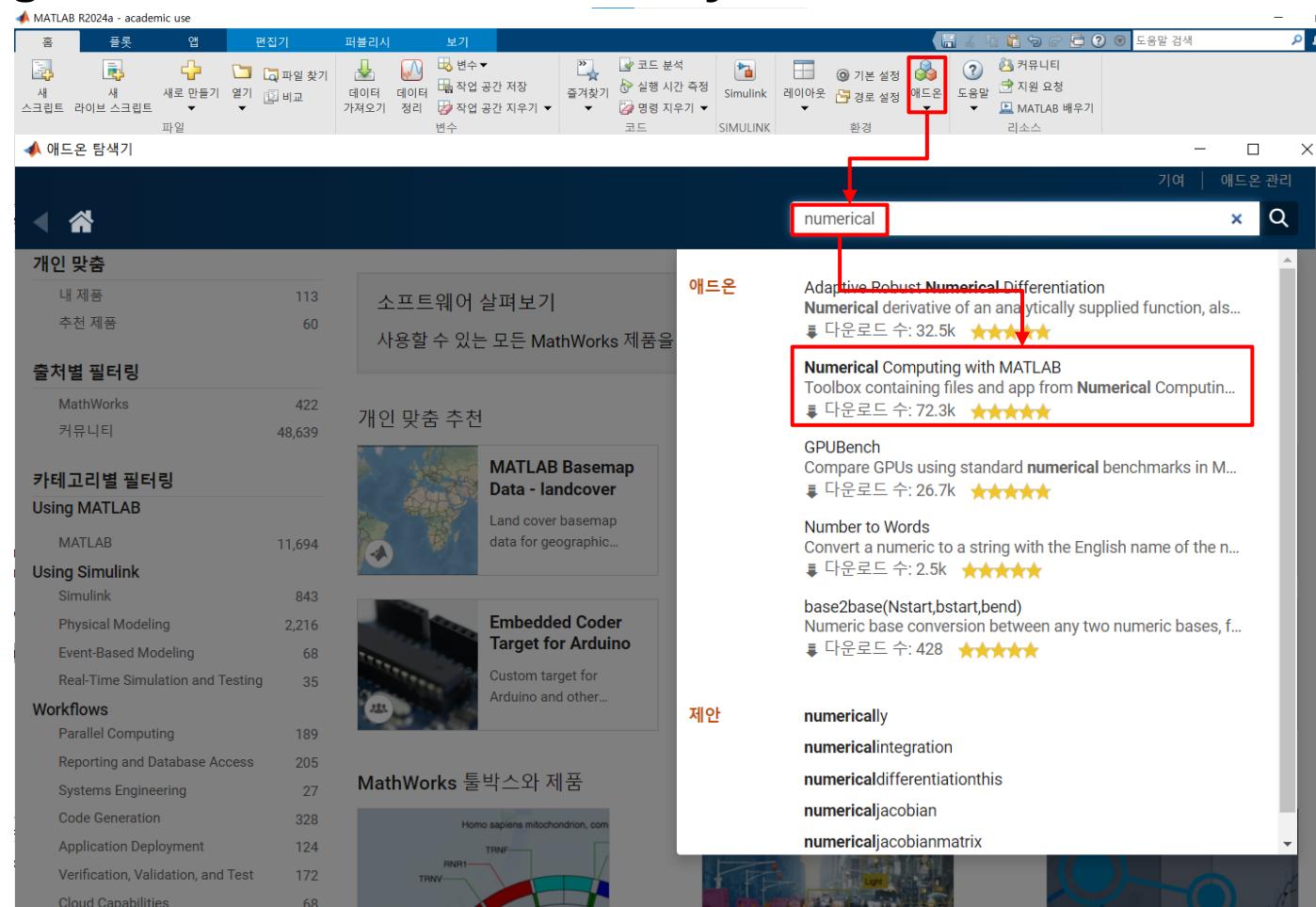
- ▶ In case of given vector is in a two-dimensional real vector space.
- ▶ Formal method
 - Using Gram-Schmidt process

■ If same linear transformation is taken for two orthogonal vectors,

- ▶ Those two vectors are not guaranteed to be orthogonal.
- ▶ Example of this is on the next page.

Preparation for 'Numerical Computing toolbox'

- You need 'Numerical Computing Toolbox' to run the code in this lecture.
- Follow the procedure to install the toolbox
- Also add the given files to current directory.



Code Exercise of Visualization of Orthogonality After Linear Transformation

■ Code Exercise (14_01)

- ▶ Visualize the orthogonality of matrix after linear transformation.
- ▶ Run the code and type anything in command window, then the figure will appear.

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% REQUIREMENT
% You need 'Numerical Computing with MATLAB toolbox' to use 'eigshow()'

A =[1 3;4 2]/4;
n_steps = 100;
step_mtx = eye(2);
[x, y] = ndgrid(-1:0.15:1);
xy_min = min(min(A*[x(:), y(:)]'))*1.5;
xy_max = max(max(A*[x(:), y(:)]'))*1.5;

dot_colors = jet(length(x(:)));

xlim([xy_min, xy_max]);
ylim([xy_min, xy_max]);
pause;
for i_steps = 1:n_steps
    step_mtx = (A-eye(2))/n_steps*i_steps;

    new_xy = (eye(2)+step_mtx)*[x(:), y(:)]';
    scatter(new_xy(1,:), new_xy(2,:),30,dot_colors,'filled')
    grid on;
    xlim([xy_min, xy_max]); ylim([xy_min, xy_max]);
    pause(0.01);
end
```

```
% Animation with circle
t=linspace(0,2*pi,100);
x=cos(t);
y=sin(t);
plot(x,y);
[temp] = A*[x;y];
plot(temp(1,:),temp(2,:))
XLIM=[xy_min, xy_max];
YLIM=[xy_min, xy_max];

% Animation
figure;
plot(x,y);
grid on;
xlim(XLIM);
ylim(YLIM);
pause;

for i_steps = 1:n_steps
    step_mtx = (A-eye(2))/n_steps*i_steps;
    new_xy = (eye(2)*step_mtx)*[x;y];
    plot(new_xy(1,:), new_xy(2,:));
    grid on;
    xlim(XLIM);
    ylim(YLIM);
    pause(0.01);
end

% Eigshow
figure;
eigshow(% Todo)
```

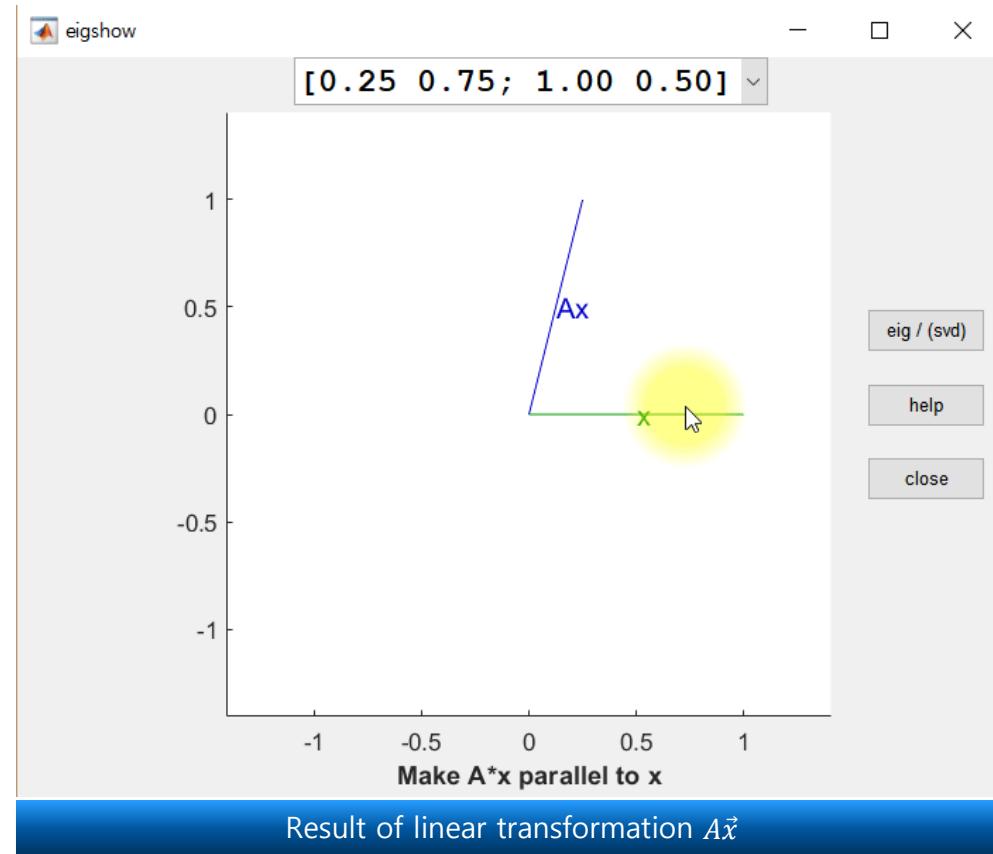
MATLAB code of visualize the orthogonality



Visualization of Orthogonality After Linear Transformation

■ Figure below shows results of linear transformation $A\vec{x}$.

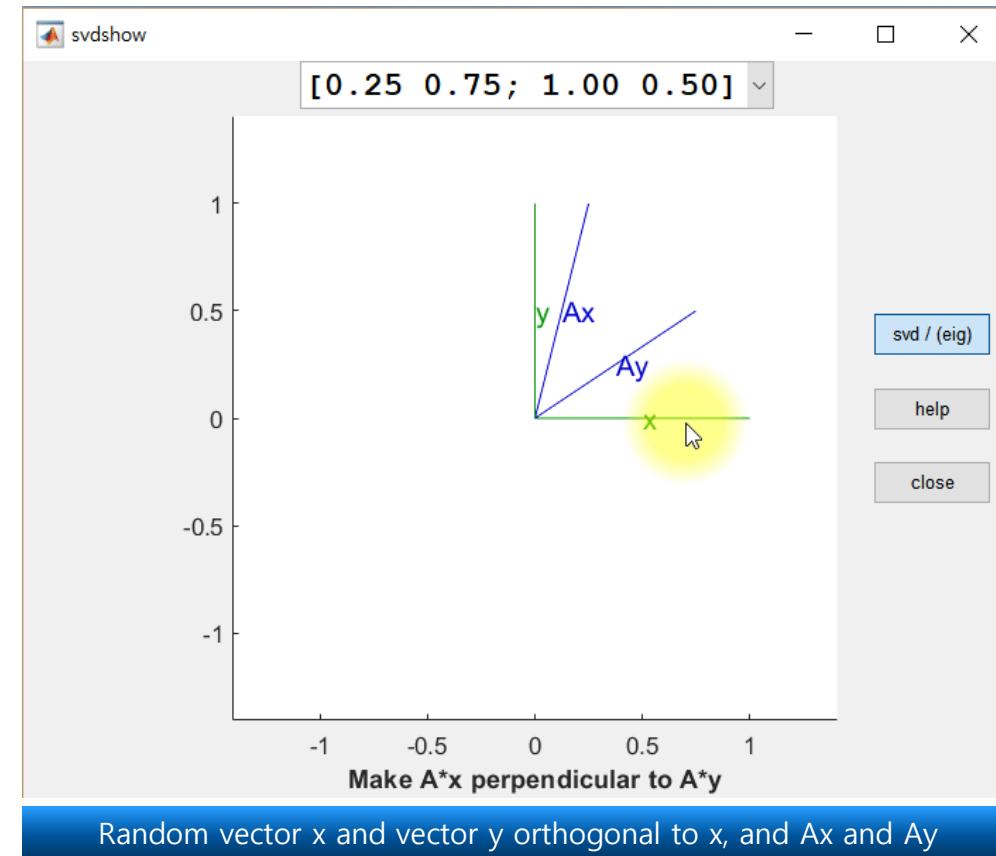
- Matrix $A = \begin{pmatrix} 0.25 & 0.75 \\ 1 & 0.5 \end{pmatrix}$, random vector \vec{x} .



Two Orthogonal Vectors Remain Orthogonal After Linear Transformation

■ Figure below shows results of linear transformation $A\vec{x}, A\vec{y}$.

- ▶ \vec{x} and \vec{y} are two orthogonal vectors.



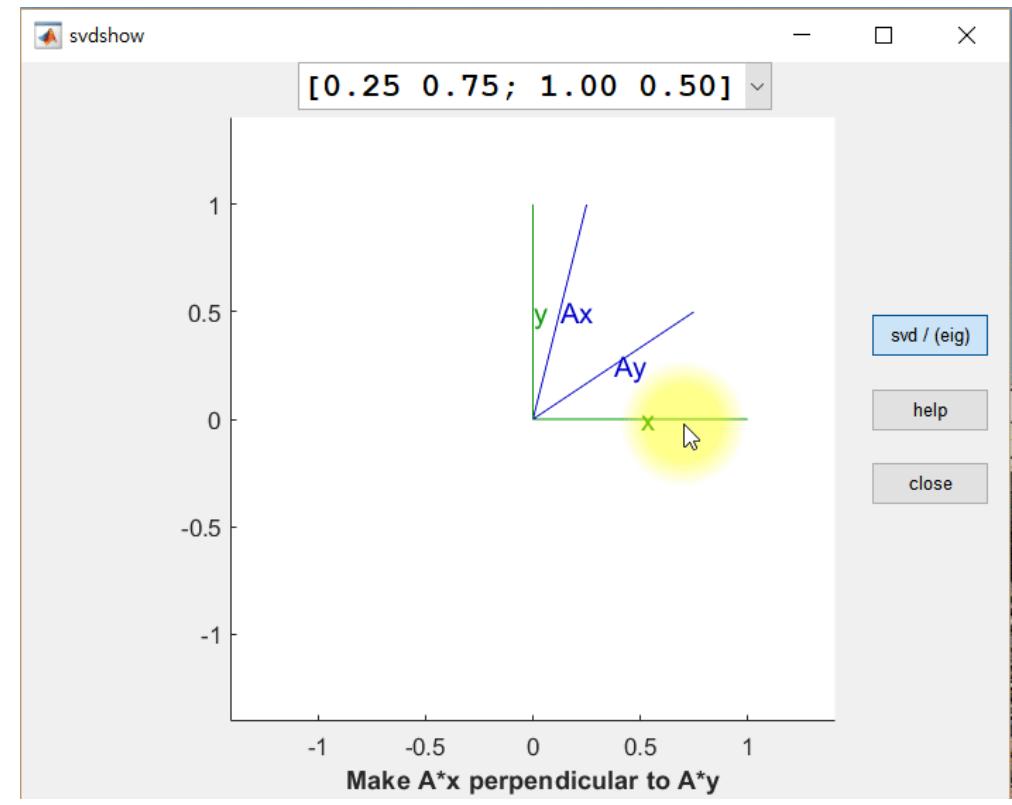
Two Main Things to Note In The Figure

- Not just one case where are orthogonal.

- ## ■ Length changed slightly.

- ▶ After \vec{x} and \vec{y} are converted through a matrix A .
 - ▶ These changed length value are **scaling factor**.
 - Generally called Singular Value.
 - Start from the largest values: $\sigma_1, \sigma_2, \dots$

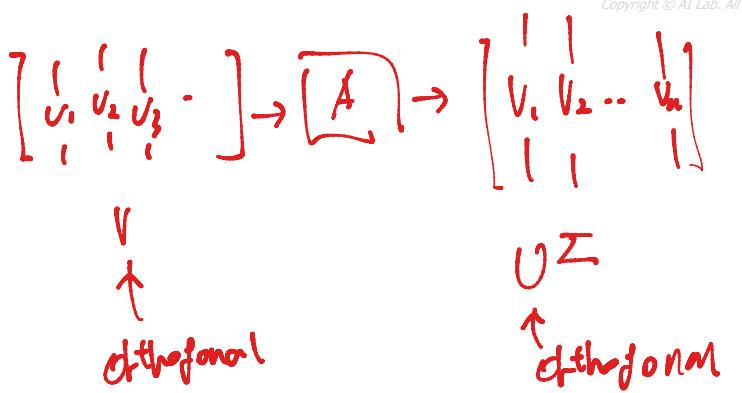
- ## ■ Let's go back to SVD.



Random vector x and vector y orthogonal to x , and Ax and Ay

Definition of SVD

$$AV = U\Sigma$$



■ $A = U\Sigma V^T \rightarrow AV = U\Sigma$

- ▶ V : Matrix of **Orthogonal** vectors **before** linear transformation.
- ▶ Σ : **Diagonal** matrix consisting of singular values.
- ▶ U : Matrix of **Orthogonal** vectors **after** linear transformation.
 - Each vectors are normalized to 1.

$$V = [\vec{x} \quad \vec{y}] \quad U = [\vec{u}_1 \quad \vec{u}_2] \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

Element matrices of SVD

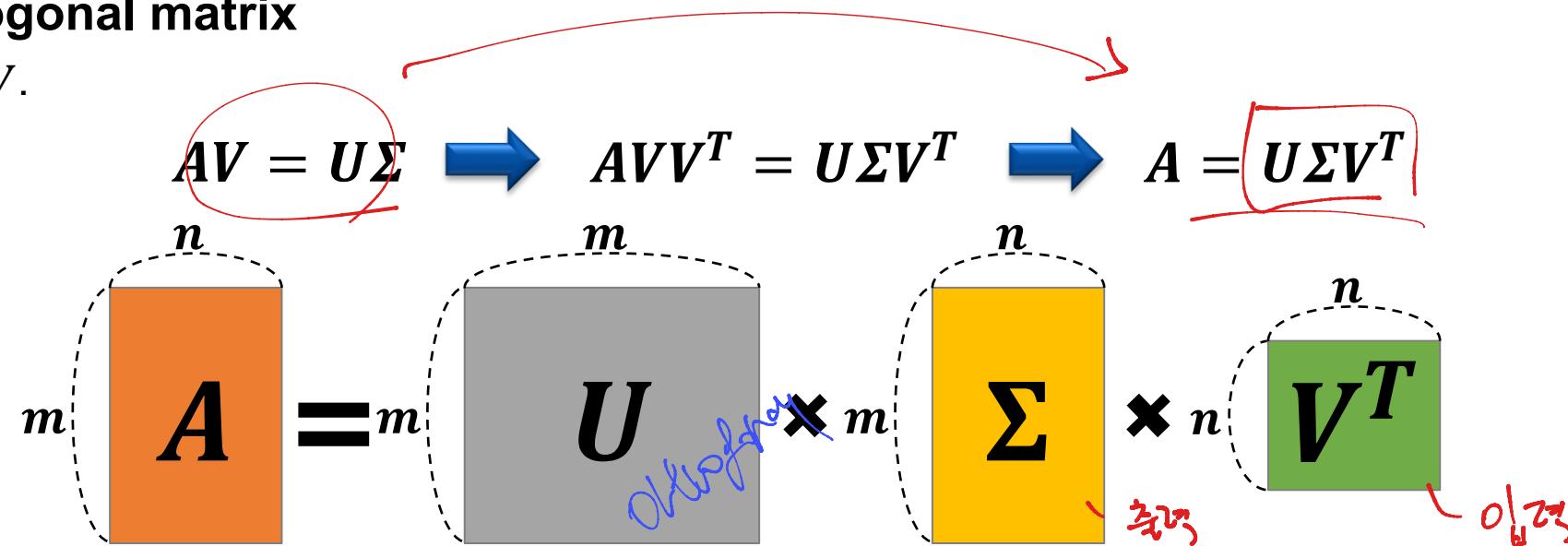
Relationships between 4 matrices (A, V, U, Σ)

From a linear transformation perspective: $AV = U\Sigma$.

- ▶ It gives you a question.
- ▶ Is it possible to find matrix U ?
 - After linearly transforming column vector in matrix V through matrix A .
 - Size of matrix V is changed by **singular value σ_1, σ_2** .
 - But column vectors in U are still **orthogonal**.

V is orthogonal matrix

- ▶ $V^{-1} = V$.



Visualization of the results of SVD of an arbitrary matrix A .

SVD with Different Input and Output Dimensions



Decomposition of Matrix A In Case of Non-Square Matrix

■ Matrix A is $m \times n$ dimensions.

- ▶ If A is 2×3 matrix,
 - Matrix A lowers the dimension from 3D to 2D.
- ▶ Question about what SVD requires:
 - Linearly transform 3 vectors that were orthogonal in a 3-dimensional space.
 - Convert them to 2 dimensions.
 - Is it possible to make two vectors orthogonal in a 2-dimensional space?

Code Exercise of Visualization of Linear Transformation by Unsquared Matrix A

■ Code Exercise (14_02)

- ▶ Visualize the linear transformation by unsquared matrix.
- ▶ Run the code and type anything in command window, then the figure will appear.

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Define A matrix and vectors
vector1 = [-1,2,1]';
vector2 = [1,1,1]';
A = [vector1/norm(vector1) vector2/norm(vector2)]*[vector1/norm(vector1)
vector2/norm(vector2)]';

% Animation with dots
[X,Y,Z] = ndgrid(-1:0.3:1);
n_steps = 100;
step_mtx = eye(3);
newXYZ = A*[X(:), Y(:), Z(:)]';
xyz_min = min(min(min([newXYZ(:, newXYZ(:, newXYZ(:))'])))*1.5;
xyz_max = max(max(max([newXYZ(:, newXYZ(:, newXYZ(:))])))*1.5;
LIMS = [xyz_min, xyz_max];

dot_colors = jet(length(X(:)));
figure(2)
scatter3(X(:, Y(:, Z(:,30, dot_colors,'filled'));
xlim(LIMS); ylim(LIMS); zlim(LIMS);

grid on;
hold on;
line([xyz_min, xyz_max], [0,0], [0,0], 'linewidth', 3)
line([0,0], [xyz_min, xyz_max], [0,0], 'linewidth', 3)
line([0,0], [0,0], [xyz_min, xyz_max], 'linewidth', 3)
xlabel('x'); ylabel('y'); zlabel('z')
hold on;
pause;

for i_steps = 1:n_steps
step_mtx = (A-eye(3))/n_steps*i_steps;

new_xyz = (eye(3)+step_mtx)*[X(:, Y(:, Z(:))'];
scatter3(new_xyz(1, :), new_xyz(2, :), new_xyz(3, :), 30, dot_colors,
'filled');
grid on;
hold on;
line([xyz_min, xyz_max], [0,0], [0,0], 'linewidth', 3)
line([0,0], [xyz_min, xyz_max], [0,0], 'linewidth', 3)
line([0,0], [0,0], [xyz_min, xyz_max], 'linewidth', 3)
hold off;
xlim(LIMS); ylim(LIMS); zlim(LIMS);
xlabel('x'); ylabel('y'); zlabel('z');
pause(0.01);
end

% SVD
[U,S,V] = %TODO%(A);
hold on;
for i = 1:3
mArrow3([0,0,0], [U(1, i)*S(i, i), U(2, i)*S(i, i), U(3, i)*S(i,i)], 'color', 'b');
hold on;
end

for i = 1:3
mArrow3([0,0,0], [V(1, i), V(2, i), V(3, i)], 'color', 'g');
end
```

MATLAB code of visualize linear transformation by unsquared matrix

Visualization of Linear Transformation by Non-Square Matrix A

■ Figure below shows transition from 3D vector space to 2D vector space.

- ▶ Matrix A is 2×3 matrix.
- ▶ Figure 1.
 - Transformation that projects 3 dimensional vectors onto a plane by matrix A .
- ▶ Figure 2, 3.
 - Apply SVD to matrix A , visualize vector orthogonal before and after linear transformation.
 - **Green** arrows: orthogonal vectors before linear transformation.
 - **Blue** arrows: orthogonal vectors after linear transformation.

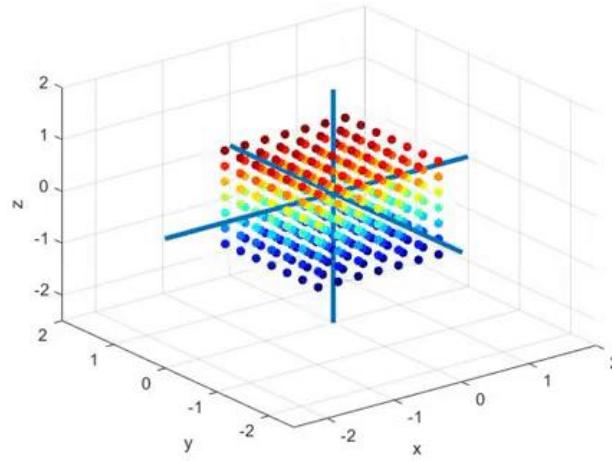


Figure 1.

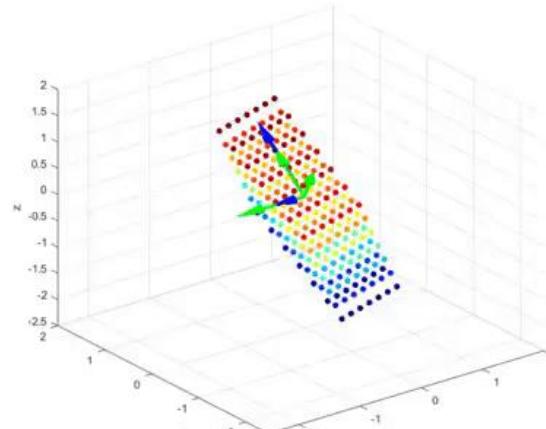


Figure 2.

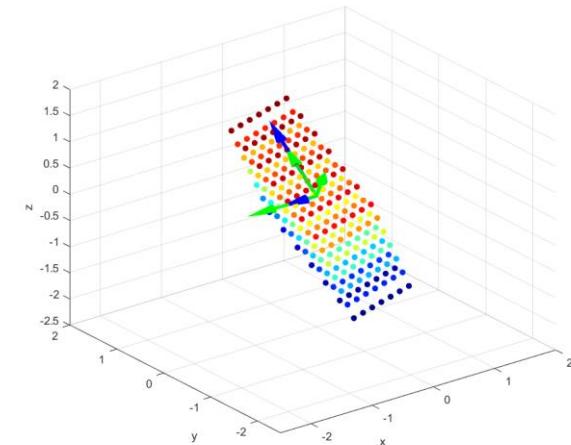


Figure 3.

Calculation Method of SVD



Example of SVD Calculation: Calculate U

- $A = U\Sigma V^T$
- $A^T = V\Sigma^T U^T$

Eigen decomposition

- $AA^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T = U\Sigma^2 U^T = U\Sigma^2 U^{-1}$
- $A^T A = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T = V\Sigma^2 V^T = V\Sigma^2 V^{-1}$

$$\boxed{V} \rightarrow \boxed{A} \rightarrow \boxed{U\Sigma}$$

$$AV = V\Lambda$$

$$\boxed{A = V\Lambda V^{-1}}$$

Eigen decomposition

Example of SVD Calculation: Calculate U

■ $A = U\Sigma V^T$

▶ Calculate U (left singular vector).

- Calculate AA^T .

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \quad AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

- Calculate eigenvalues and eigenvectors of AA^T .

- Arrange eigenvectors in order of largest eigenvalue.

$$\lambda_1 = 10 \rightarrow \vec{v}_1 = [1 \quad -1]$$

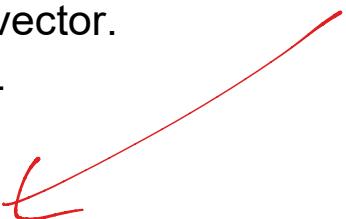
$$\lambda_2 = 12 \rightarrow \vec{v}_2 = [1 \quad 1]$$

→ $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

- Normalize each eigenvector.

- Then, you can get U .

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$



Process to calculate U

$$\begin{aligned} \Delta &= \begin{bmatrix} 10 & 0 \\ 0 & 12 \end{bmatrix} \\ \frac{1}{\sqrt{2}} &= \begin{bmatrix} \frac{\sqrt{10}}{2} & 0 \\ 0 & \frac{\sqrt{12}}{2} \end{bmatrix} \end{aligned}$$

Example of SVD Calculation: Calculate V^T

■ $A = U\Sigma V^T$

▶ Calculate V (right singular vector).

- Calculate $A^T A$.

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \quad A^T A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

- Calculate eigenvalues and eigenvectors of $A^T A$.

- Arrange eigenvectors in order of largest eigenvalue.

$$\begin{aligned} \lambda_1 &= 12 \rightarrow \vec{v}_1 = [1 \ 2 \ 1] \\ \lambda_2 &= 10 \rightarrow \vec{v}_2 = [2 \ -1 \ 0] \\ \lambda_3 &= 0 \rightarrow \vec{v}_3 = [1 \ 2 \ -5] \end{aligned} \xrightarrow{\text{ }} \begin{bmatrix} 1 & 2 & 1 \\ 2 & -1 & 0 \\ 1 & 0 & -5 \end{bmatrix}$$

- Normalize each eigenvector.

- Then you can get V .

$$V = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{6}} & 0 & -\frac{5}{\sqrt{30}} \end{bmatrix} \xrightarrow{\text{ }} V^T = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & -\frac{5}{\sqrt{30}} \end{bmatrix}$$

Process to calculate V

2/21 6h 30m

Example of SVD Calculation: Calculate Σ

■ $A = U\Sigma V^T$

► Calculate Σ .

- Σ is $m \times n$ **rectangular** diagonal matrix.
 - Same size of matrix A .
- It's diagonal elements:
 - Square root of eigenvalues obtained through eigenvalue decomposition of matrices $A^T A$ or AA^T .
- Arrange the values diagonally starting from the **largest** value.

$$AA^T = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \rightarrow \lambda_1 = 10, \lambda_2 = 12$$

$$A^T A = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} \rightarrow \lambda_1 = 12, \lambda_2 = 10, \lambda_3 = 0$$

$$\Sigma = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

Process to calculate Σ

Reduced SVD



HANYANG UNIVERSITY



Full SVD

■ Decomposing $m \times n$ matrix A into SVD as shown Fig 1..

- ▶ Only in case of $m > n$.

■ In reality...,

- ▶ It is rare to perform full SVD.
- ▶ It is common to perform reduced SVD.

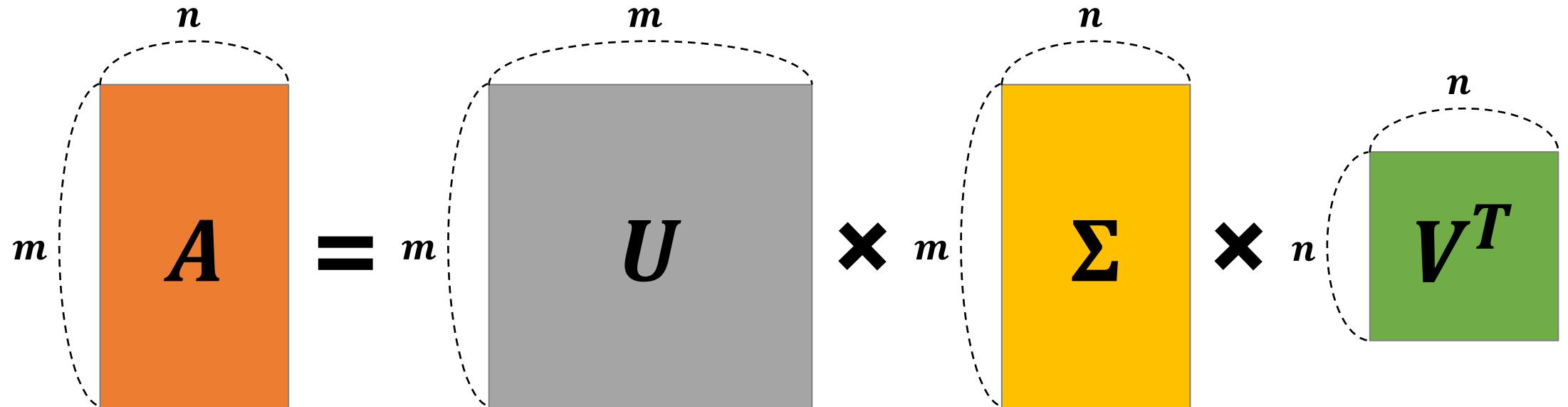


Fig 1. Full SVD

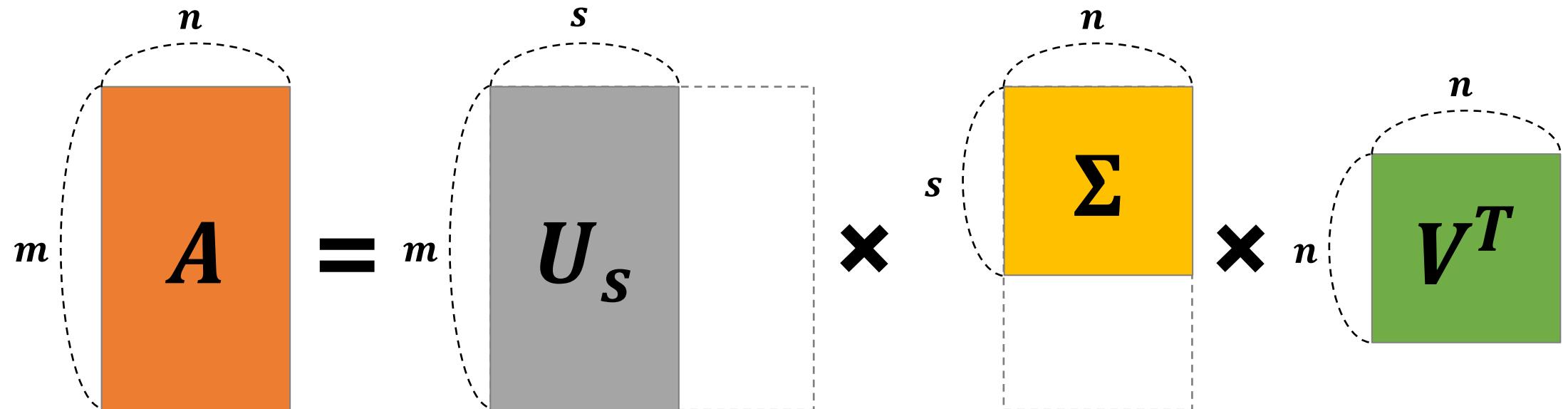
Reduced SVD: Thin SVD

■ Assume

- ▶ $s = n$

■ Form

- ▶ In Σ
 - Non-diagonal part consisting of 0 is removed.
- ▶ In U
 - Corresponding column vectors are removed.



Reduced SVD: Compact SVD

■ Assume

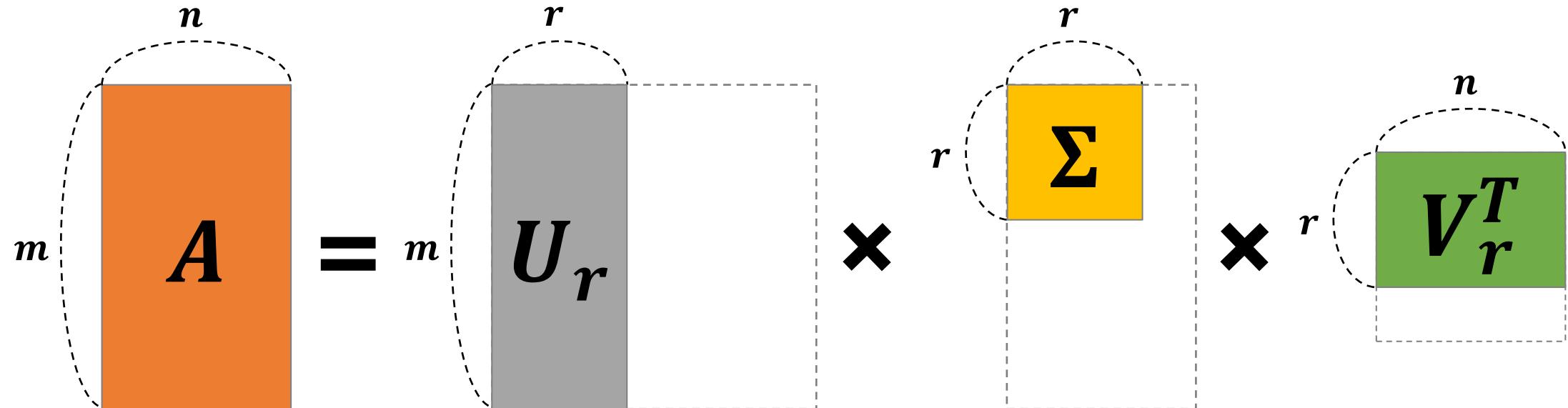
- ▶ r is number of non-zero values, among $s = n$ singular values.

■ Form

- ▶ In Σ
 - Not only off-diagonal elements but also singular values of 0 are removed.

■ It can be easily confirmed.

- ▶ Calculated A is same matrix as original A .



Compact SVD

Reduced SVD: Truncated SVD

~~AI~~

■ Assume

- ▶ $t < r$
 - r is number of non-zero values, among $s = n$ singular values.

diagonal truncated

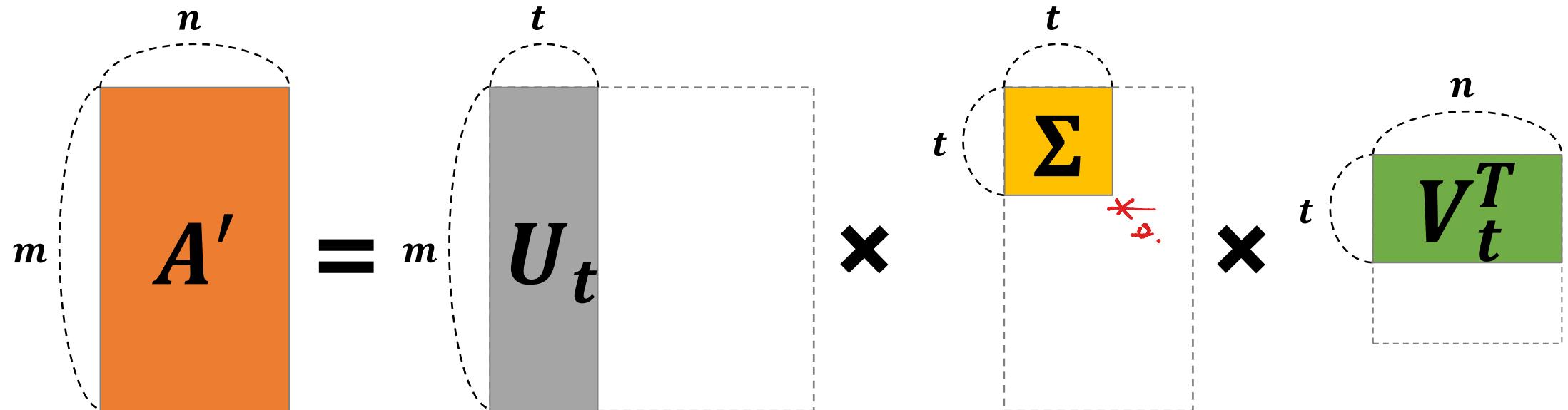
0 0 0 0 0
0 0 0 0 0

0 0 0 0 0
0 0 0 0 0

■ Form

- ▶ In Σ
 - Even singular values other than 0 are removed.

■ Approximation matrix A' for A is produced.



Truncated SVD

Reduced SVD: Truncated SVD

Rewrite formula for SVD below.

$\vec{u}_1 \vec{v}_1^T$: $m \times n$ matrix.

- Component values in matrix $\vec{u}_1 \vec{v}_1^T$ have values between -1 and 1 .
 - Because \vec{u}_1 and \vec{v}_1 are normalized vector.

$\sigma_1 \vec{u}_1 \vec{v}_1^T$: size of this matrix is determined by σ_1 .

Decompose a random matrix A into several matrices with same size of A matrix.

By using SVD!

Size of element value of each decomposed matrix is determined by σ .

$$\begin{aligned}
 A &= U \Sigma V^T \\
 &= \left[\vec{u}_1 \vec{u}_2 \cdots \vec{u}_m \right] \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \\ & & \sigma_m \end{bmatrix} \left[\vec{v}_1^T \vec{v}_2^T \cdots \vec{v}_n^T \right]^T \\
 &= \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_m \vec{u}_m \vec{v}_m^T
 \end{aligned}$$

Formula of SVD

Truncated

*(If val ↓
size ↓)*

Matrix A'

■ Matrix approximated by truncated SVD.

■ Rank t matrix

- ▶ Minimize matrix norm $\|A - A'\|$

■ Application

- ▶ Data compression
- ▶ Noise removal

Application of SVD



HANYANG UNIVERSITY



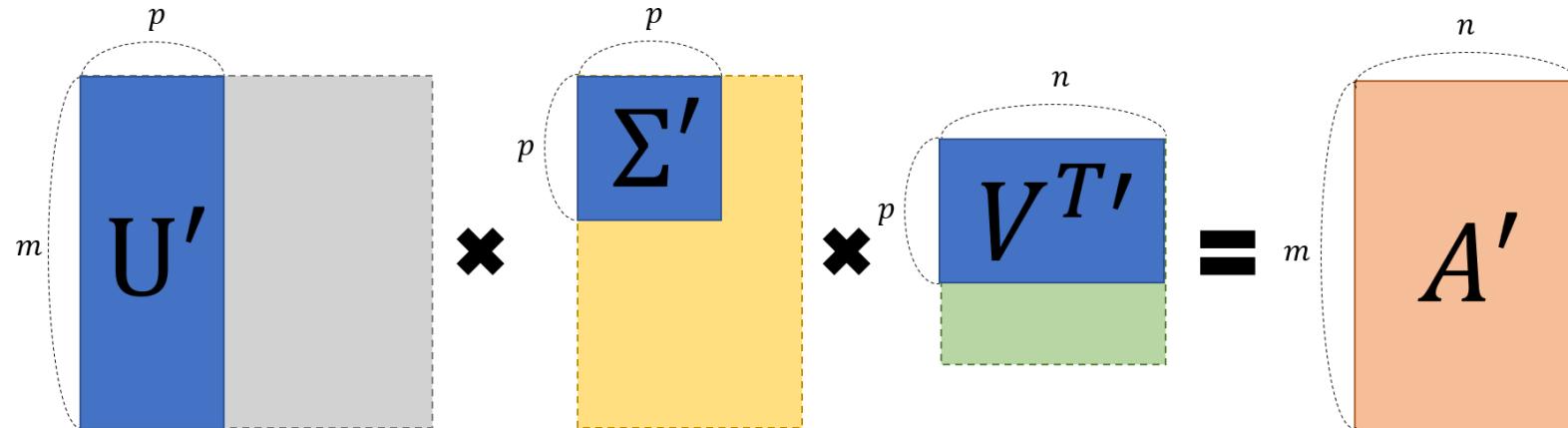
Use of SVD

SVD shines its applicability in the process of reCombining decomposed matrix.

- Rather than in the decomposition process.

Decomposed matrix A can be partially restored.

- Using only p singular values.
- Amount of information A is determined by size of singular value.
 - Sufficient useful information can be maintained even with several large singular values.



Process of partially restoring appropriate A' using only part of the U .

Code Exercise of Partial Restoration Process Through Photos

■ Code Exercise (14_03)

- ▶ Run the code with given image 'lena_std.tif'.

```
% Clear workspace, command window, and close all figures  
clc; clear; close all;  
  
% Use image  
img = double(rgb2gray(imread('lena_std.tif')));  
  
[U, S, V] = svd(img);  
figure;  
  
for i = 1:size(S,1)  
    imagesc(U(:, 1:i)*S(1:i, 1:i)*V(:, 1:i)');  
    colormap('gray')  
    name = ['layers added upto ', num2str(i)];  
    title(name);  
    if i<30  
        pause(0.5)  
    elseif 3<=i<100  
        pause(0.1)  
    else  
        pause(0.01)  
    end  
end
```

MATLAB code of partial restoration process



Partial Restoration Process Through Photos

■ Use only important information

- ▶ Size of the photo will be reduced.
- ▶ But still be able to **preserve** the content that the photo wants to show.



Can apply SVD to partially restore the photo

Example of Data Compression

■ Let's compress 600×367 image with SVD.

- ▶ 1. Take 600×367 matrix A with pixel values of this image as element values.
- ▶ 2. Perform truncated SVD.
 - Rotate original image by 90 degrees.
 - To create typical $m > n$ form
 - Apply SVD.
- ▶ 3. Obtain approximation matrix A' .
- ▶ 4. Display it as image again.



One of Wallis' "dressed to kill" advertising images

Display Result of Data Compression as Image

■ Approximation with t singular values



Original image



Approximation with 100 singular values



Approximation with 50 singular values



Approximation with 20 singular values

Numerical Result of Data Compression

■ Memory

- ▶ In original image
 - 220,200
 - $367 * 600$
- ▶ In case of $t = 20$
 - 19,360
 - $600 * 20(\mathbf{U}) + 20(\mathbf{\Sigma}) + 20 * 367(\mathbf{V})$

■ Data compression ratio

- ▶ 8.8%
- $19,360 / 220,200 * 100$

■ Looking at image quality

- ▶ It is not good compression method.



Data compression with 8.8% can represent original image well

■ But...,

- ▶ Data approximation through truncated SVD captures core of original data well.

■ We will practice data compression with SVD in next week!

SVD and Pseudo Inverse

Least Square



Linear System $Ax = b$

■ If inverse matrix of A exists

- ▶ Solution to this system can be easily found as $A^{-1}b$.
- ▶ However, in most real problem
 - There are very few cases where **inverse matrix exists**.
- ▶ In such cases
 - **Pseudo Inverse** can be used!

■ If inverse matrix of A doesn't exist

- ▶ Solution to this system can be calculated as $x = A^+b$.
- A^+ is **Pseudo Inverse** of A .
- ▶ x becomes solution.
- Minimizes $\|Ax - b\|$.

■ Finding solution using **pseudo inverse** has same meaning.

- ▶ As the least square method.

Pseudo Inverse

■ It can be defined for random $m \times n$ matrices.

- ▶ Originally, inverse matrix was defined only for square matrices.

■ SVD

- ▶ One of the most powerful (and stable) methods
 - To compute **pseudo inverse**.

■ If SVD of matrix A is Eq 1.

- ▶ Pseudo inverse of A is Eq 2.

- Σ^+ is matrix obtained by taking **reciprocal** of **Singular** values in original Σ and then **transposing** it.

$$A = U\Sigma V^T$$

Eq 1. SVD of matrix A

$$A^+ = V\Sigma^+ U^T$$

Eq 2. Pseudo inverse of matrix A

$$\begin{aligned} A^+ &= (A^T A)^{-1} A^T \\ &= (V \Sigma U^T U \Sigma V^T)^{-1} V \Sigma U^T \\ &= (V \Sigma^2 V^T)^{-1} V \Sigma U^T \\ &= (V^T)^{-1} \Sigma^{-2} V^{-1} V \Sigma U^T \\ &= V \Sigma^{-2} \Sigma U^T \\ &= V \Sigma^{-1} U^T \end{aligned}$$

Eq 3. Derivation process of pseudo inverse of matrix A

Note about Pseudo Inverse

- Order of U and V changes.
- Σ also changes from $m \times n$ to $n \times m$ matrix.

- If 0 is included among singular values
 - ▶ Only non-zero singular values are reciprocated.
 - ▶ Original 0 is left as 0 in Σ^+ .
 - Reciprocal is only for non-zero values.

$$A = U \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_s \\ & & 0 \end{pmatrix} V^T \rightarrow A^+ = V \begin{pmatrix} 1/\sigma_1 & & \\ & \ddots & \\ & & 1/\sigma_s \\ & & 0 \end{pmatrix} U^T$$

SVD of matrix A to pseudo inverse of matrix A

How to Find Identity Matrix

■ If all singular values are positive

► $m \geq n$ — tall

- A^+A becomes $n \times n$ identity matrix.

► $m \leq n$ — wide

- AA^+ becomes $m \times m$ identity matrix.

$$AA^{-1} = I$$

$$A^{-1}A = I$$

Square matrix of $n \times n$

■ If singular values of 0 is included

► No matter what order you multiply it, **you will not get I matrix.**

Tall — $A^+A = (V\Sigma^+U^T)(U\Sigma V^T) = V\Sigma^+\Sigma V^T = VV^T = E_n \quad (m \geq n)$

Wide — $AA^+ = (U\Sigma V^T)(V\Sigma^+U^T) = U\Sigma\Sigma^+U^T = UU^T = E_m \quad (m \leq n)$

How to find identity matrix when all singular values are positive

Apply Pseudo Inverse to System of Linear Equation

■ $m \geq n$ (usually)

- ▶ Number of **Equations (data)** is greater than number of **Unknown**.
- ▶ Multiply front of both sides of $Ax = b$ by A^+ .
- ▶ Find x in form $A^+Ax = A^+b \Rightarrow x = A^+b$.

■ $m < n$

- ▶ Number of **Unknown** is greater than number of **Equations (data)**.
- ▶ Solution is not uniquely determined.
- ▶ This is like problem of determining a plane using two points.
- ▶ Pseudo inverse can be obtained.
 - But since it must be used in form of multiplication after A as in Eq 1..
 - It is not valid for problems of form $Ax = b$.

Pseudo Σ 가 Σ 를 대체함.

$$AA^+ = (U\Sigma V^T)(V\Sigma^+ U^T) = U\Sigma\Sigma^+U^T = UU^T = E_m \quad (m < n)$$

How to find identity matrix when $m < n$

When Singular Value is Close to 0

■ Even if singular value is not 0, if it is very close to 0.

- ▶ It is common to treat it as **Noise**.

- Change it to 0.
- Then obtain **pseudo inverse**.

■ Find pseudo inverse of matrix A using SVD.

- ▶ Find singular values.
- ▶ Change singular values to 0.
 - Very close to 0
- ▶ Take reciprocal of only singular values above threshold.
 - Default threshold used in MATLAB is $1e - 10$.
- ▶ Find pseudo inverse.



연산량 ↓
40[2 3] 강인

Threshold of Singular Value

- It is called *Tolerance* of SVD.
- It is closely related to truncated SVD.
- Resulting pseudo inverse and linear system solutions may be changed.
 - ▶ How much of value is considered noise.
 - ▶ How tolerance value is given.

Summary



HANYANG UNIVERSITY



Summary

- SVD can be decomposed into two orthogonal matrices and one diagonal matrix.
 - ▶ $A = U\Sigma V^T$
- SVD can be decomposed even if it is not a square matrix.
- SVD is widely used in data compression and noise removal process.
- SVD is the most powerful way to calculate pseudo inverse.

Code Exercises



HANYANG UNIVERSITY



Properties of a Symmetric Matrix

■ You learned that for a symmetric matrix, the singular values and the eigenvalues are the same. How about the singular vectors and eigenvectors? Use MATLAB to answer this question using a random $5 \times 5 A^T A$ matrix. Next, try it again using the additive method for creating a symmetric matrix ($A^T + A$). Pay attention to the signs of the eigenvalues of $A^T + A$.

```
% create a symmetric matrix
A = randn(5,5);
A = A' * A;

% eigendecomposition

% sorting them helps the comparison!

% SVD

% compare the eigenvalues and singular values
disp('Eigenvalues and singular values:')
disp([evals, s])

% now compare the left and right singular vectors
disp('Left-Right singular vectors (should be zeros)')
disp(round(U - V, 10)) % remember to compare V not V^T!

% then compare singular vectors with eigenvectors
disp('Singular vectors - eigenvectors (should be zeros)')
disp(round(U - evecs, 10)) % subtract and
disp(' ')
disp(round(U + evecs, 10)) % add for sign indeterminacy
```

Code sample



Economy SVD

- MATLAB can optionally return the “economy” SVD, which means that the singular vectors matrices are truncated at the smaller of M or N . Consult the docstring to figure out how to do this. Confirm with tall and wide matrices. Note that you would typically want to return the full matrices, economy SVD is mainly used for really large matrices and/or really limited computational power.

```
% sizes (try tall and wide)
m = 10;
n = 4;

% random matrix and its economy (aka reduced) SVD
A = ;
[U, S, V] = ;

% print sizes
disp(['Size of A: ', num2str(size(A, 1)), ' ', ' ', num2str(size(A, 2)), ']']);
disp(['Size of U: ', num2str(size(U, 1)), ' ', ' ', num2str(size(U, 2)), ']']);
disp(['Size of V': [', num2str(size(V, 1)), ' ', ' ', num2str(size(V, 2)), ']']);
```

Code sample



Properties of Orthogonal Matrix

- One of the important features of an orthogonal matrix (such as the left and right singular vectors matrices) is that they rotate, but do not scale, a vector. This means that the magnitude of a vector is preserved after multiplication by an orthogonal matrix. Prove that $\|Uw\| = \|w\|$. Then demonstrate this empirically in MATLAB by using a singular vectors matrix from the SVD of a random matrix and a random vector.

```
% The proof that |Uw| = |w| comes from expanding the vector magnitude to the dot product:  
% |Uw|^2 = (Uw)'(Uw) = w'U'U'w = w'Iw = w'w = |w|^2  
  
% random matrix size -> 5,5  
% empirical demonstration:  
[U, S, V] = ;  
w = randn(5, 1);  
  
% print out the norms  
disp(norm(U * w));  
disp(norm(w));
```

Code sample





**THANK YOU
FOR YOUR ATTENTION**



HANYANG UNIVERSITY

