MAEG 5720: Computer Vision in Practice

Supplementary Notes: Useful Mathematics I

Dr. Terry Chang

2021-2022

Semester 1

香港中文大學
The Chinese University of Hong Kong

Department of Mechanical and Automation Engineering
機械與自動化工程學系

Content

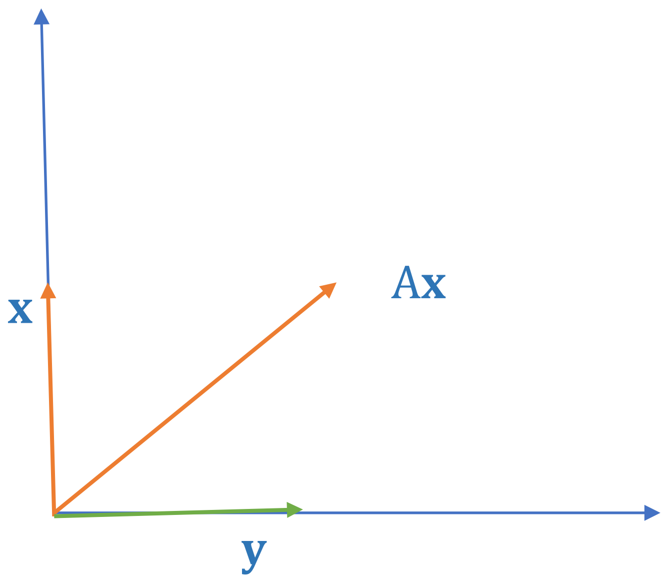
- Eigen-Value and Eigen-Vector Decomposition
- Singular Value Decomposition
    - Image compression
    - Pseudo-inverse, least-square and regression
    - Least-square on homogenous linear equations

Eigen values and Eigen Vector

- Consider a 2x2 matrix $A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$ and vector $\mathbf{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$$\mathrm{A}\mathbf{x} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

- Direction of $\mathbf{x}$ changed



- Consider another vector $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$$\mathrm{A}\mathbf{x} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- Direction of $\mathbf{y}$ unchanged

Meaning of Eigenvalue and Eigenvector

- Consider the second case

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

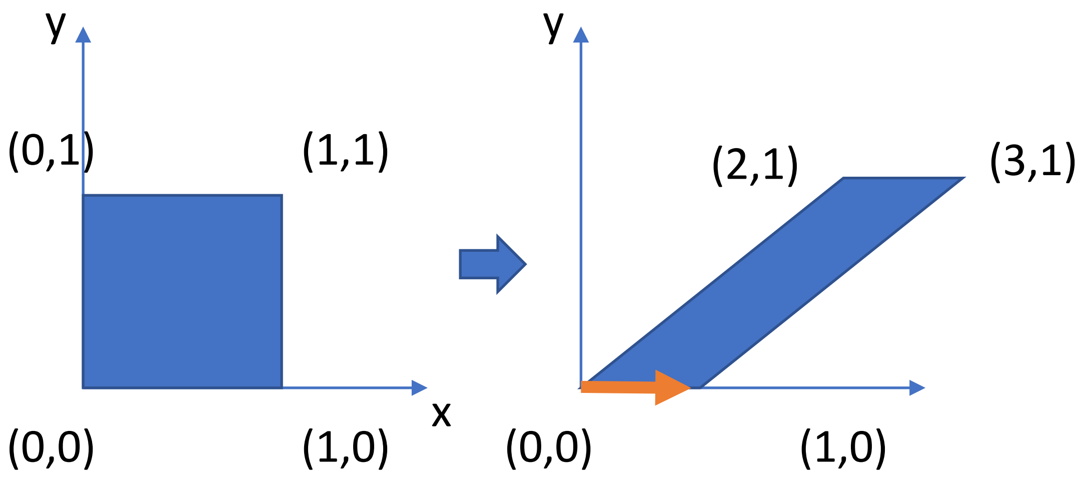
We can write

$$A\mathbf{y} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \lambda\mathbf{y}$$

where $\boldsymbol{\lambda}$=1

$\boldsymbol{\lambda}$=1 is the *eigenvalue*

$\boldsymbol{y}$ is corresponding *eigenvector*



Eigenvalue/Eigenvector

- Definition of ***Eigenvalue***/***Eigenvector***
  - Given a square matrix $\mathrm{A}_{m\times m}$ and vector $\mathbf{x}$ represented below

$$\mathrm{A}\mathbf{x} = \boldsymbol{\lambda}\mathbf{x}$$

where

$\boldsymbol{\lambda}$ is called the ***eigenvalue*** of A with the corresponding ***eigenvector*** $\mathbf{x}$

How to find the eigenvalue?

Let

$$\mathrm{A}\mathbf{X} = \boldsymbol{\lambda}\mathbf{X}$$

we have

$$\mathrm{A}\mathbf{X} - \boldsymbol{\lambda}\mathbf{X} = \mathbf{0}$$

$$(\mathrm{A} - \boldsymbol{\lambda I})\mathbf{X} = \mathbf{0}$$

Solving

$$\det(\mathrm{A} - \boldsymbol{\lambda I}) = 0$$

- A 2x2 matrix example

$$A = \begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix}$$

- We solve

$$\det(\begin{bmatrix} 25 - \lambda & 20 \\ 20 & 25 - \lambda \end{bmatrix}) = 0$$

$$(25 - \lambda)(25 - \lambda) - 400 = 0$$

Finally

λ=45 or 5

How to find the eigenvalue?

- Substitute $\lambda = 45$ into

$$(\mathrm{A} - \boldsymbol{\lambda I})\mathbf{X} = \mathbf{0}$$

$$\begin{bmatrix} 25-45 & 20 \\ 20 & 25-45 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -20 & 20 \\ 20 & -20 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

normalize to

$$X_{\lambda=45} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Similarity, we have

Eigen-Vector correspond to $\sqrt{5}$

$$\begin{bmatrix} 25-5 & 20 \\ 20 & 25-5 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$X_{\lambda=5} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

You can also calculate by Matlab

Eigenvalue and Eigenvector in MATLAB

- [V, D]=eig(A)

Eigenvalues of Matrix

Use `gallery` to create a symmetric positive definite matrix.

```
A=[25 20;20 25]
```

```
A = 2×2
    25    20
    20    25
```

Calculate the eigenvalues of `A`. The result is a column vector.

```
[V,D] = eig(A)
```

```
V = 2×2
   -0.7071    0.7071
    0.7071    0.7071

D = 2×2
     5     0
     0    45
```

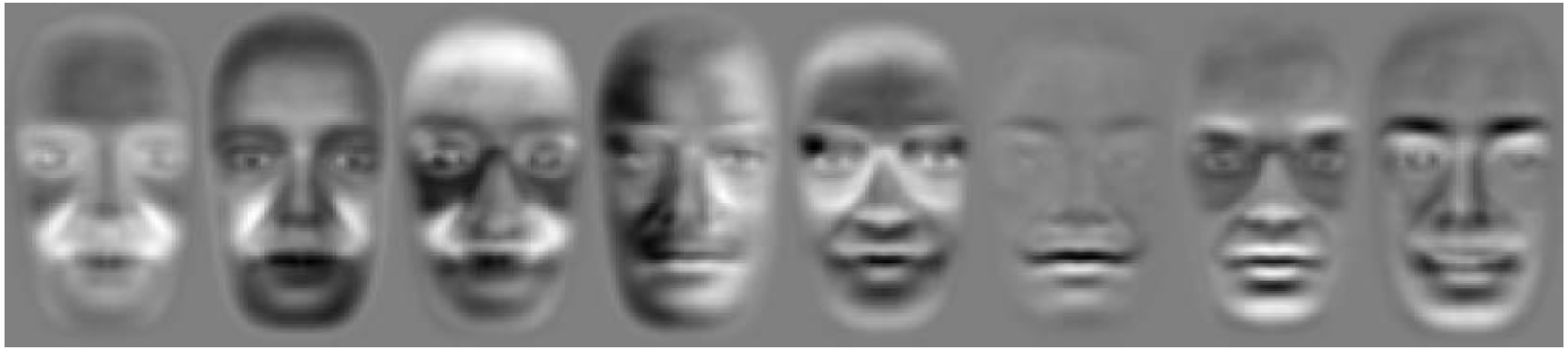
Alternatively, use `eigvalOption` to return the eigenvalues in a diagonal matrix.

```
D = eig(A,'matrix')
```

```
D = 2×2
     5     0
     0    45
```

Use of Eigenvalue/Eigenvector

- Google's PageRank (https://www.intmath.com/matrices-determinants/8-applications-eigenvalues-eigenvectors.php)
- Interest Point Detection (Harris Corner)
- Face Recognition - Eigenfaces



Singular Value Decomposition (SVD)

[U,S,V] = svd(A)

Singular Value Decomposition (SVD)

- We are interested in analyzing a large data set $\mathrm{A} \in \mathbb{C}^{m \times n}$:

$$\mathrm{A} = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & | & | \end{bmatrix}$$

- The columns $\mathrm{a}_k \in \mathbb{C}^n$ may be measurements from simulations or experiments. The index k is the k^{th} distinct set of measurements.

Singular Value Decomposition (SVD)

- The SVD is a unique matrix decomposition that exists for every complex-valued matrix $\mathrm{A} \in \mathbb{C}^{m \times n}$

$$\mathrm{A} = \mathrm{U\Sigma V^T}$$

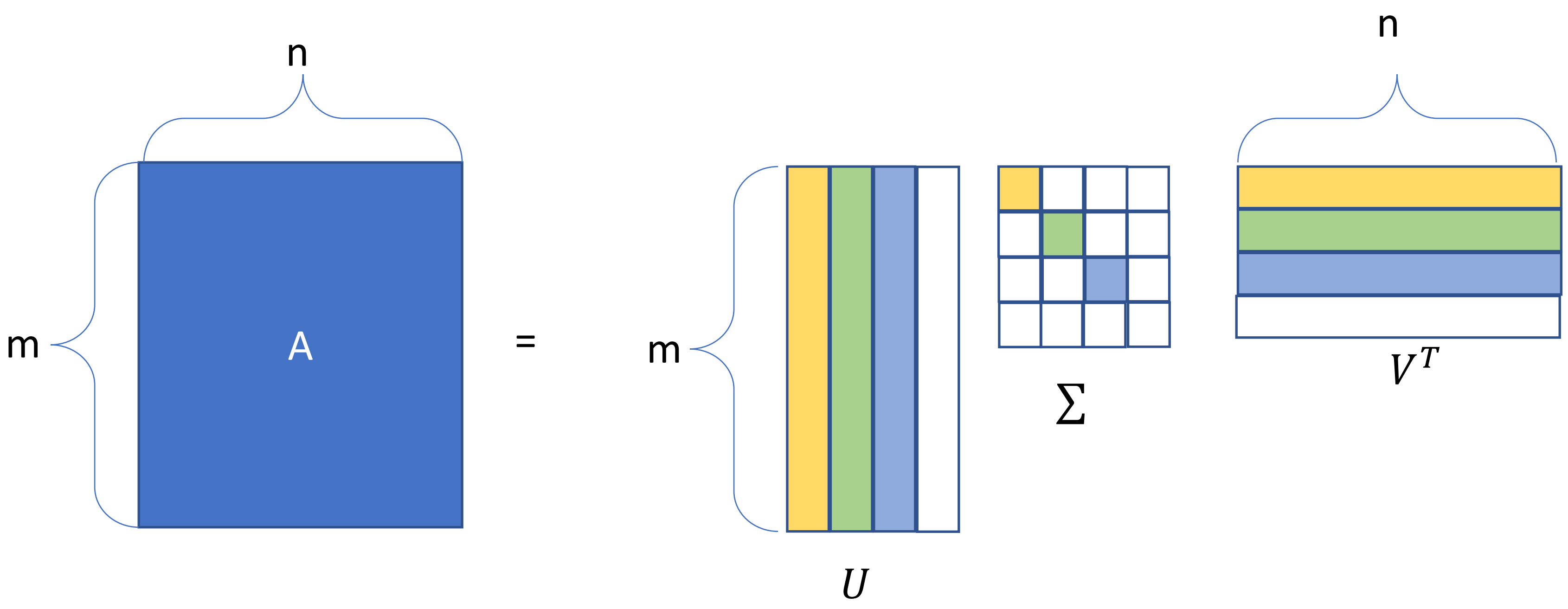
- The matrix can be expressed as:
-

$$\mathrm{A} = \mathrm{U\Sigma V^T} = \begin{bmatrix} | & | & | & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & | & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & \overline{} & \\ & 0 & \end{bmatrix} [v_1 , v_2 \ \dots v_n]$$

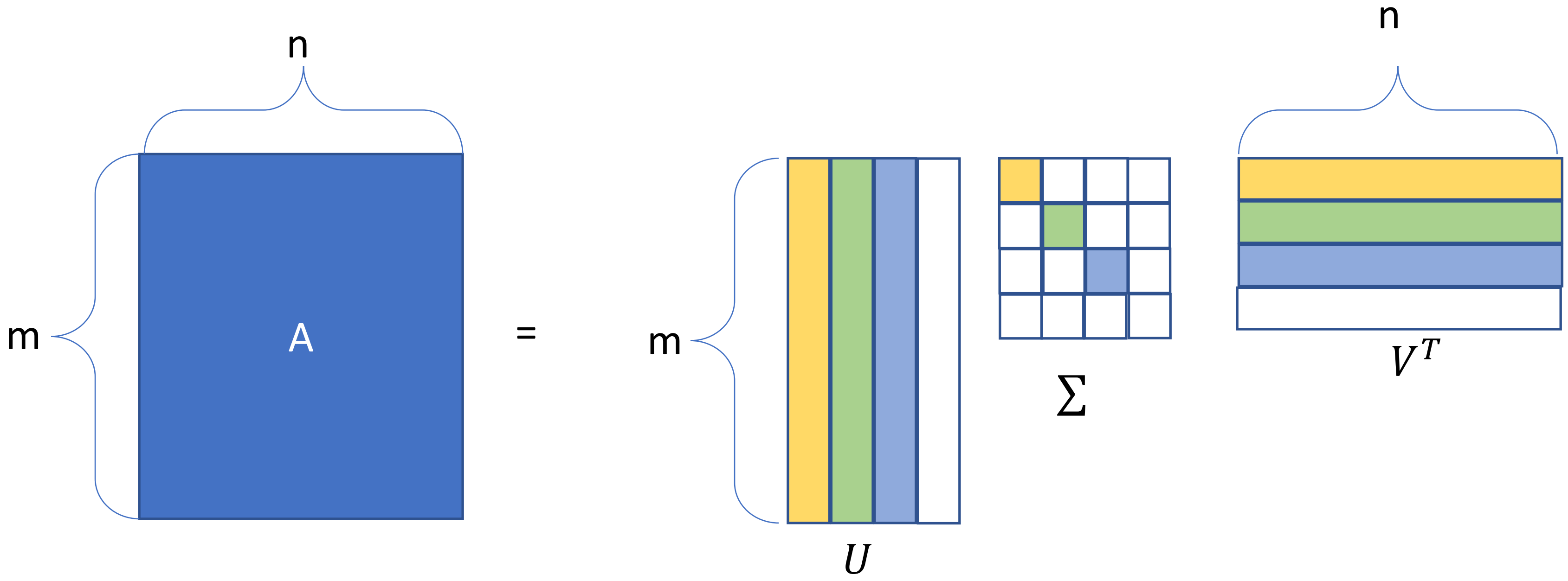
Singular Value Decomposition (SVD)

- Given an $m \times n$ matrix A, it can be decomposed into

$$A_{[m\times n]} = U_{[m\times k]}\, \Sigma_{[k\times k]} {V^T}_{[k\times n]}$$

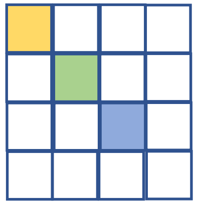


Properties of **U** and **V**



- U_i and V_i are singular vectors
  - Every column is *orthonormal*
  - $U_i \cdot U_i^T = 1$ $U_i \cdot U_j^T = 0$ for $i \neq j$ $U^T U = I$
  - $V_i \cdot V_i^T = 1 = 1$ $V_i \cdot V_j^T = 0$ for $i \neq j$ $V^T V = I$

Singular Value Decomposition (SVD)



Σ

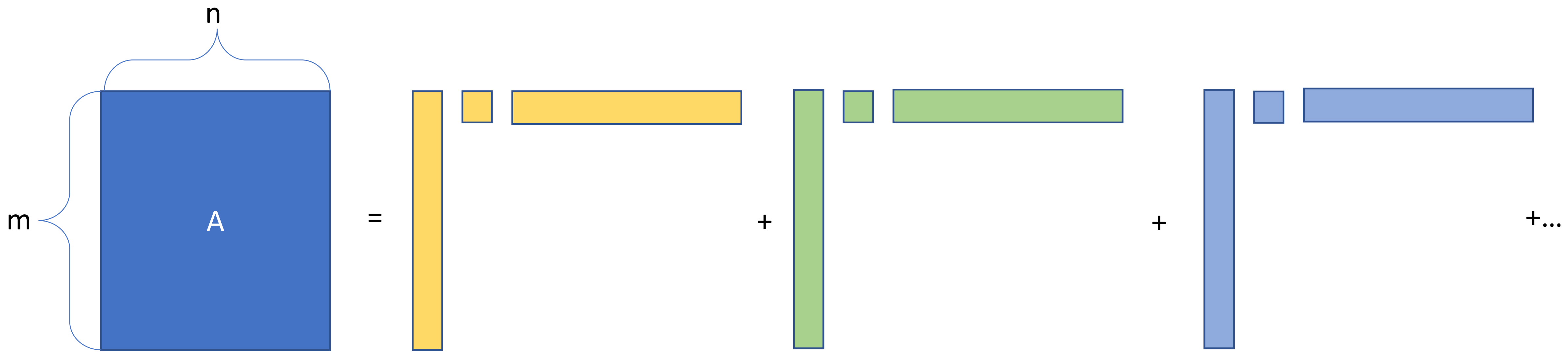
$= diag\{\sigma_1, \sigma_2, \dots, \sigma_r\}$

Where $\sigma_1 \geq \sigma_2 \geq \sigma_3 \dots \geq \sigma_r \geq 0$

- $\sum$ is singular values
  - Diagonal matrix with non-negative real numbers $\sigma_1 \dots \sigma_k$ on ***diagonal*** where $\sigma_1 > \sigma_2 > \cdots > \sigma_k$

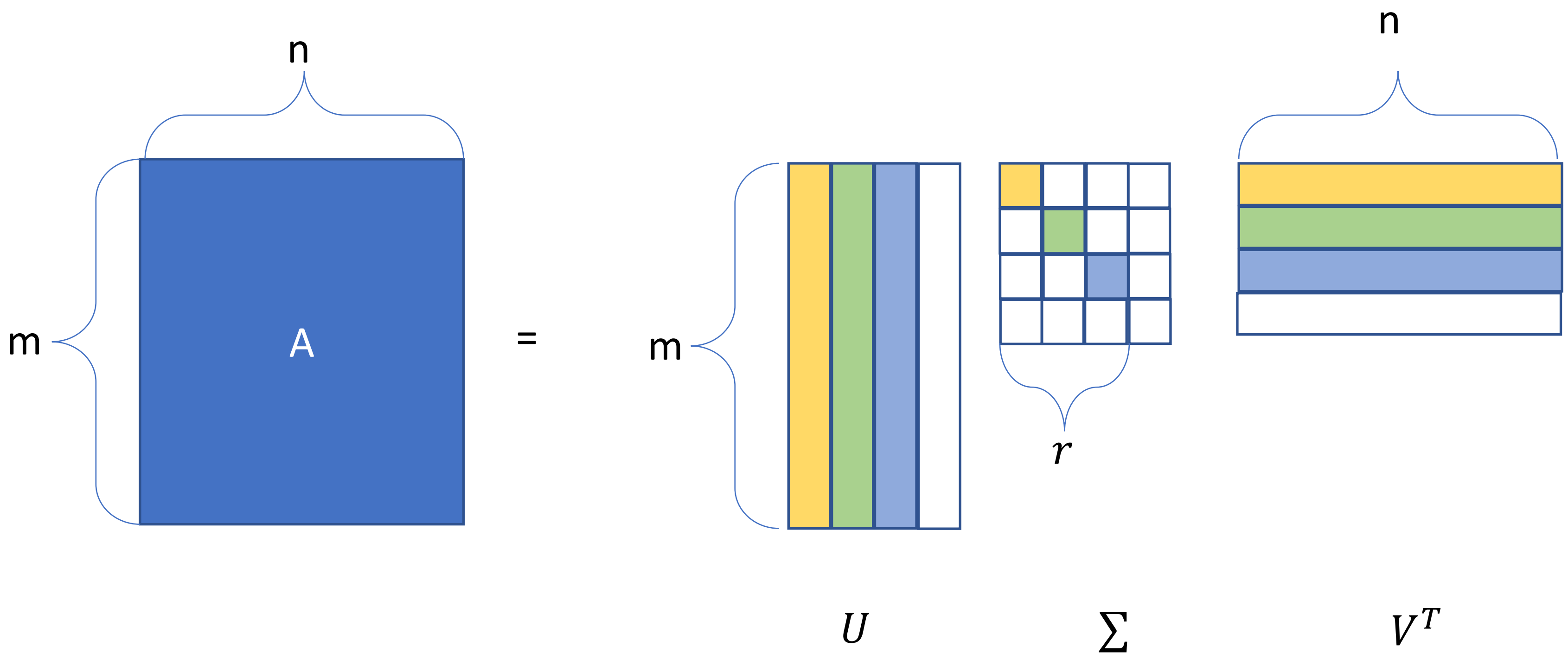
Singular Value Decomposition (SVD)

- A$= U \sum V^T = u_1\sigma_1 v_1^{\mathrm{T}} + \cdots + u_n\sigma_n v_n^{\mathrm{T}}$

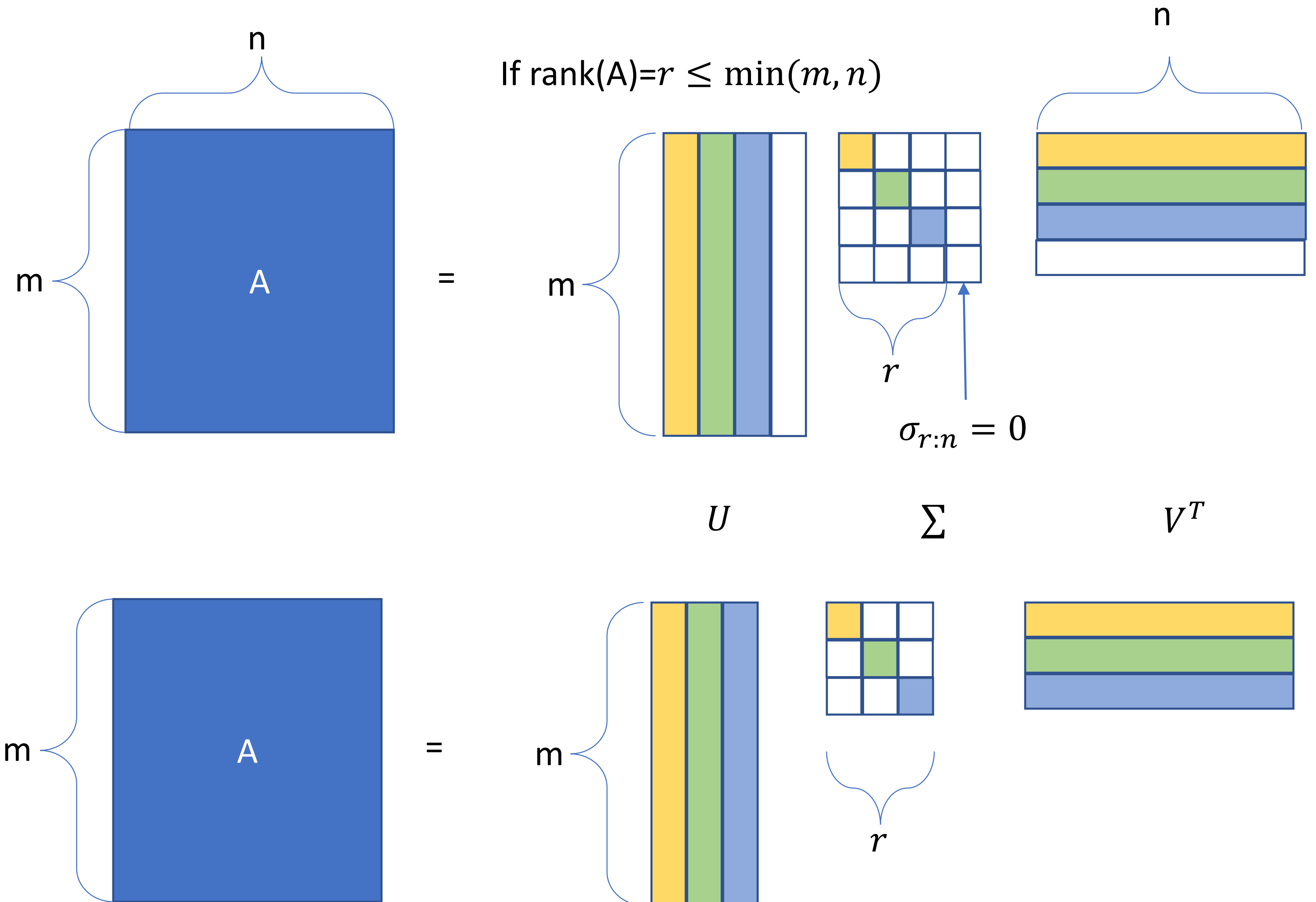


Rank of A

- rank(A)=$r \leq \min(m, n)$

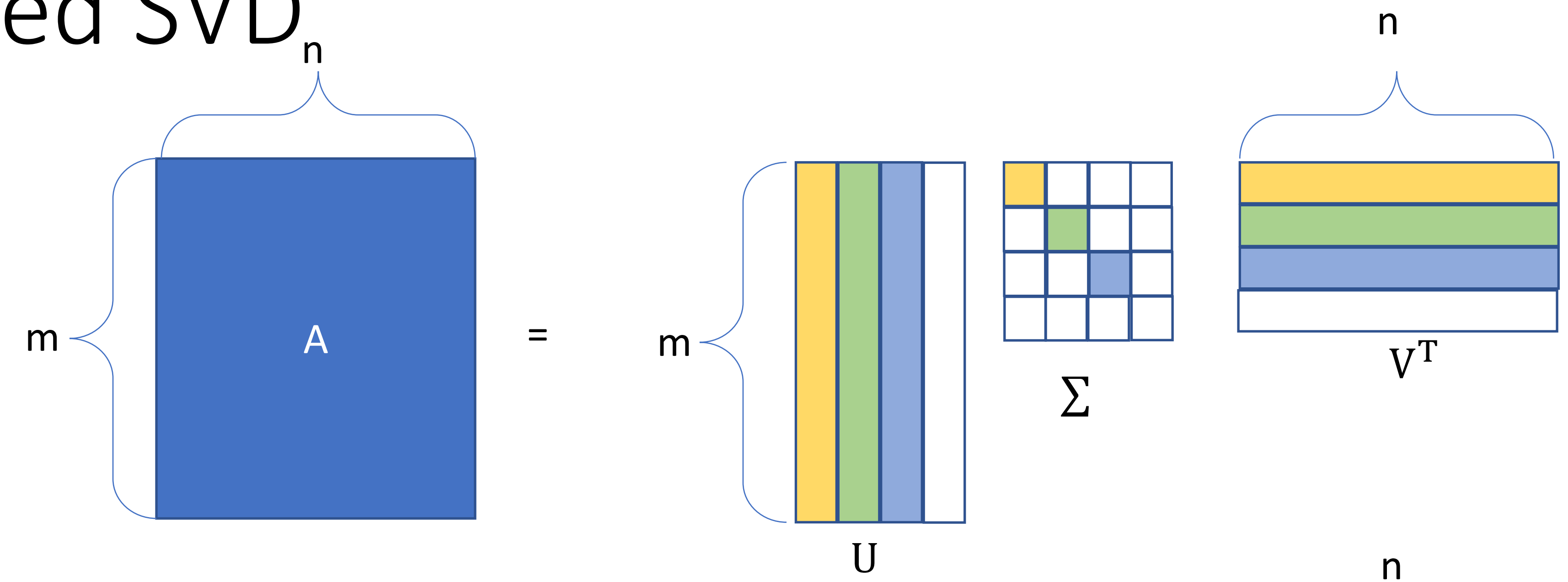


Rank of A

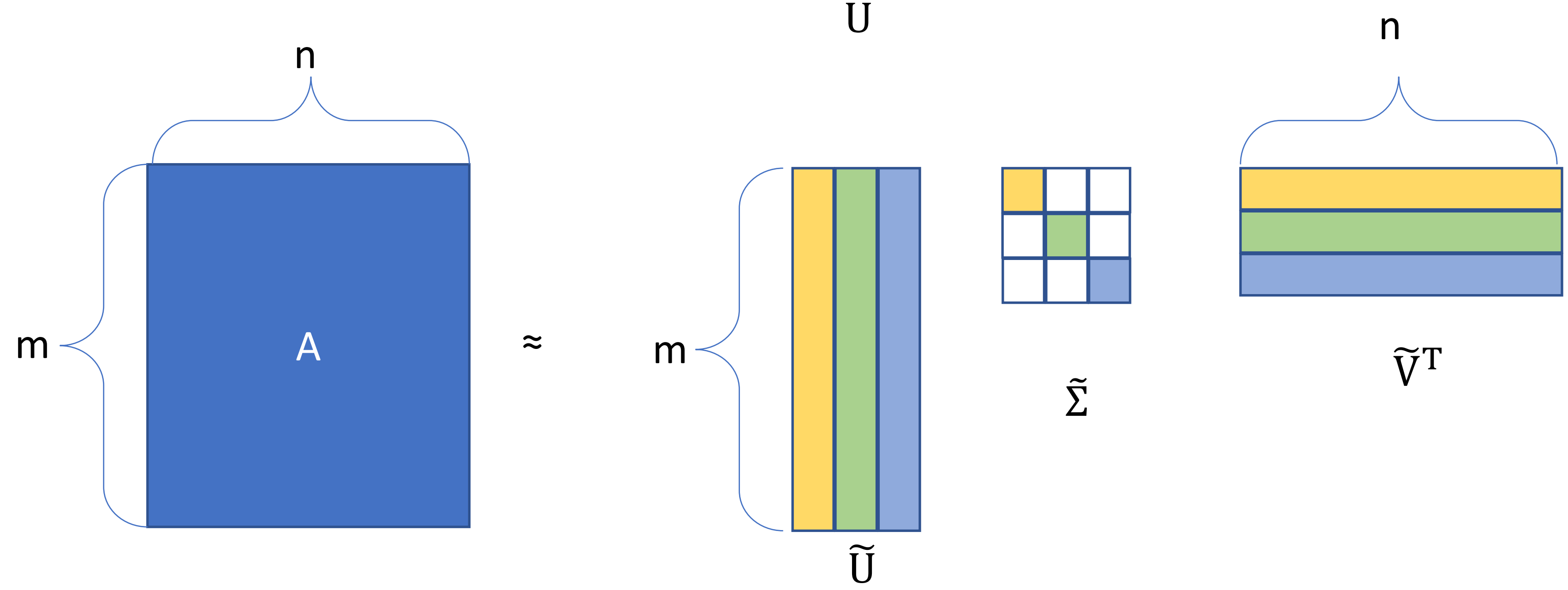


The truncated SVD

- Full SVD

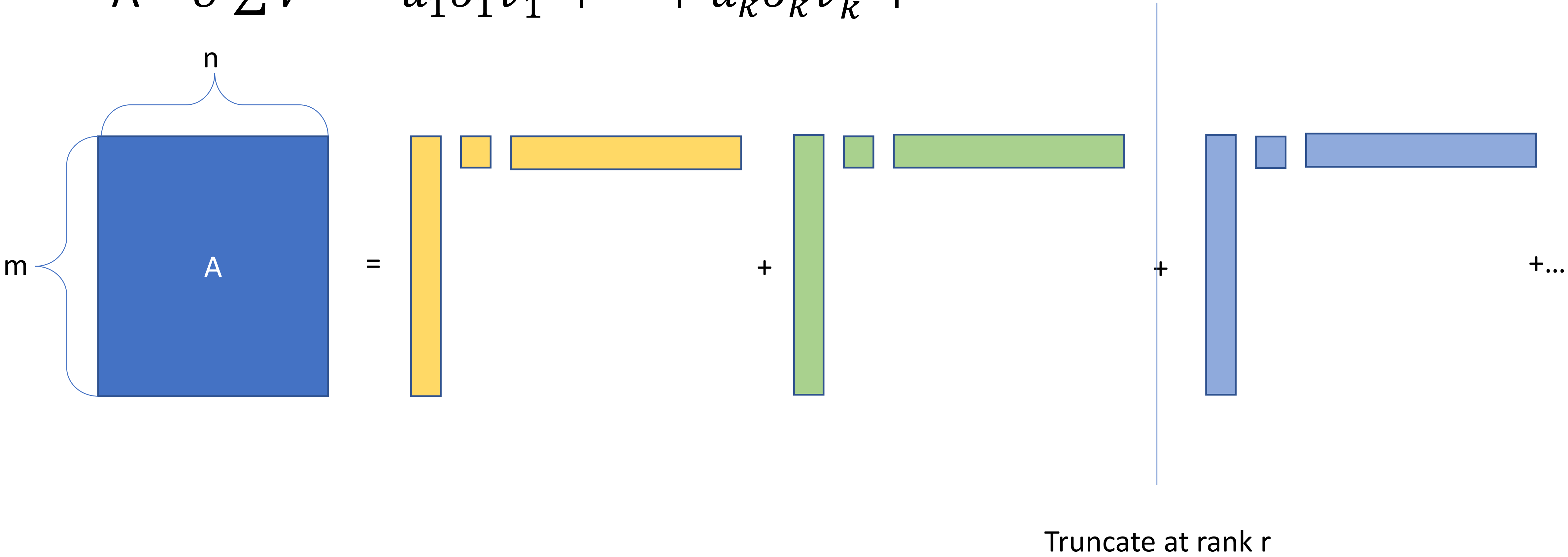


- Economy SVD



SVD as Matrix Approximation

- A$= U \sum V^T = u_1 \sigma_1 v_1^{\mathrm{T}} + \cdots + u_k \sigma_k v_k^{\mathrm{T}} + \cdots$



Application for Image Compression

- MATLAB Example

```
A=imread('../DATA/dog.jpg');
X=double(rgb2gray(A)); % Convert RBG->gray, 256 bit->double.
nx = size(X,1); ny = size(X,2);
imagesc(X), axis off, colormap gray
```

- Take SVD

```
[U,S,V] = svd(X);
```

Source: SL. Brunton, “Date Driven Science & Engineering
Machine Learning, Dynamical Systems, and Control,

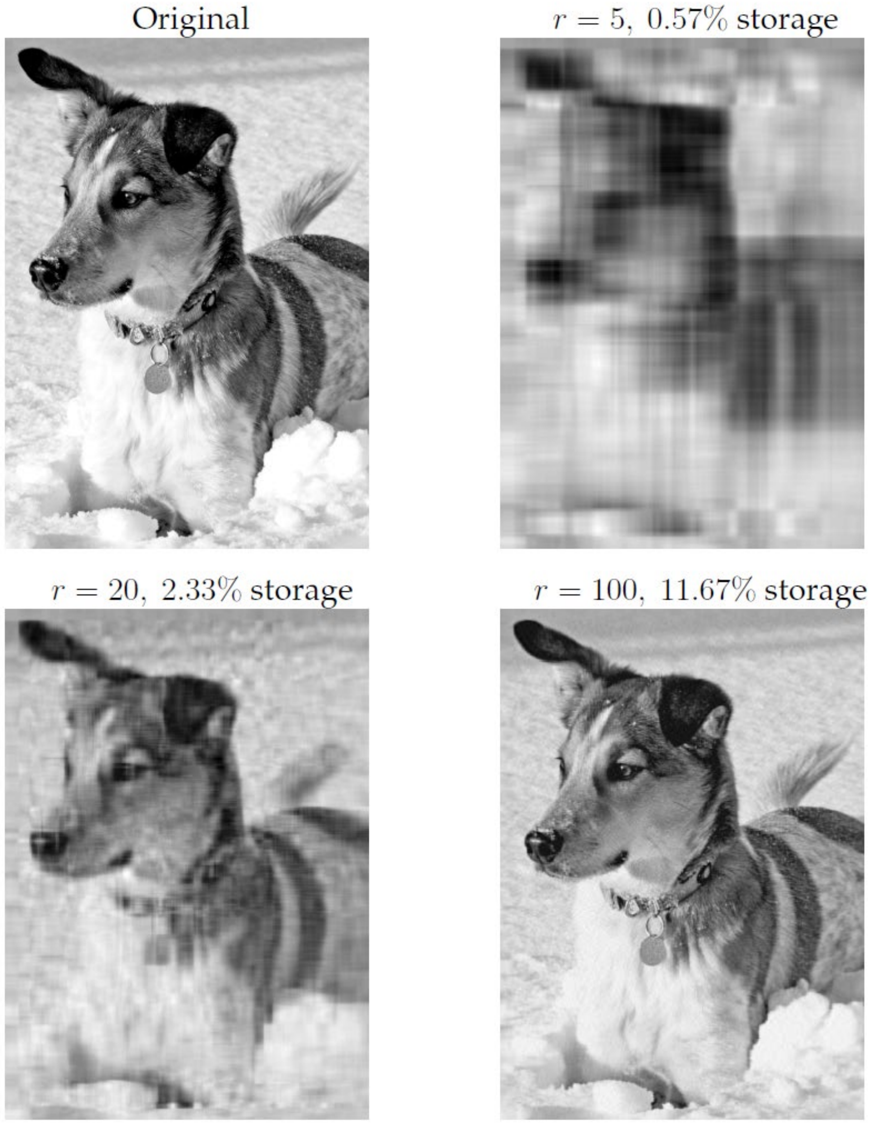
Image Compression

- Next compute the approximate matrix using truncated SVD for rank (r=5, 20 and 100)

```
for r=[5 20 100];  % Truncation value
    Xapprox = U(:,1:r)*S(1:r,1:r)*V(:,1:r)'; % Approx. image
    figure, imagesc(Xapprox), axis off
    title(['r=',num2str(r,'%d'),']);
end
```

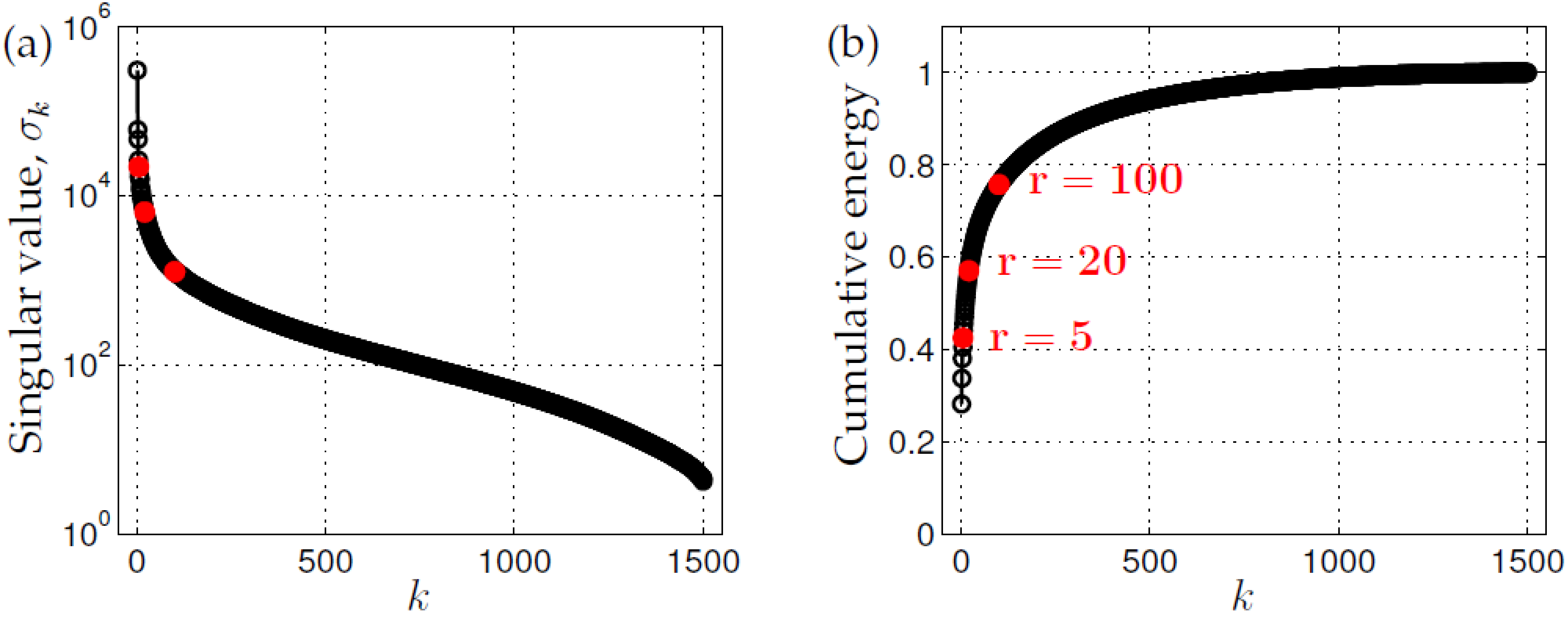
Source: SL. Brunton, “Date Driven Science & Engineering Machine Learning, Dynamical Systems, and Control,

Image Compression Result



Source: SL. Brunton, “Date Driven Science & Engineering Machine Learning, Dynamical Systems, and Control,

Study on the singular value and cumulative energy




```
subplot(1,2,1), semilogy(diag(S),'k')
subplot(1,2,2), plot(cumsum(diag(S))/sum(diag(S)),'k')
```

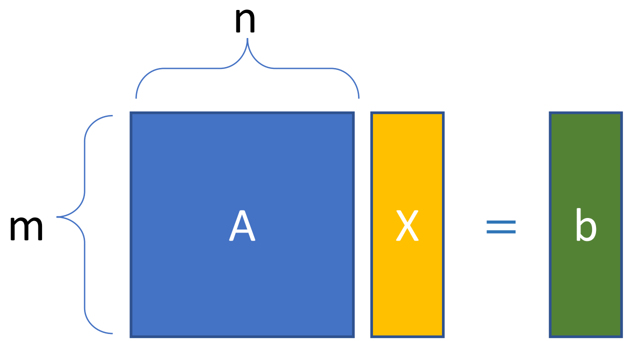
Source: SL. Brunton, “Date Driven Science & Engineering Machine Learning, Dynamical Systems, and Control,

Solving Linear System of equations with SVD

- Many systems can be represented as a linear system of equations

$$\mathrm{Ax} = \mathrm{b}$$

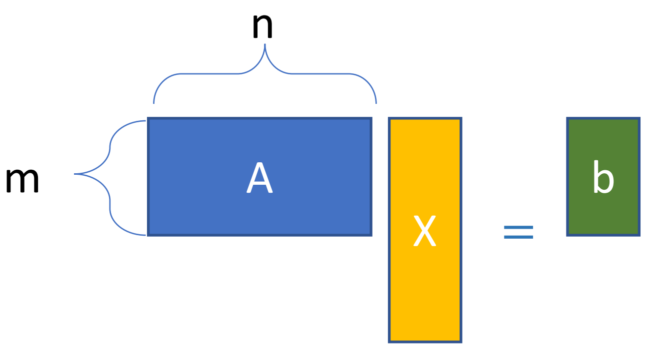
- Both constraint matrix A and vector b are known, we need to solve for vector x



- SVD allows us to generalized into to non-square matrix $A \in \mathbb{C}^{m \times n}$ where $m \neq n$

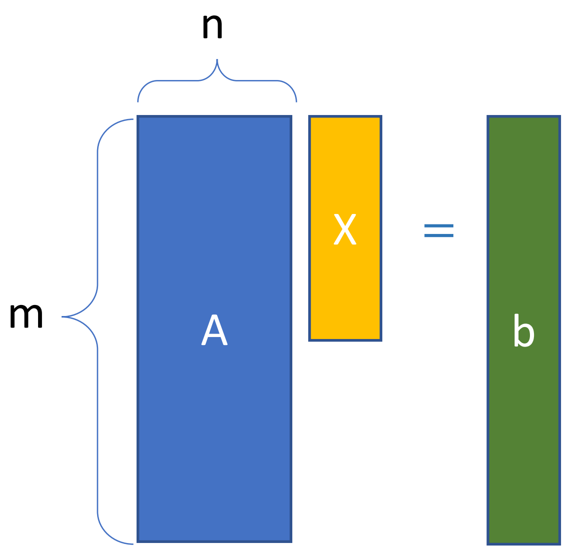
Two cases

- Underdetermined case $m < n$
- Short-fat matrix A



- Infinite-many solutions of X

- Overdetermined case $n < m$
- Tall-skinny matrix A



- No solutions.

Least Square Solution

- In the overdetermined case when no solution exists, we would like to find the solution x that minimize the ***sum-squared error***

$$\min||\mathrm{Ax} - \mathrm{b}||_2^2$$

- This is the so-called ***least-squares*** solutions and SVD is the technique for these important optimization problems.

Least Square Solution

- By SVD, we have

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

- The truncated SVD

$$A = \tilde{U}\tilde{\Sigma}\tilde{V}^T$$

- Let A^+ be the pseudo-inverse of A

$$A^+ = \tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^T \Rightarrow A^+A = I_{m\times m}$$

- Then

$$Ax = b$$
$$A^+A\tilde{x} = A^+b$$
$$\tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^T\tilde{U}\tilde{\Sigma}\tilde{V}^T = \tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^Tb$$
$$\tilde{x} = \tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^Tb$$

Advantages of SVD on least square solution

- By truncated ***SVD***, we have

$$\tilde{x} = \tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^{T}b = A^{+}b$$

- Computing the ***pseudo-inverse*** A^{+} is computationally *efficient*, after the expensive up-front cost of computing the SVD

- Inverting the matrix $\tilde{U}$ and $\tilde{V}$ involves matrix multiplication by the transpose matrices.

- Inverting the diagonal matrix $\tilde{\Sigma}$ is even more efficient.

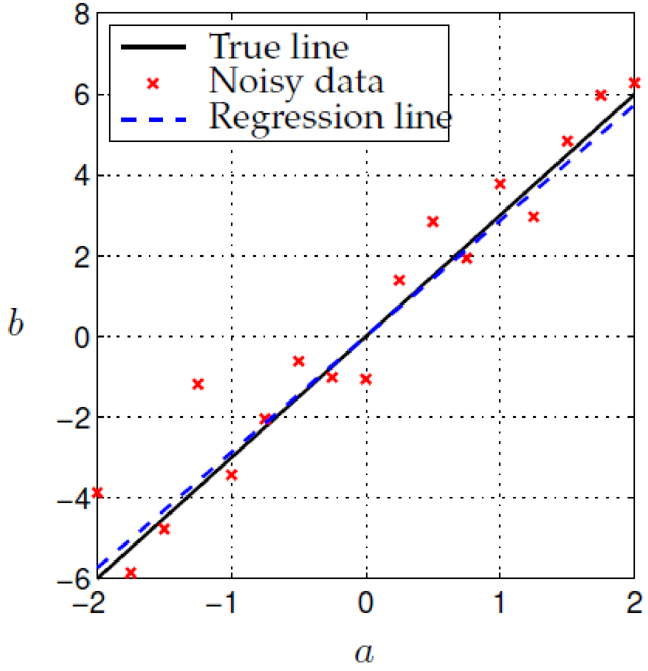
One-dimensional linear regression

- ***Regression*** is an important statistical tool to relate variables to one another based on data.

- Red x's are obtained by adding Gaussian noise

- Assume the data is linearly related.

- We can use pseudo-inverse to find the least-square solution for the slope x



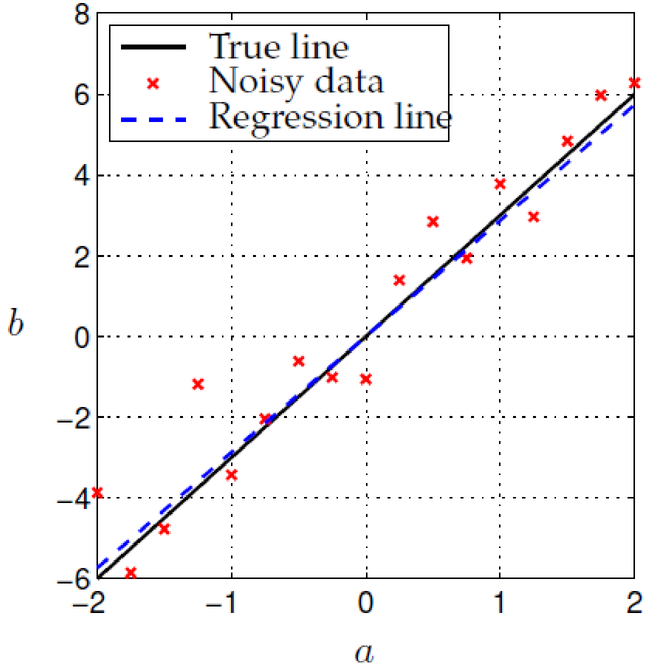
Source: SL. Brunton, "Date Driven Science & Engineering Machine Learning, Dynamical Systems, and Control,

One-dimensional linear regression

- Let

$$\begin{bmatrix} | \\ \mathrm{b} \\ | \end{bmatrix} = \begin{bmatrix} | \\ \mathrm{a} \\ | \end{bmatrix} x = \tilde{\mathrm{U}}\tilde{\Sigma}\tilde{\mathrm{V}}^{\mathrm{T}} x$$

$$\Rightarrow x = \tilde{\mathrm{V}}\tilde{\Sigma}^{-1}\tilde{\mathrm{U}}^{\mathrm{T}}\mathrm{b}$$



Source: SL. Brunton, “Date Driven Science & Engineering Machine Learning, Dynamical Systems, and Control,

MATLAB Code

- Plot on MATLAB

```
x = 3;                          % True slope
a = [-2:.25:2]';
b = a*x + 1*randn(size(a));     % Add noise
plot(a,x*a,'k')                 % True relationship
hold on, plot(a,b,'rx')         % Noisy measurements
```

- Build-in functions in MATLAB

```
xtilde1 = V*inv(S)*U'*b
xtilde2 = pinv(a)*b
xtilde3 = regress(b,a)
```

Source: SL. Brunton, “Date Driven Science & Engineering Machine Learning, Dynamical Systems, and Control,

Least Square Fit (Linear homogeneous Equations)

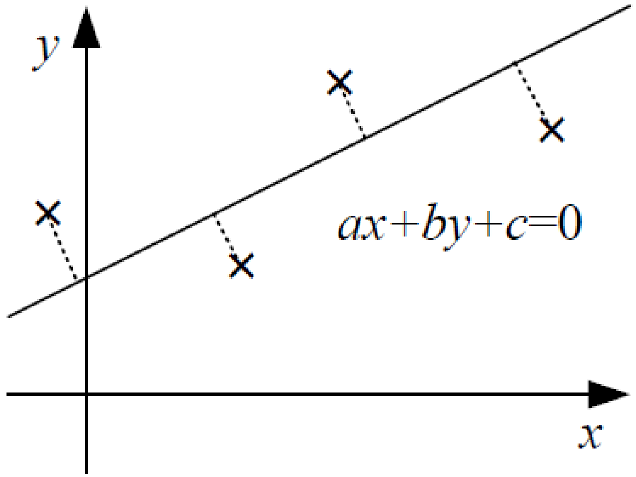
- Suppose we have a system of equations

$$Ax = 0$$

- Matrix $A \in \mathbb{C}^{m \times n}$ is known and we need to solve x
- As $Ax \neq 0$, we aim to

$$\min_{\mathbf{x}} ||A\mathbf{x}||^2$$

- One trivial solution $\mathbf{x}$ =0, but we are not interested.
  $||\mathbf{x}||$ =1



Least Square Fit (Linear homogeneous Equations)

- We have

$$\min_{\mathbf{x}} \|A\mathbf{x}\|^2 \quad \text{and} \quad \|\mathbf{x}\| = \mathbf{1}$$

- Apply SVD on $A = U\Sigma V^T$, we have

$$\min_{\mathbf{x}} \|A\mathbf{x}\|^2 = \min_{\mathbf{x}} \left\|U\Sigma V^T\mathbf{x}\right\|^2$$

- Since U and V are **orthonormal**

$$\left\|U\Sigma V^T\mathbf{x}\right\|^2 = \left\|\Sigma V^T\mathbf{x}\right\|^2 \quad \text{and} \quad \left\|\Sigma V^T\right\| = \|\Sigma\|$$

Least Square Fit (Linear homogeneous Equations)

- Let $\mathbf{y} = \mathrm{V}^{\mathrm{T}}\mathbf{x}$, we have

$$\min_{\mathbf{x}} \left\| \Sigma \mathrm{V}^{\mathrm{T}} \mathbf{x} \right\|^2 = \min_{\mathbf{y}} \| \Sigma \mathbf{y} \|^2 \text{ where } \|\mathbf{y}\| = \mathbf{1}$$

$$\Sigma \mathrm{y} = \begin{bmatrix} S_1 & 0 & \dots & 0 \\ \vdots & S_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & S_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{bmatrix} \text{ where } \|\mathbf{y}\| = \mathbf{1} \quad \text{Since } S_1 > S_2 > \cdots S_n, \mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{V}\mathbf{y} = [\mathrm{V}_1 | \mathrm{V}_1 | \dots | \mathrm{V}_n] \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- Finally we have $\mathbf{x} = \mathrm{V}_{\mathrm{n}}$

Reference

- Steven L. Brunton, “Data Driven Science & Engineering, Machine Learning, Dynamical Systems, and Control”, Part I