SINGULAR VALUE DECOMPOSITION (SVD)

&

PRINCIPAL COMPONENT ANALYSIS (PCA)

CS115

MỤC LỤC

01. Singular Value Decomposition

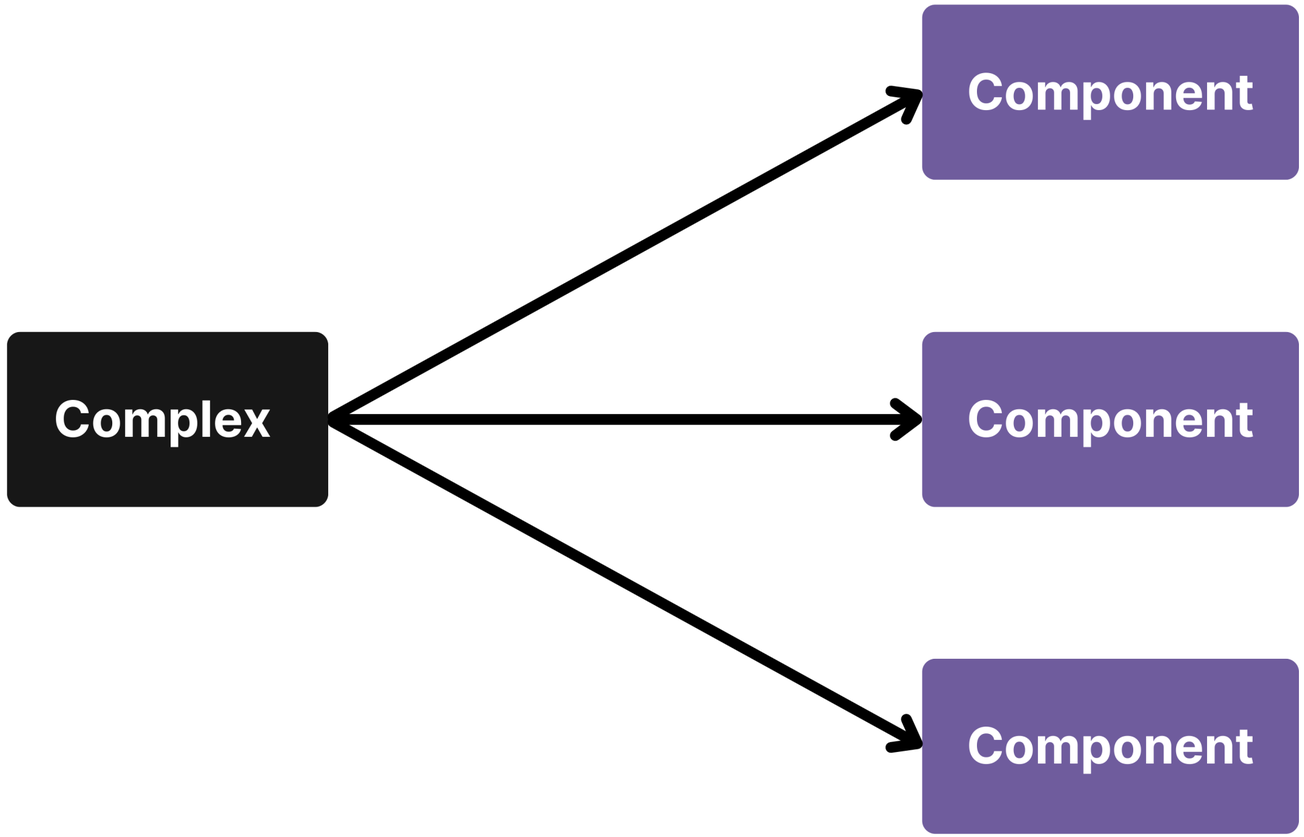
02. Principle Component Analysis

03. Implement in Python

1. Singular Value Decomposition (SVD)

1.1. What is SVD?

Idea



1.1. What is SVD?

Eigen decomposition

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

1.1. What is SVD?

Eigen decomposition

Eigen Decomposition.

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

1.1. What is SVD?

Eigen decomposition

Eigen Decomposition.

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

$$AP = PD$$

$$Ap_i = Pd_i$$

$$Ap_i = p_i d_{ii}$$

1.1. What is SVD?

Eigen decomposition

Eigen Decomposition.

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

$$AP = PD$$

$$Ap_i = Pd_i$$

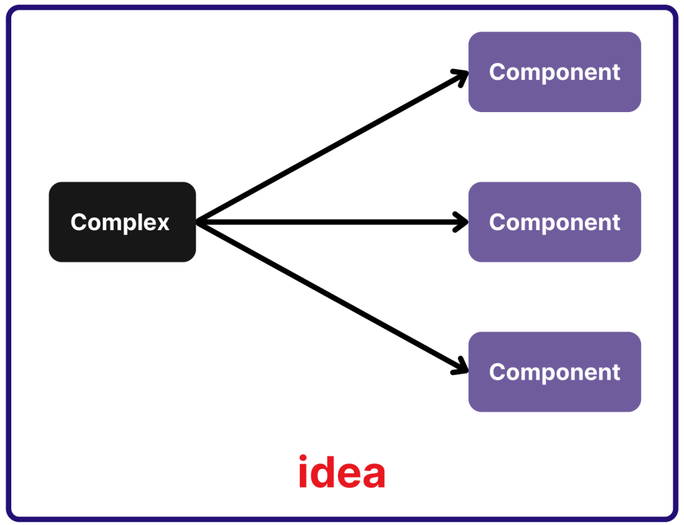
$$Ap_i = p_i d_{ii}$$

Eigen vector

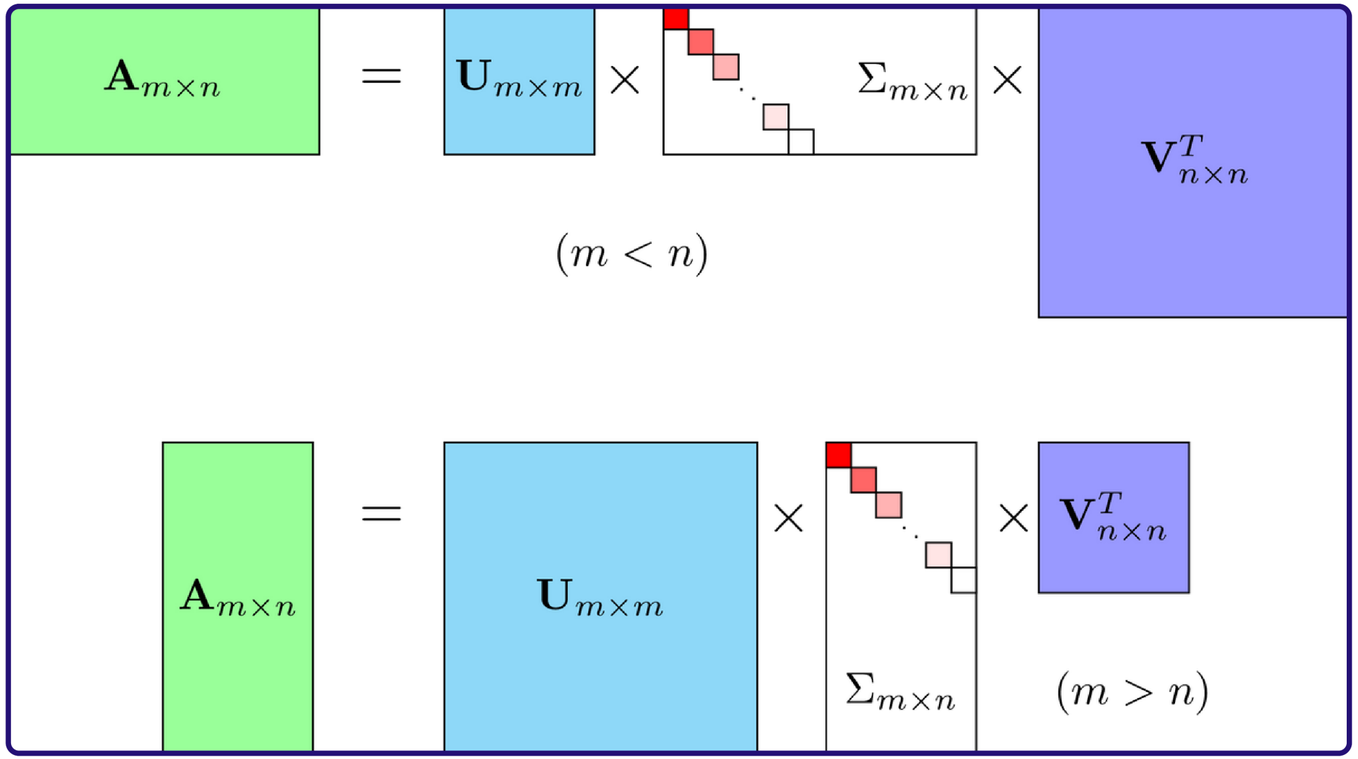
Eigen value

1.1. What is SVD?

Singular Value Composition (SVD)



$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$



1.1. What is SVD?

Singular Value Composition (SVD)

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^{-1}$$

1.1. What is SVD?

Singular Value Composition (SVD)

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

$$A = PDP^{-1}$$
$$Ap_i = p_i d_{ii}$$

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^{-1}$$

$$\begin{pmatrix} \lambda_1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & \lambda_K & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \lambda_{K+1} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \lambda_N \end{pmatrix}$$

$$\lambda_i = {\sigma_i}^2$$

$$\sigma_1^2, \sigma_2^2 \dots \sigma_m^2$$

$$\sigma_j$$

Singular Value

1.1. What is SVD?

Singular Value Composition (SVD)

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

column → left-singular vectors

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^{-1}$$

1.1. What is SVD?

Singular Value Composition (SVD)

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^{-1}$$

column

left-singular vectors

$\sigma_1^2, \sigma_2^2 \ldots \sigma_m^2$

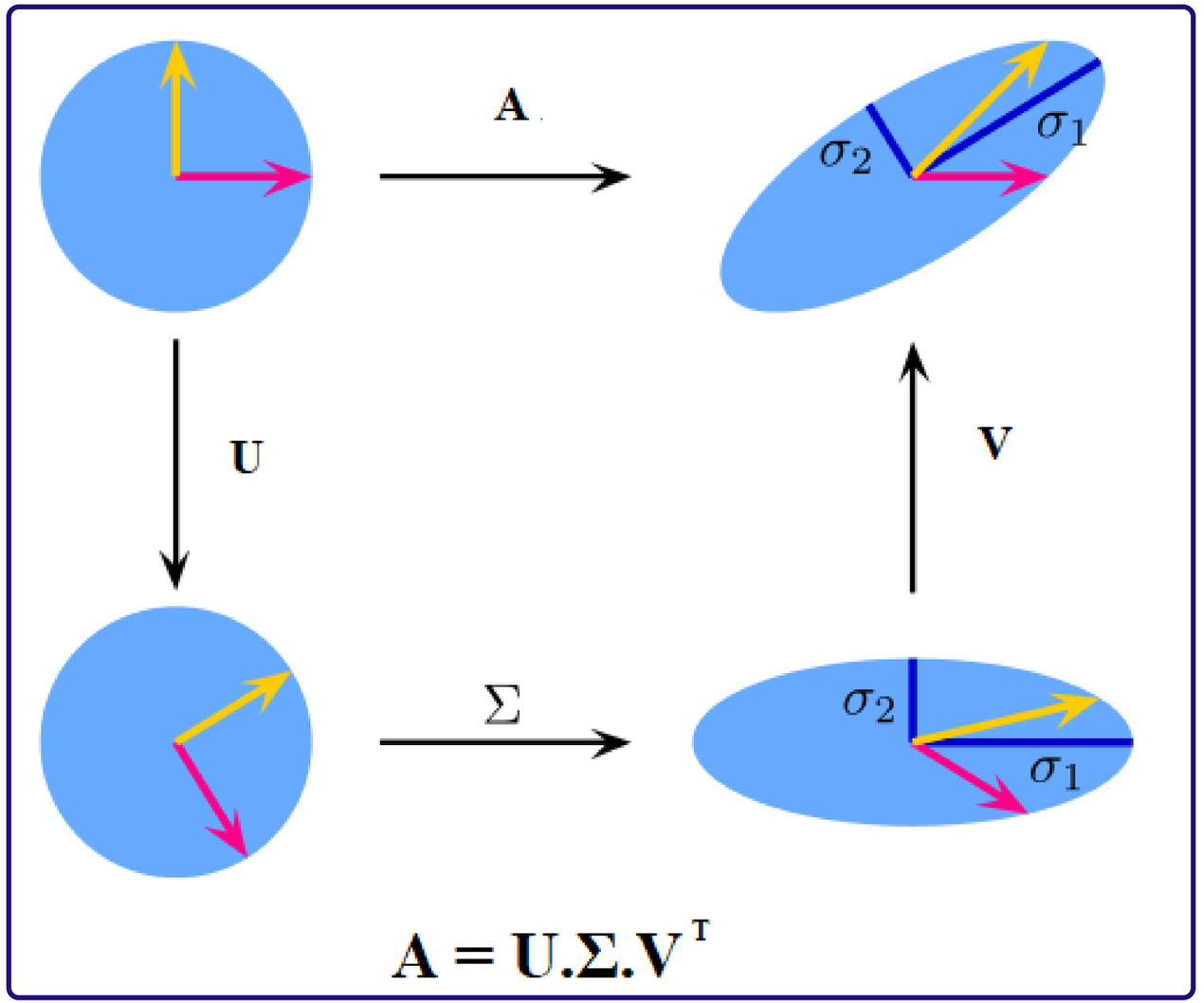
σ_j Singular Value

right-singular vectors

column

$$\mathbf{A}^T\mathbf{A} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^{-1}$$

1.1. What is SVD?



1.1. What is SVD?

Procedure

Step 1: Finding the eigenvalues

Step 2: Finding left/ right singular vectors

Step 3: Finding right/ left singular vectors

1.2. SVD - How?

Some special SVDs

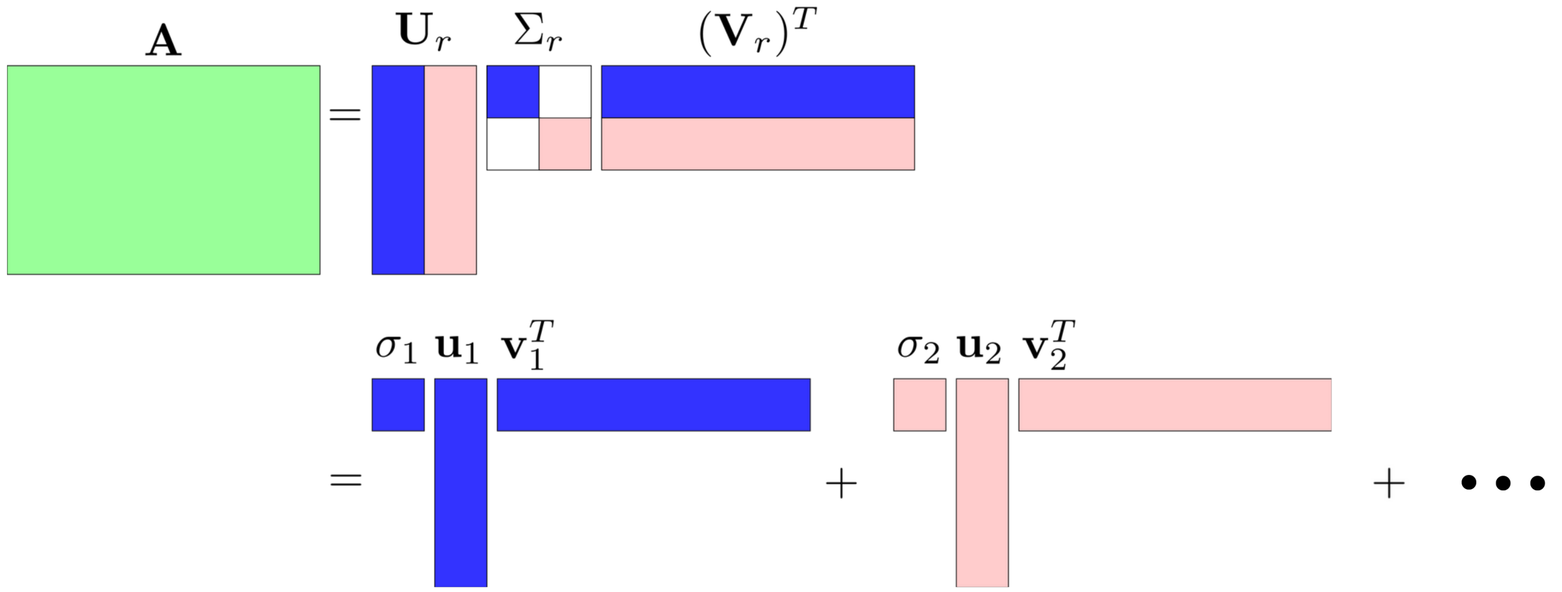
- **Compact SVD**
- **Truncate SVD**
- **Best(Low) Rank ‘*k*’ Approximation**

1.2. SVD - How?

SVDs -> Compact SVD

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

$$\mathbf{A} = \mathbf{U}_r\Sigma_r(\mathbf{V}_r)^T = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \cdots + \sigma_r\mathbf{u}_r\mathbf{v}_r^T$$

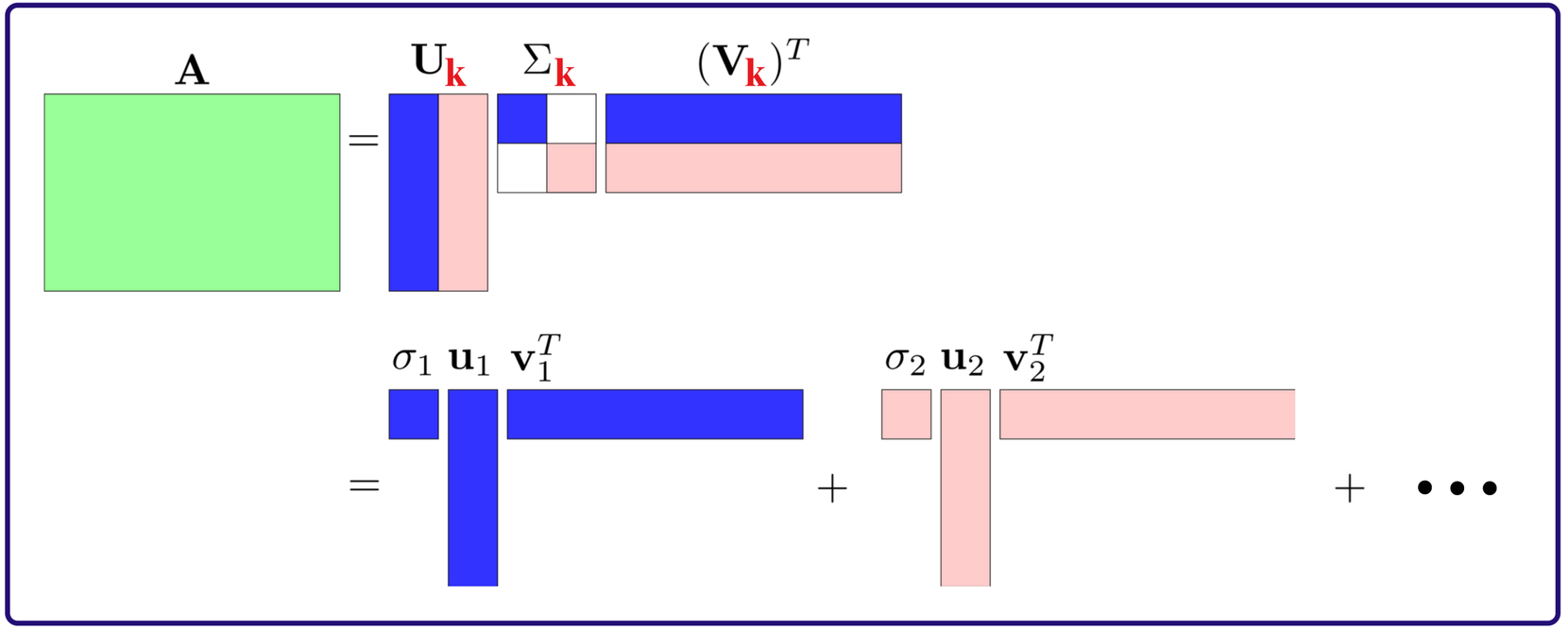


1.2. SVD - How?

SVDs -> Truncated SVD

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

$$\mathbf{A} \approx \hat{\mathbf{A}} = U_k\Sigma_k V_k^T = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \cdots + \sigma_k\mathbf{u}_k\mathbf{v}_k^T$$



1.2. SVD - How?

SVDs -> Truncated SVD

Theorem: $$||\mathbf{A} - \mathbf{A}_k||_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

The error will equal to total square of the cut-off eigenvalues in truncated SVD.

With $k = 0$, we got: $$||\mathbf{A}||_F^2 = \sum_{i=1}^{r} \sigma_i^2$$

$$\frac{||\mathbf{A} - \mathbf{A}_k||_F^2}{||\mathbf{A}||_F^2} = \frac{\sum_{i=k+1}^{r} \sigma_i^2}{\sum_{j=1}^{r} \sigma_j^2}$$

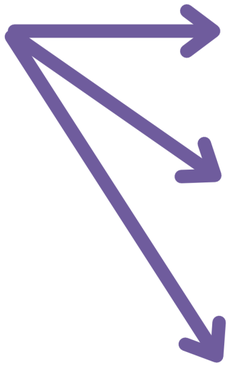
1.2. SVD - How?

SVDs -> Best(Low) Rank ‘k’ Approximation

$$\min_{\mathbf{A}} ||\mathbf{X} - \mathbf{A}||_F$$
$$\text{s.t. } \text{rank}(\mathbf{A}) = K$$

1.3. SVD - Summarise

SVD



untangle data into independent components

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

$$\mathbf{A} \approx \hat{\mathbf{A}} = U_k\Sigma_k V_k^T = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \cdots + \sigma_k\mathbf{u}_k\mathbf{v}_k^T$$

1.3. SVD - Summarise

SVD → **untangle data into independent components**

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$

$$\mathbf{A} \approx \hat{\mathbf{A}} = U_k\Sigma_k V_k^T = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \cdots + \sigma_k\mathbf{u}_k\mathbf{v}_k^T$$

Pros:

- Simplifies data
- Removes noise
- Data compression
- ...

Cons:

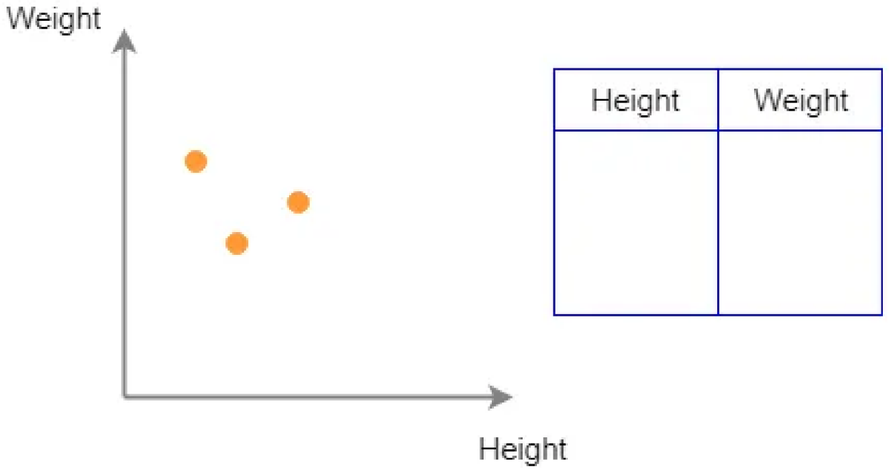
- Transformed data may be difficult to understand
- Limitations in non-linear relationships
- ...

2. Principle Component Analysis (PCA)

2.1. Why PCA?

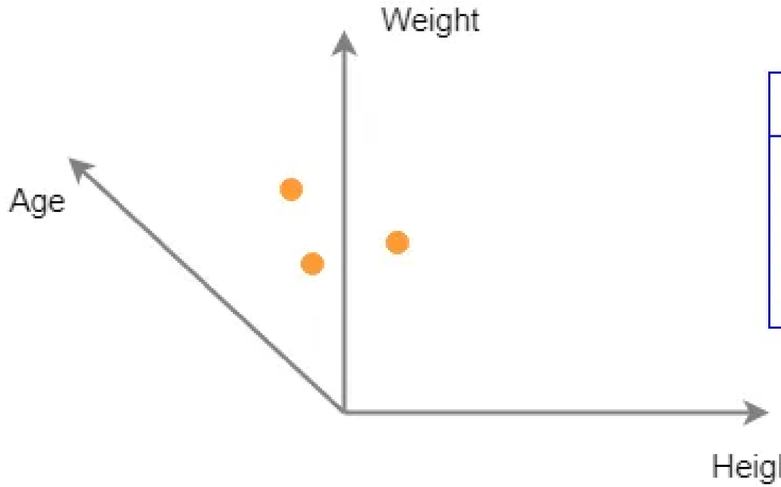
Dataset’s high dimensionality

few



Height	Weight

a lot



Height	Weight	Age

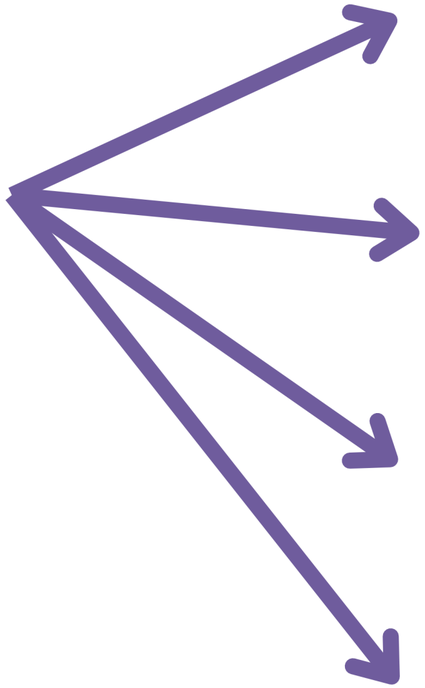
→ **a lot of variables to consider.**

→ **high-dimensional data causes challenges**

2.1. Why PCA?

Dimensionality reduction

lose some percentage but



less cost

prevent overfitting

data visualization

...

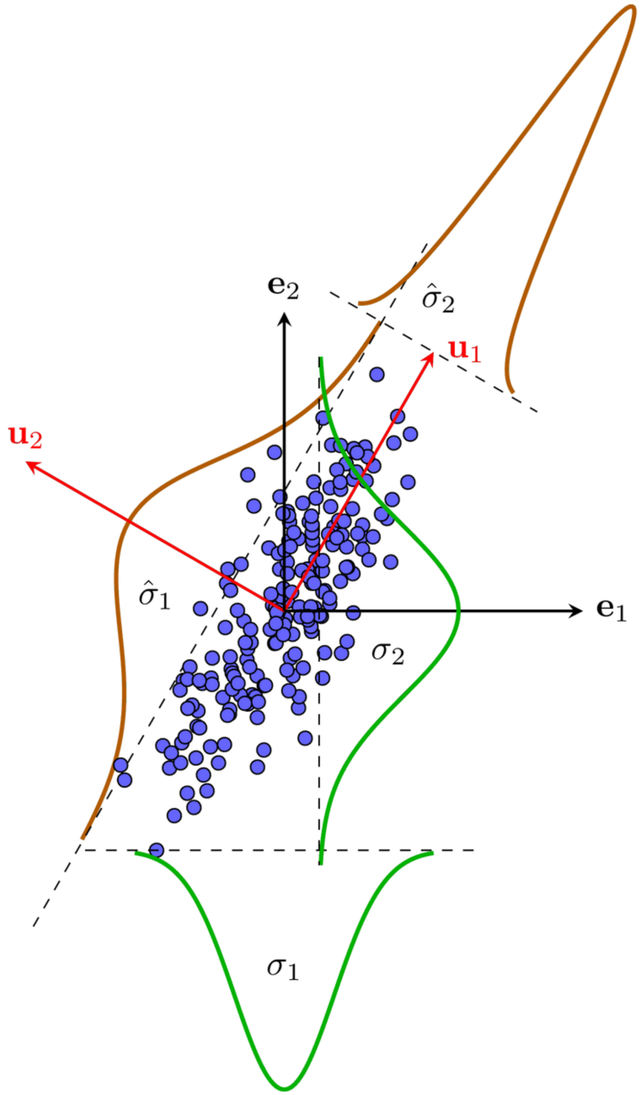
2.2. What is PCA?

Idea

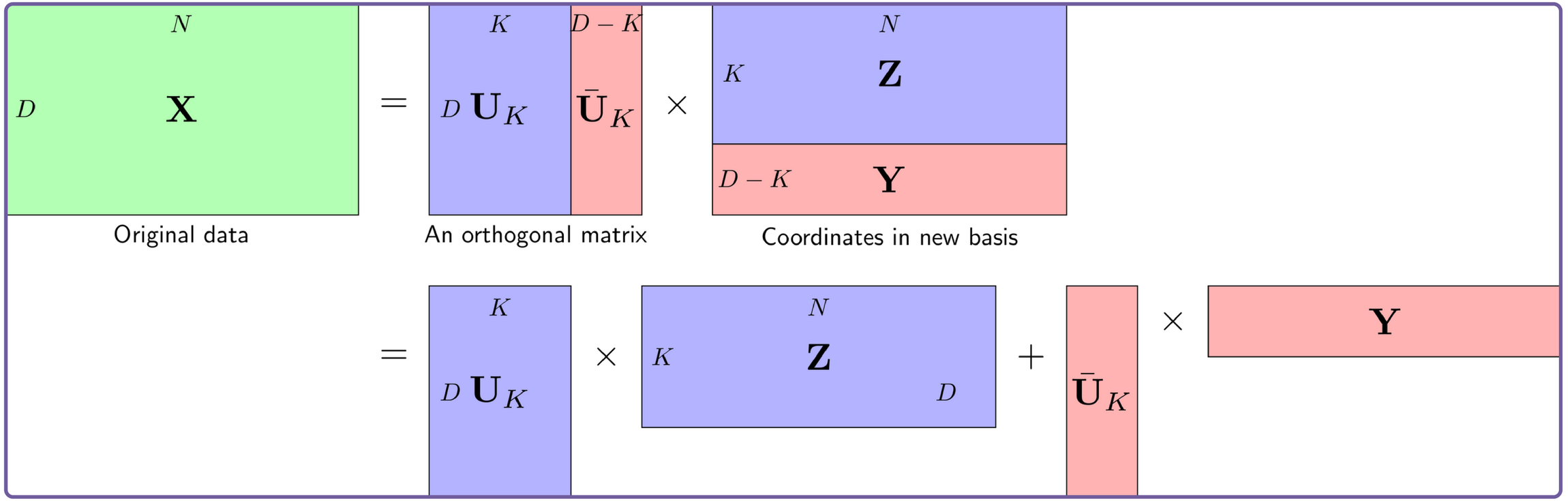
D ⟶ **K**

(K<D)

- The idea of PCA is plotting the data into a **new basis system**
- The importance of the components is **significantly different**
  -> ignoring the **least important** component



2.2. What is PCA?



$$\mathbf{X} = \mathbf{U}_K\mathbf{Z} + \bar{\mathbf{U}}_K\mathbf{Y}$$

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_K\mathbf{Z} + \bar{\mathbf{U}}_K\bar{\mathbf{U}}_K^T\bar{\mathbf{x}}\mathbf{1}^T$$

2.3. PCA - How?

Procedure

Step 1: Centering/ Standalization

Step 2: Calculate covariance matrix

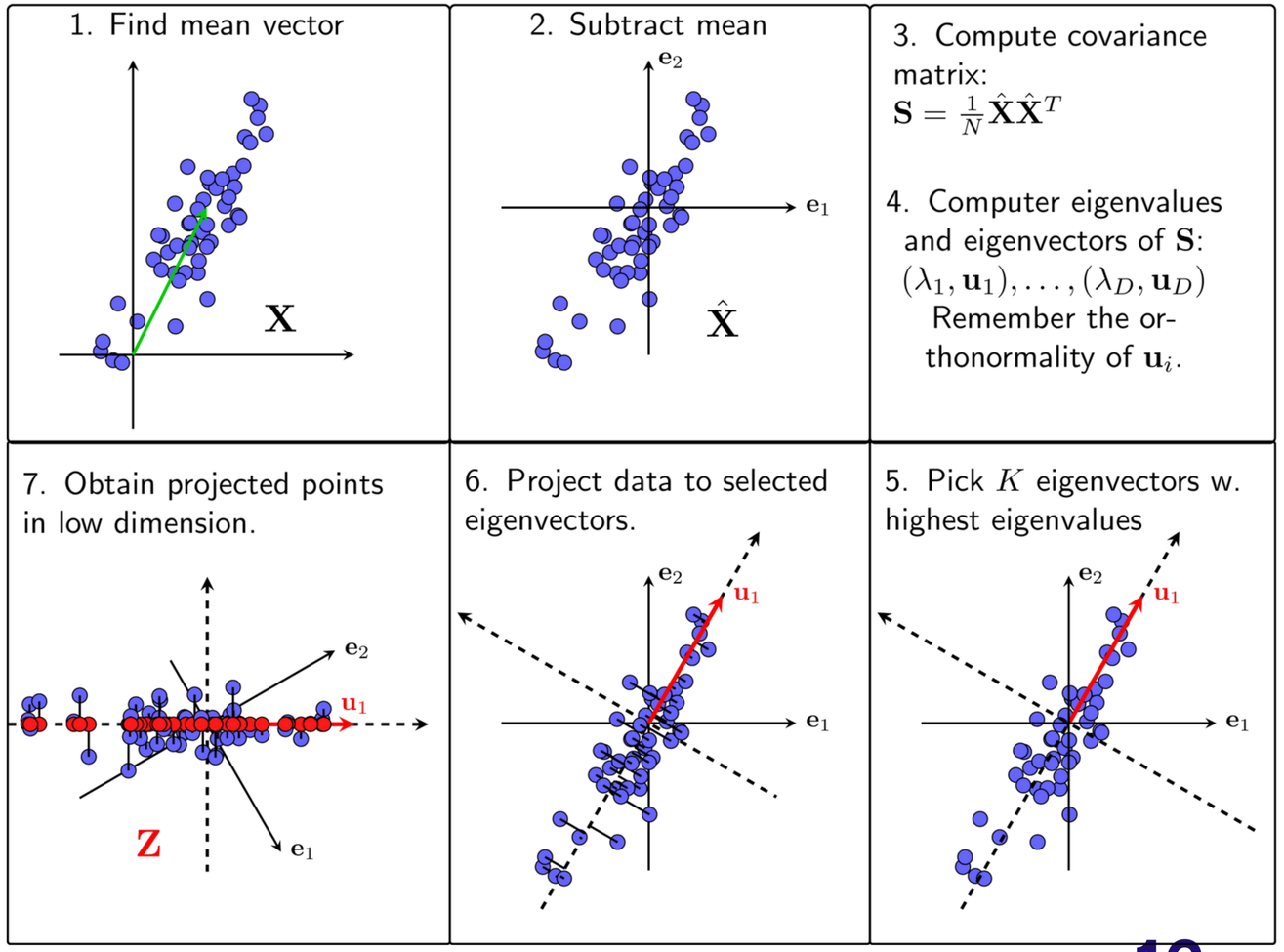
Step 3:

- Calculate eigenvalues, eigenvector
- Sort those eigenvalues in ordered descending

Step 4: Pick **K** eigenvector ***u*** corresponding to the **K** highest eigenvalues

Step 5: Project data to selected eigenvector

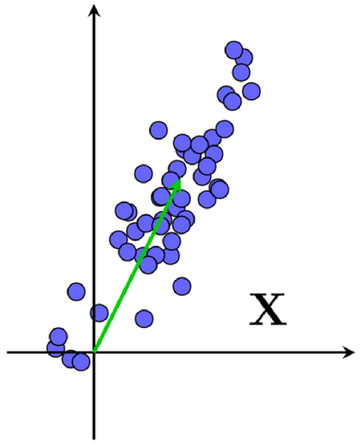
PCA procedure



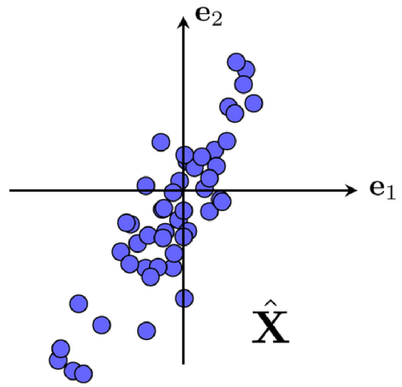
2.3. PCA - How?

Procedure

1. Find mean vector



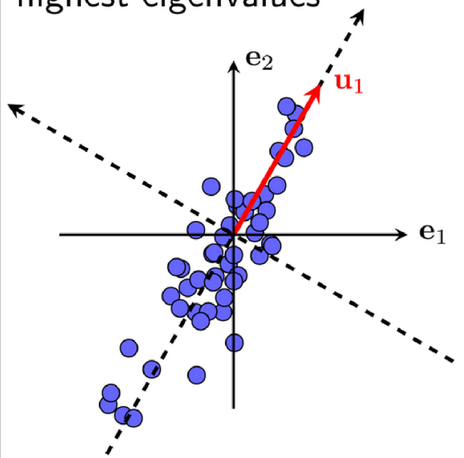
2. Subtract mean



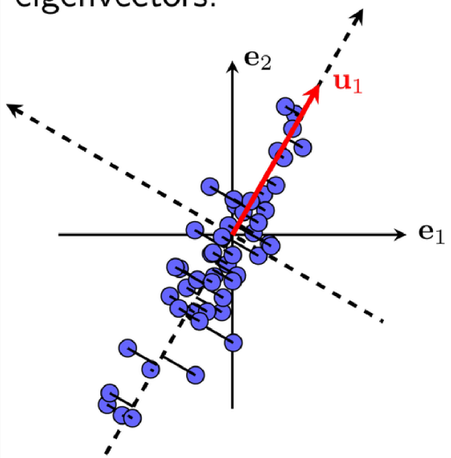
3. Compute covariance matrix:
$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$

4. Computer eigenvalues and eigenvectors of $\mathbf{S}$:
$(\lambda_1, \mathbf{u}_1), \ldots, (\lambda_D, \mathbf{u}_D)$
Remember the orthonormality of $\mathbf{u}_i$.

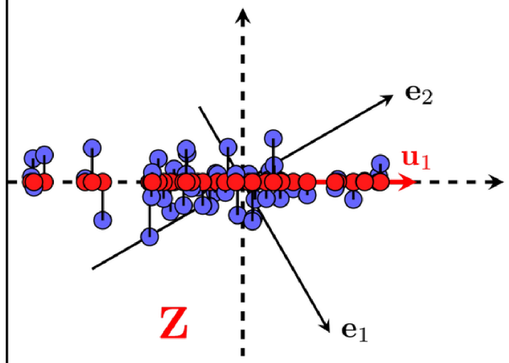
5. Pick K eigenvectors w. highest eigenvalues



6. Project data to selected eigenvectors.



7. Obtain projected points in low dimension.



2.4. PCA & SVD, any relationship?

PCA

- **PCA transforms data**

$$\mathbf{U}_K, \mathbf{Z} = \min_{\mathbf{U}_K, \mathbf{Z}} ||\mathbf{X} - \mathbf{U}_K\mathbf{Z}||_F$$

$$\text{s.t.:} \quad \mathbf{U}_K^T\mathbf{U}_K = \mathbf{I}_K$$

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_K\mathbf{Z}$$

SVD

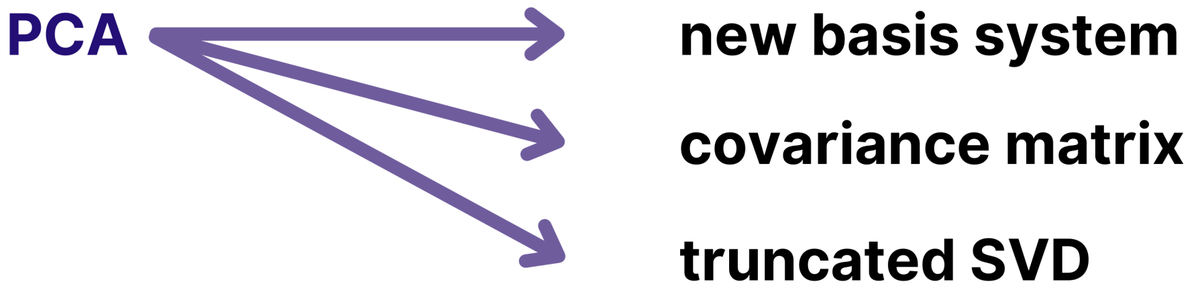
- **SVD: Decompose matrix**

$$\min_{\mathbf{A}} ||\mathbf{X} - \mathbf{A}||_F$$

$$\text{s.t. } \text{rank}(\mathbf{A}) = K$$

$$\mathbf{A} = \mathbf{U}_K\mathbf{\Sigma}_K\mathbf{V}_K^T$$

2.5. PCA - Summarise



Pros:

- Prevent overfitting
- Improve visualization
- Improve performance
- ...

Cons:

- Information Loss
- Scaling data
- Same variance
- ...

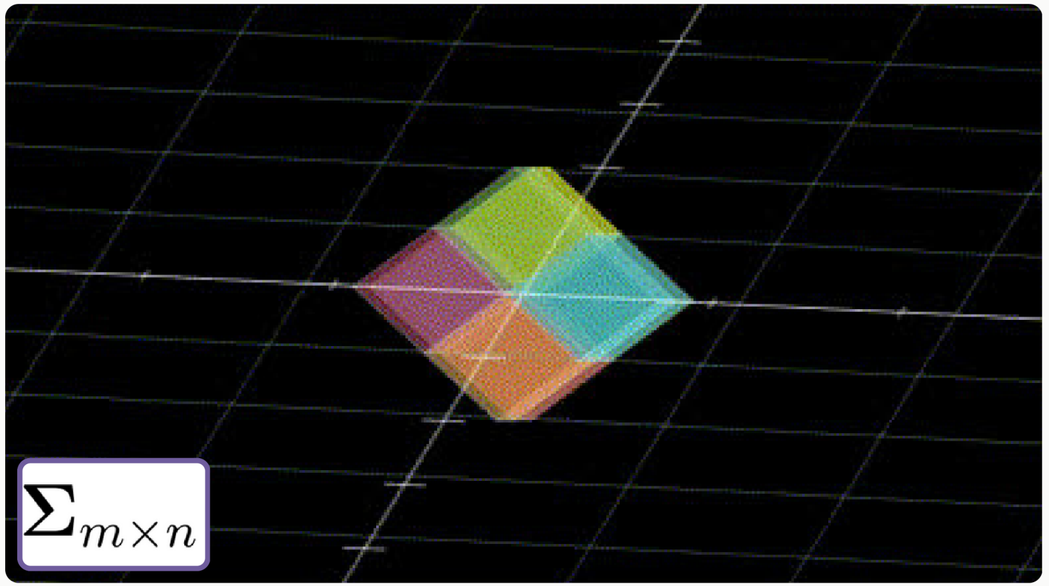
3. Implement in Python

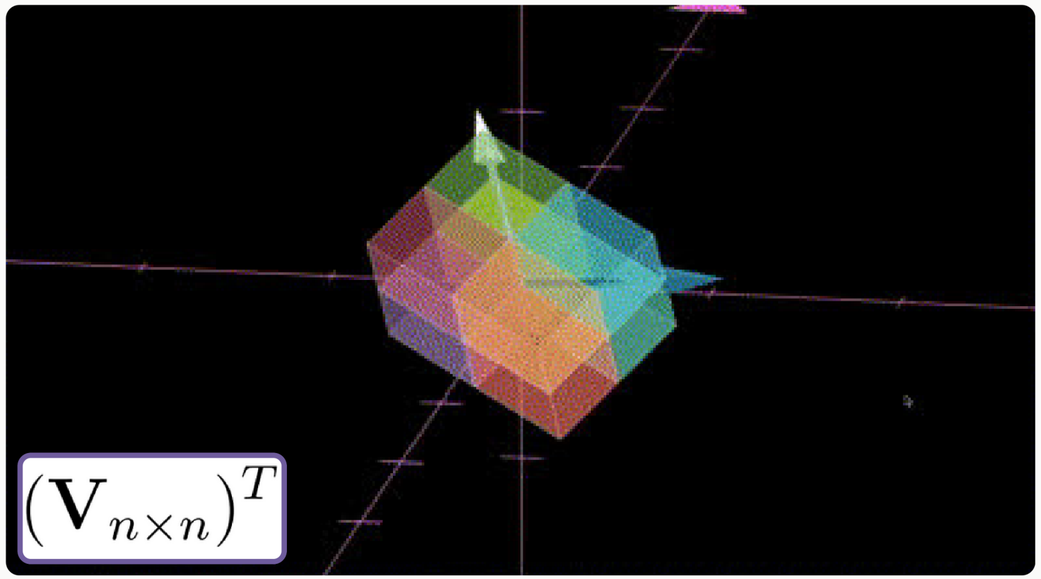
THAT’S IT

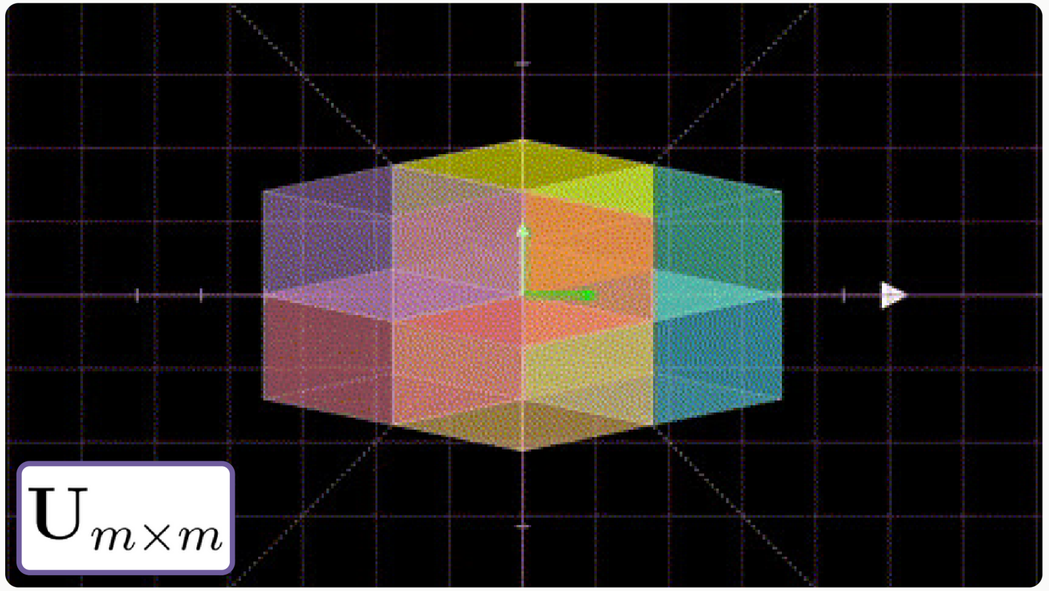
1. SVD? - How?

SVDs -> Best(Low) Rank ‘*k*’ Approximation

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}(\mathbf{V}_{n\times n})^T$$







1. SVD? - How?

SVD in Python