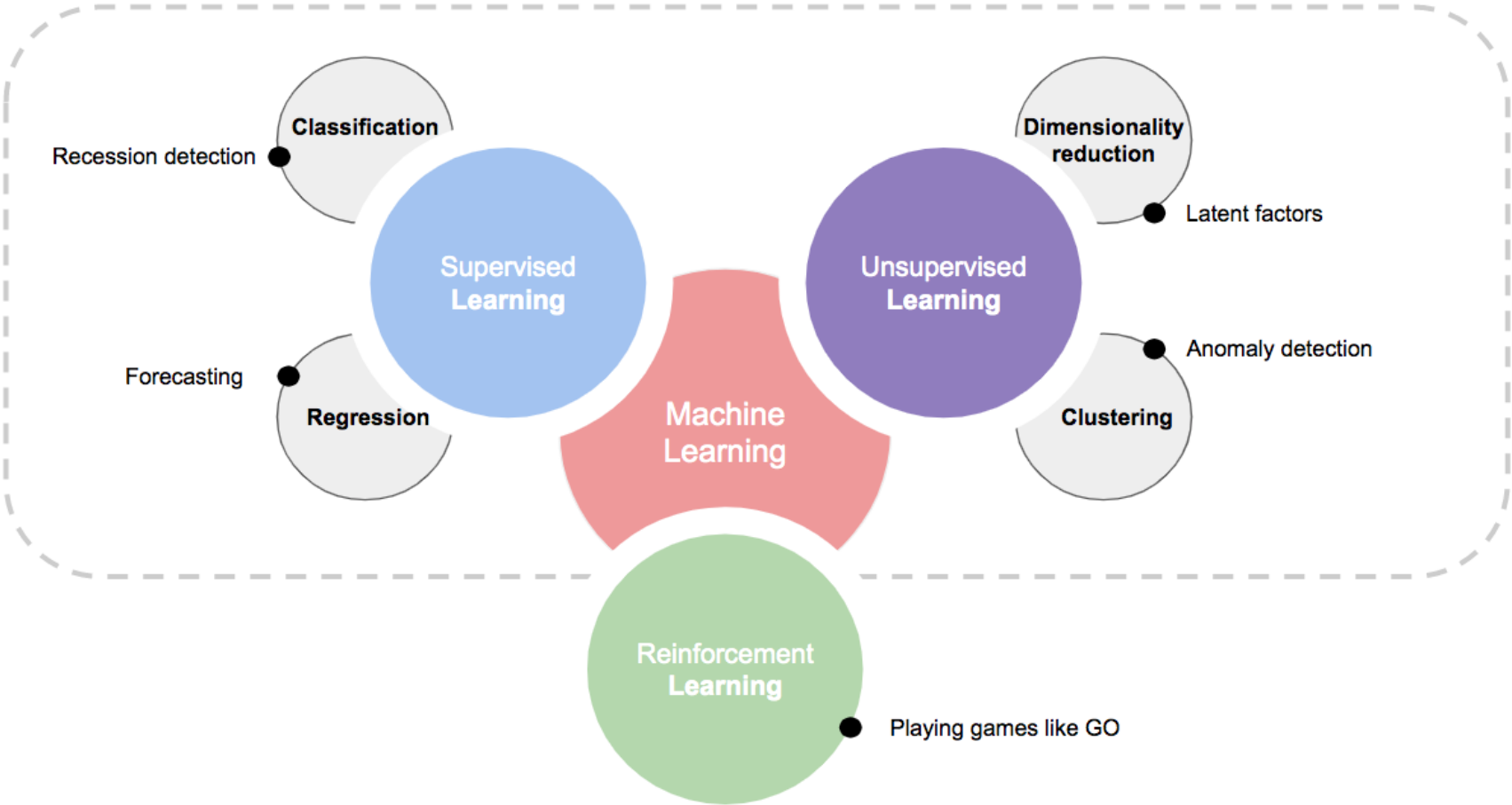
Advanced Business Analytics: Clustering

Mohamed Ndaoud

INTRODUCTION



INTRODUCTION

Overview of supervised and unsupervised learning

- **Supervised learning** is a type of machine learning where the algorithm is **trained on labeled data** to learn the mapping between input and output variables.

- **Unsupervised learning** is a type of machine learning where the algorithm is **trained on unlabeled data** to find patterns or structure in the data.

Real-world data

- There's a **greater amount of unlabeled data** compared to labeled data
- Can be used to train unsupervised models without any human involvement for labelization
- Cannot be used to train supervised models without human intervention

INTRODUCTION
WHY DO WE NEED UNSUPERVISED LEARNING ?

Unsupervised learning can help in :

- Exploring data
- Understanding the data
- Extracting patterns and relevant information

Real-world examples of unsupervised learning, such as anomaly detection, customer segmentation, image clustering, recommendation systems ...

Dimension reduction

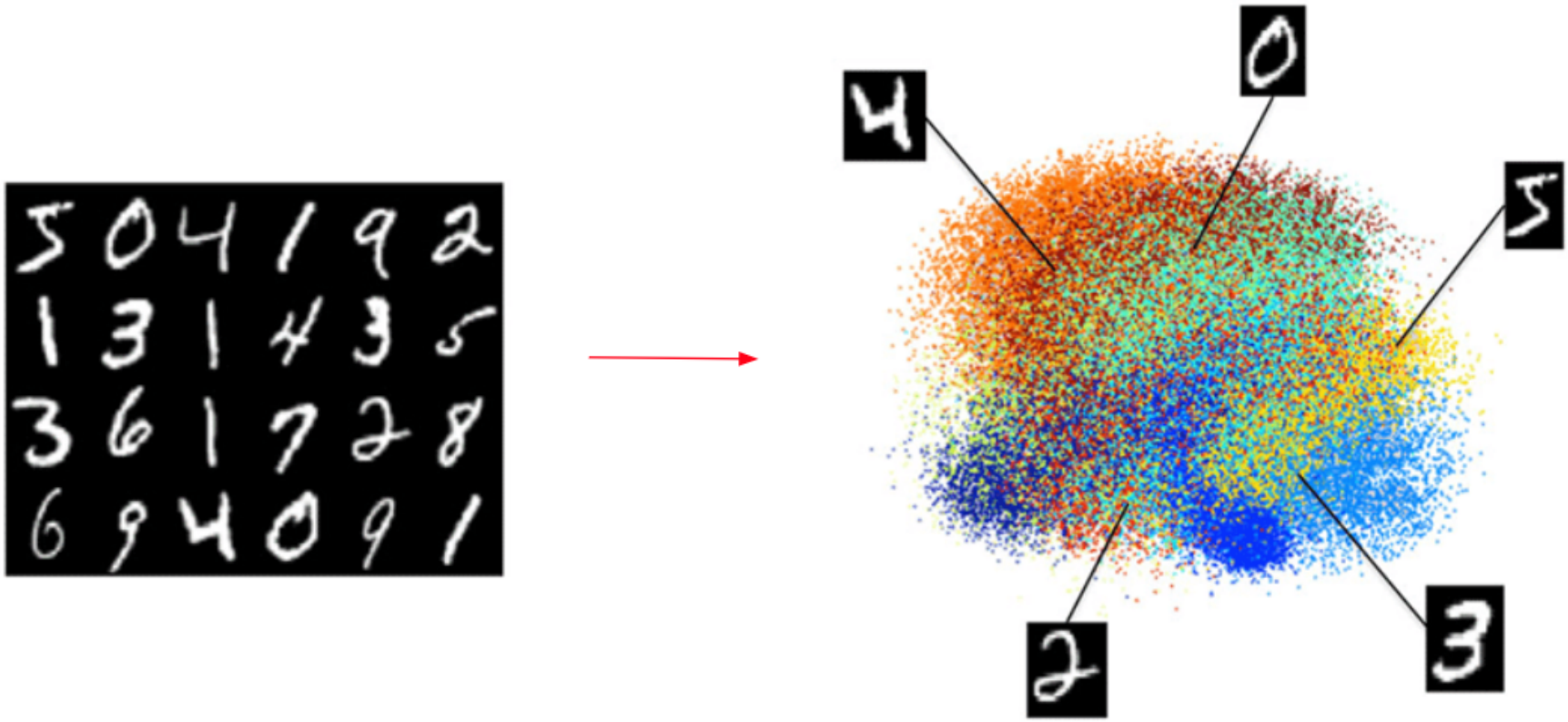
Unsupervised learning techniques like PCA can be used to reduce the dimensionality of the data, which helps in better visualization and faster training of supervised learning models.

Clustering

Unsupervised learning techniques like K-means can be used for grouping similar data points and discovering underlying patterns or structures in the data.

INTRODUCTION
DIMENSION REDUCTION

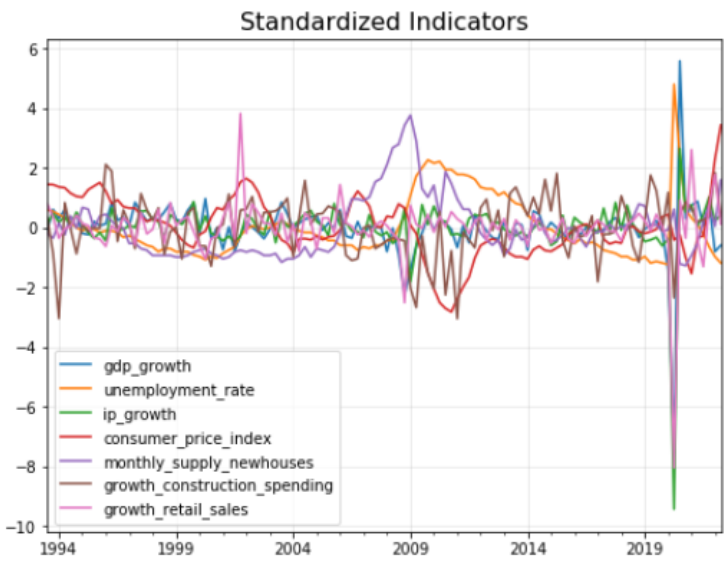
Handwritten Digits

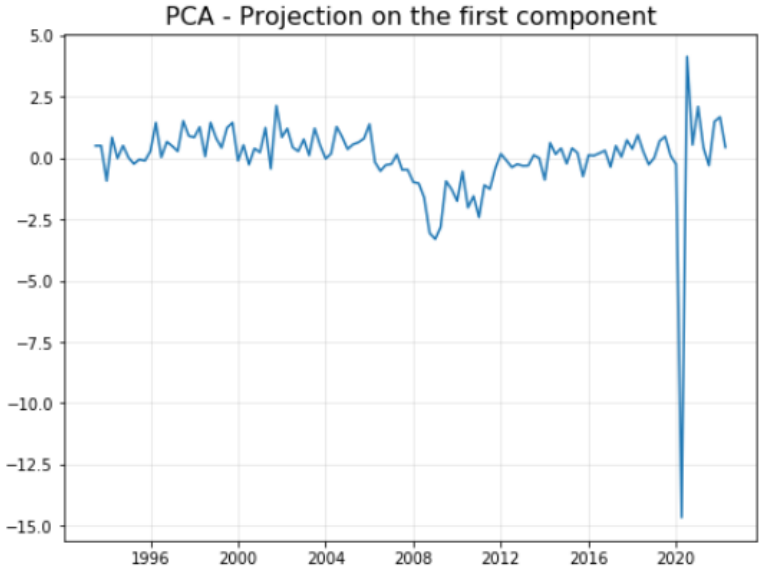


Multi-class Transductive Learning Based on 1 Relaxations of Cheeger Cut and Mumford-Shah-Potts Model

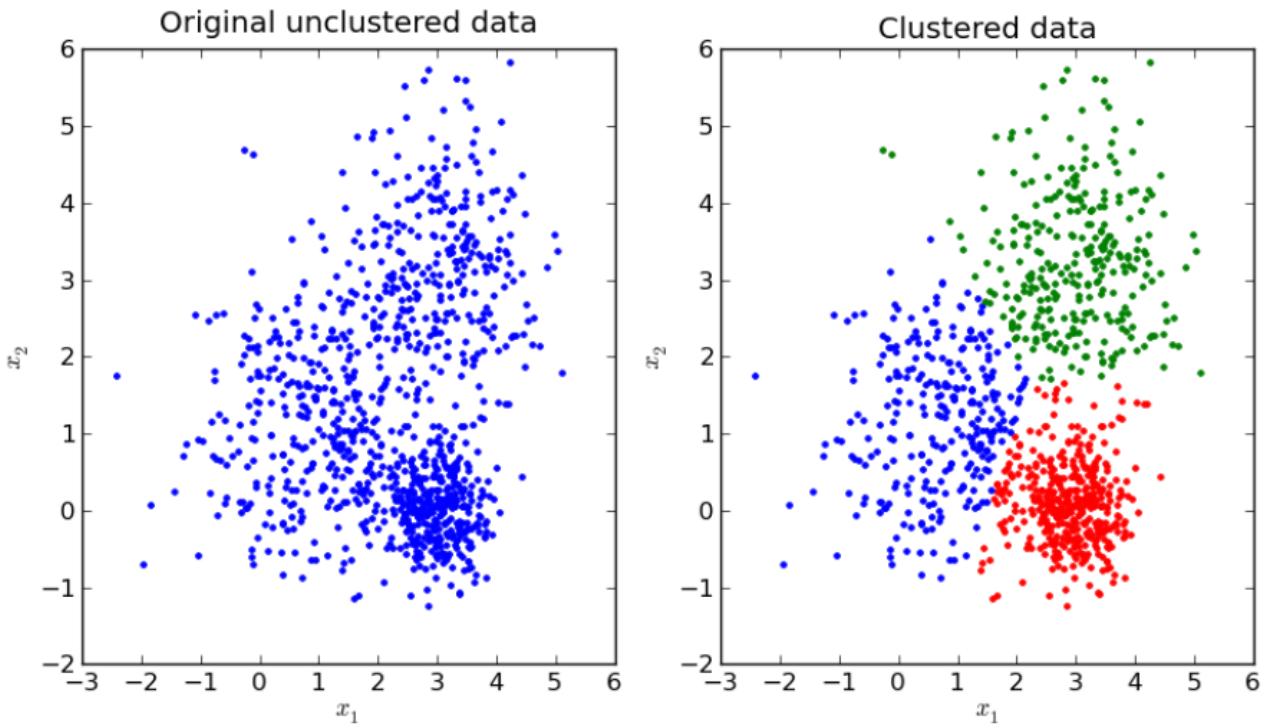
INTRODUCTION
DIMENSION REDUCTION

Macroeconomic latent factor



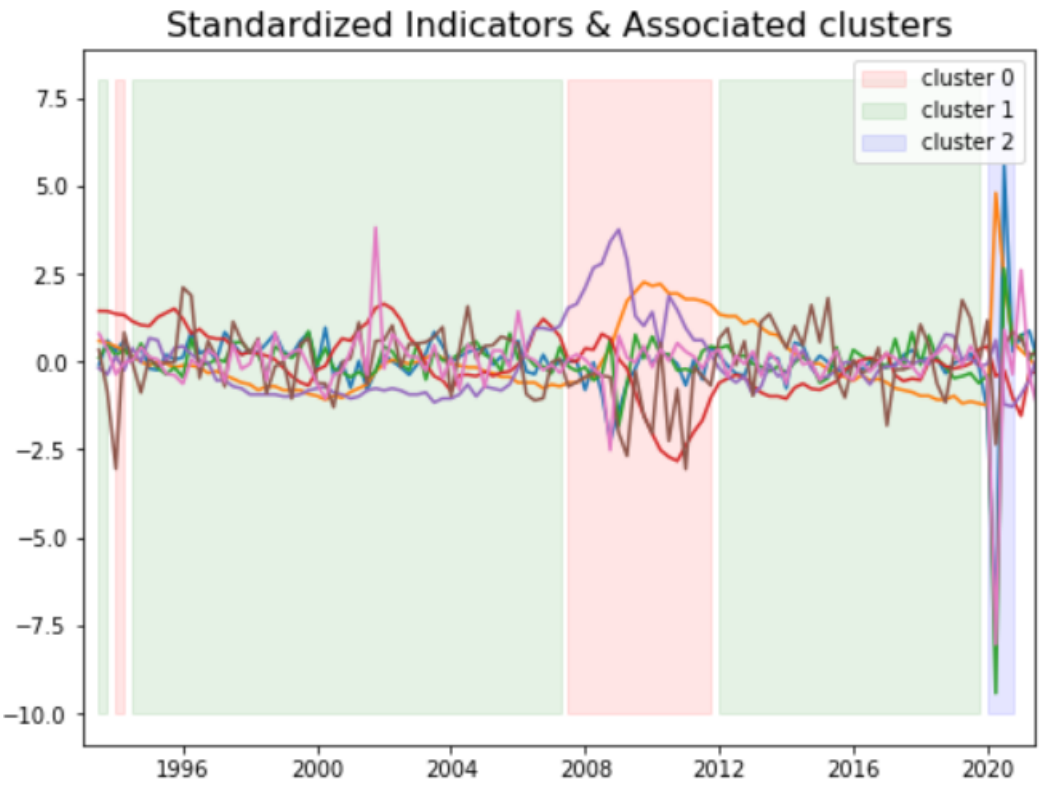


INTRODUCTION CLUSTERING



INTRODUCTION CLUSTERING

Macroeconomic clustering



SUMMARY

Contents

DIMENSION REDUCTION USING PCA

Motivations

"Curse of dimensionality"

- High-dimensional data is **difficult to analyze and visualize** :
  difficult to plot data in more than three dimensions.
  difficult to understand the relationships between the variables.

- Dimensionality reduction can help :

  Definition Process of reducing the number of variables in a dataset while retaining the most important information.

  $\longrightarrow$ Simplify complex data
  $\longrightarrow$ Facilitate visualization
  $\longrightarrow$ Improve data compression
  $\longrightarrow$ Create relevant features
  $\longrightarrow$ Reduce noise
  $\longrightarrow$ Enhance predictive modeling

DIMENSION REDUCTION USING PCA

Goals

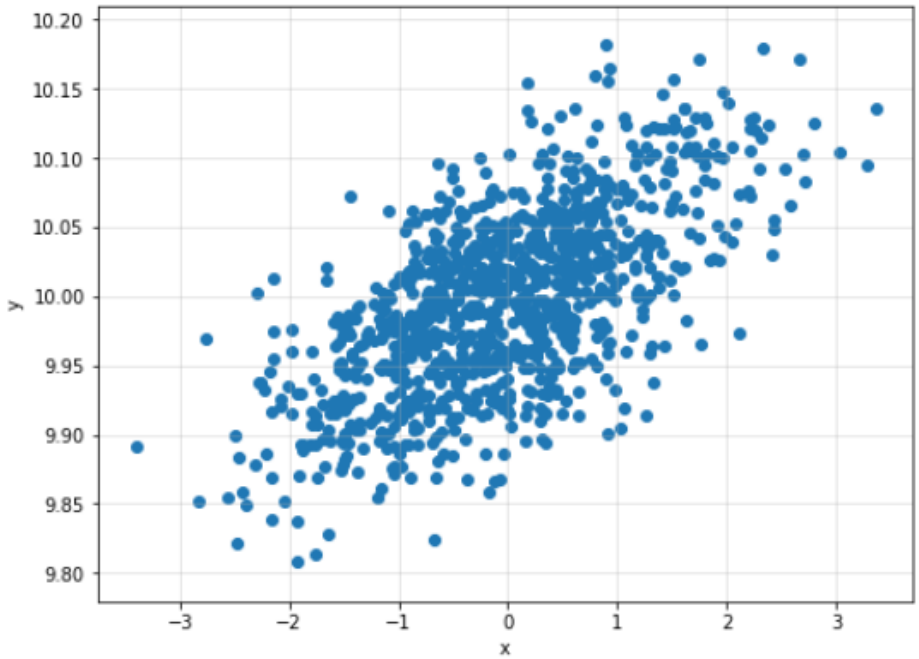
Definition

Principal Component Analysis : its primary goal is to reduce the dimensionality of a dataset while retaining as much information as possible (measured by the variance of the data).

Practical points

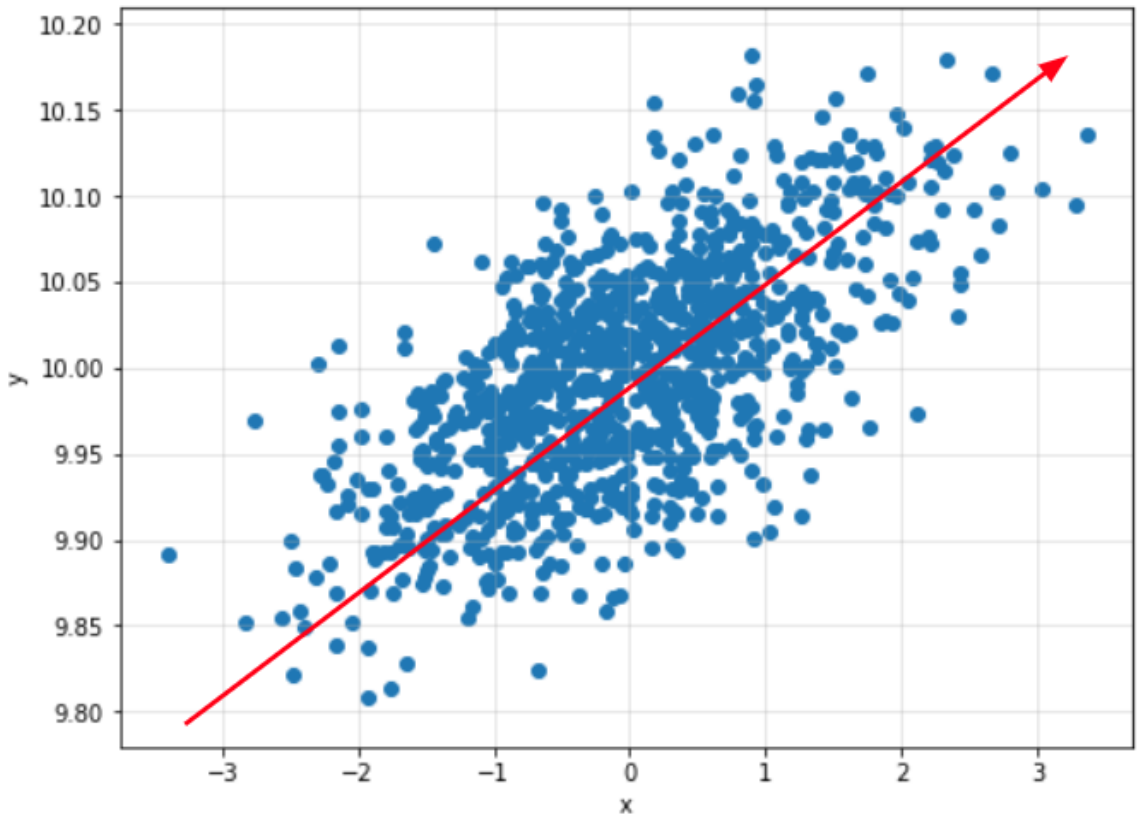
- PCA creates a **new basis for the data** by constructing orthogonal principal components.
- These **principal components** capture the most significant variation in the data.
- The first principal component captures the maximum amount of variation / variance
- The second principal component, which captures the maximum amount of remaining variance
- And so on ...

DIMENSION REDUCTION USING PCA



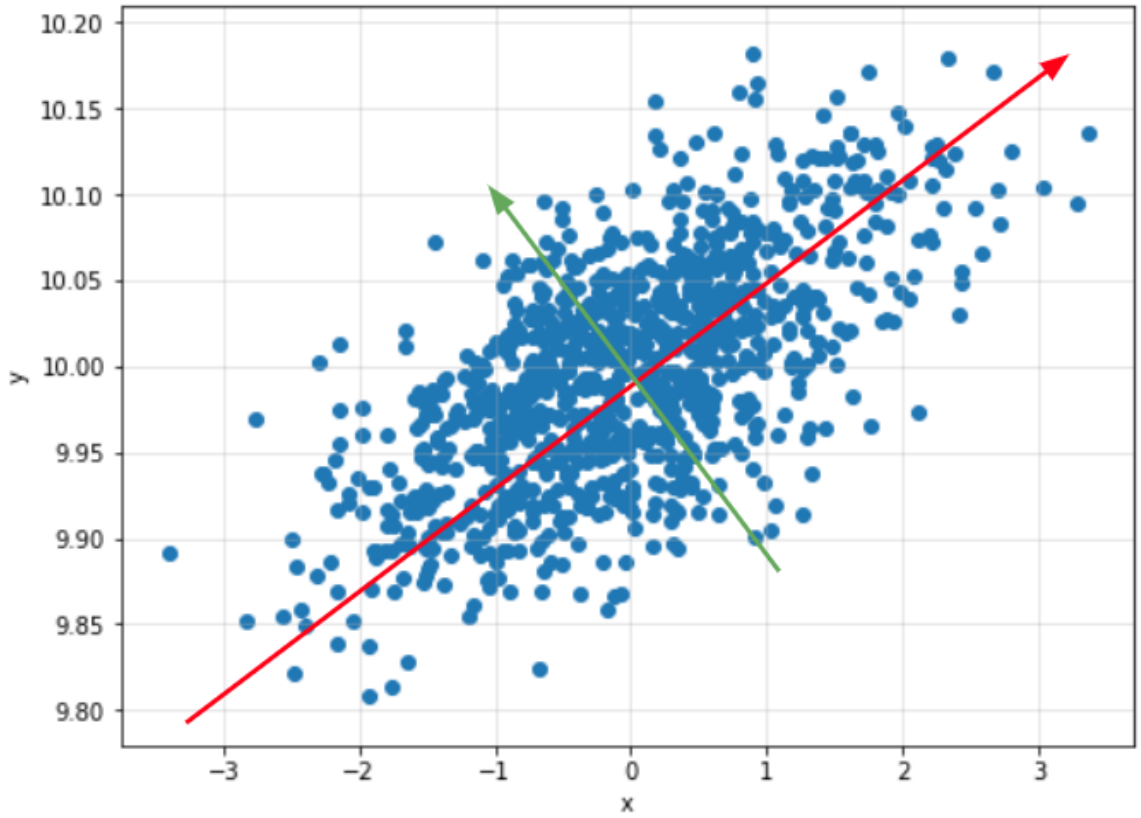
- How many new components can we create ?
- How could a new axis be created to explain the most variation in the data ?

DIMENSION REDUCTION USING PCA



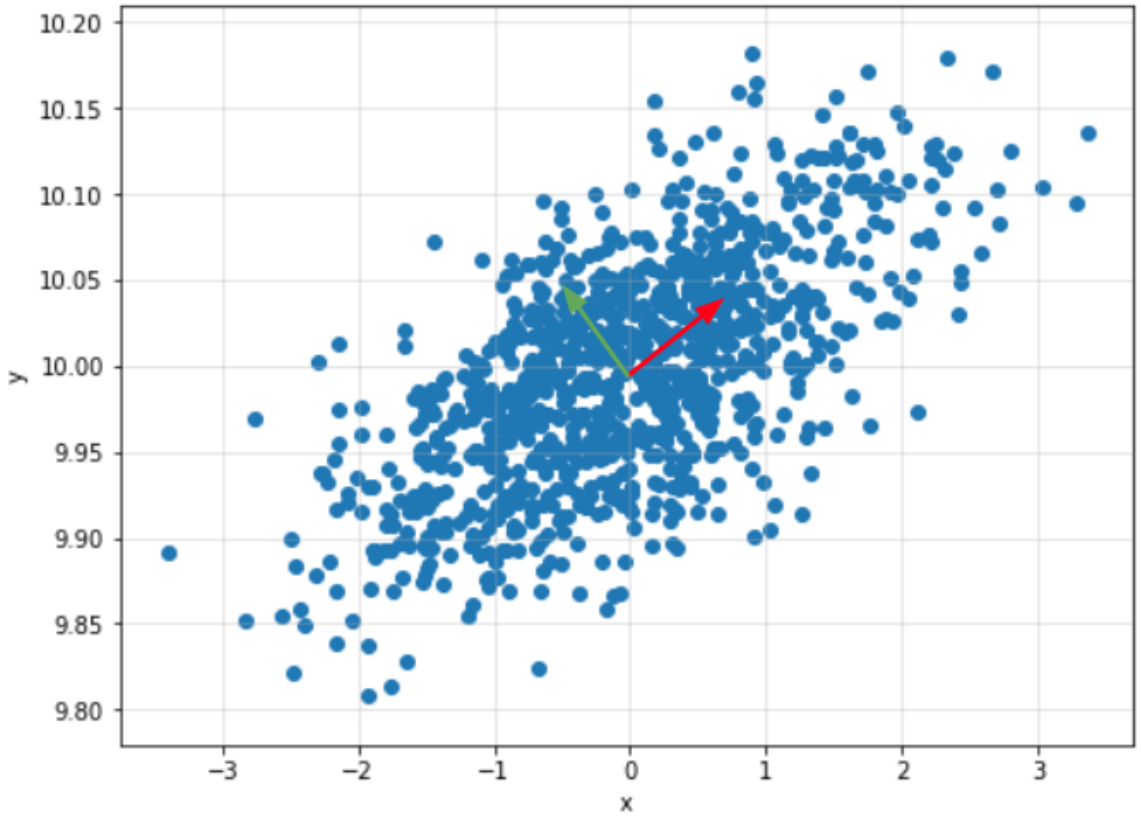
- How to create a second axis (orthogonal) to explain the remaining variance ?

DIMENSION REDUCTION USING PCA



- How to normalize the new basis ?

DIMENSION REDUCTION USING PCA



Orthonormal basis

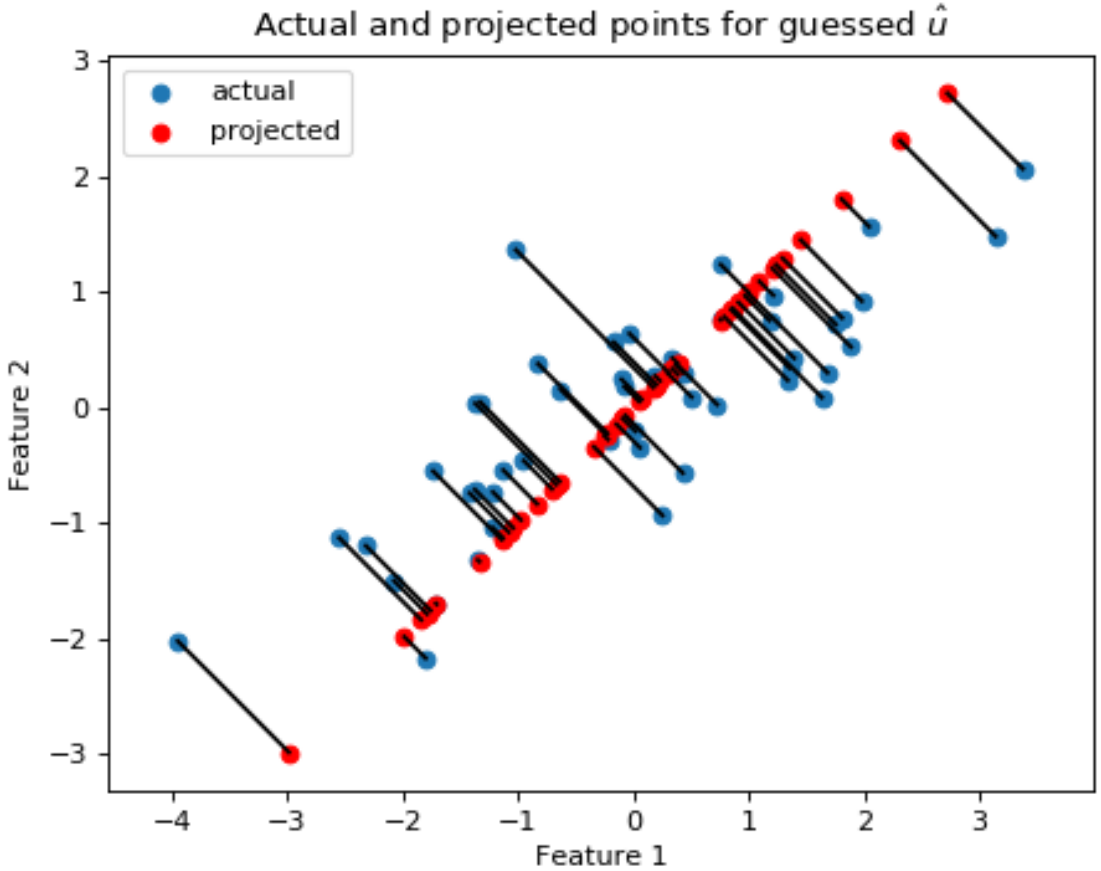
- $u_1^T u_2 = 0$
- $||u_1||^2 = u_1^T u_1 = 1$ and $||u_2||^2 = u_2^T u_2 = 1$

What is the dimension of our new vectors ?

DIMENSION REDUCTION USING PCA

How to reduce the dimension of the dataset ?

$\longrightarrow$ *Keep the coordinate of each point on the first component.*



CONTEXT AND OBJECTIVES

Let N data points in dimension d : $x_1, \cdots, x_n \in \mathbb{R}^d$

$$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,d} \\ x_{2,1} & \cdots & x_{2,d} \\ & \cdots & \\ x_{n,1} & \cdots & x_{n,d} \end{pmatrix} \in \mathbb{R}^{n \times d}$$

One want to reduce the dimension from d to m $(m \leq d)$:

$$Z = \begin{pmatrix} z_{1,1} & \cdots & z_{1,m} \\ z_{2,1} & \cdots & z_{2,m} \\ & \cdots & \\ z_{n,1} & \cdots & z_{n,m} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

$\longrightarrow$ We want to get Z by projecting X on U :

$$U = \begin{pmatrix} | & \cdots & | \\ u_1 & \cdots & u_m \\ | & \cdots & | \end{pmatrix} \in \mathbb{R}^{d \times m}$$

such as

$$\underbrace{Z}_{(n \times m)} = \underbrace{X}_{(n \times d)} \underbrace{U}_{(d \times m)}, \text{ with } U^T U = I \text{ (orthonormal basis)}$$

CONTEXT AND OBJECTIVES

$$Z = XU, \text{ with } U^T U = I \text{ (orthonormal basis)}$$

$$U^T U = I \longrightarrow \begin{cases} \forall i \neq j, \ u_j^T u_i = 0 \\ \forall i, \ ||u_i||^2 = u_i^T u_i = 1 \end{cases}$$

Simplest case $\longrightarrow$ for a sample x

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \in \mathbb{R}^d$$

- **Decomposition** ('projection')

$$\underbrace{z}_{(m\times 1)} = \underbrace{U^T}_{(m\times d)} \underbrace{x}_{(d\times 1)}$$

- **Reconstruction** : with 'potential loss' of information

$$\underbrace{x^{recons}}_{(d\times 1)} = \underbrace{U}_{(d\times m)} \underbrace{z}_{(m\times 1)}$$

What do we need to perform the projection and then the reconstruction ?

HOW TO GET THE PRINCIPAL COMPONENTS ?

PCA both fulfills both **optimization criteria** simultaneously :

(1) The minimization of the reconstruction error
The objective is to keep the distance between the reconstruction and the original instance as small as possible

$$U = \arg \min_{U^T U = I} \sum_{i=1}^{n} ||x_i - x_i^{recons}||^2 = \arg \min_{U^T U = I} \sum_{i=1}^{n} ||x_i - U z_i||^2$$

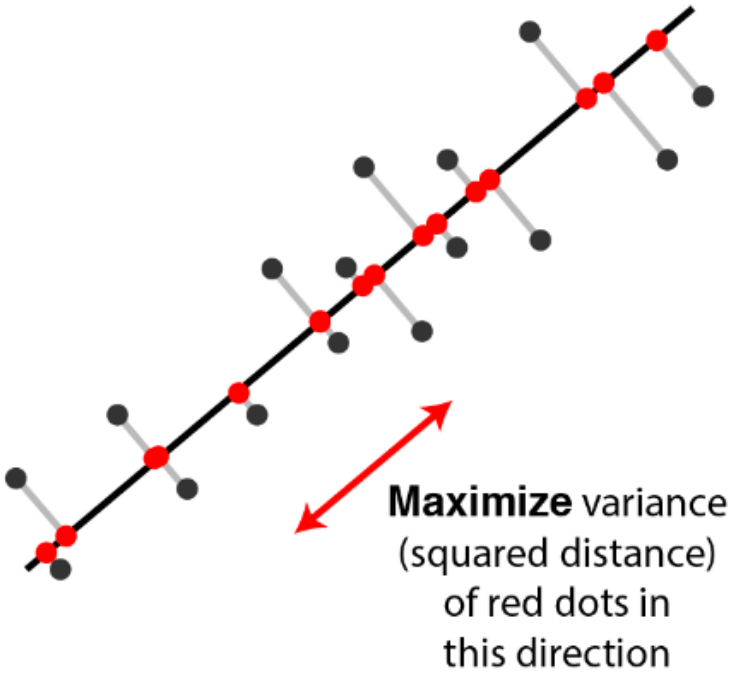
$$= \arg \min_{U^T U = I} \sum_{i=1}^{n} ||x_i - U U^T x_i||^2$$

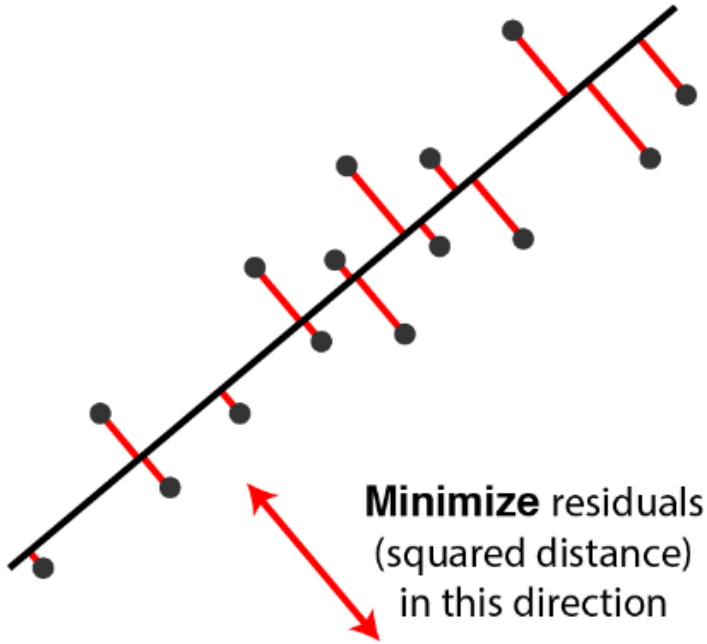
(2) The variance maximization objective
it aims to choose U to maximize the variance of the data projected onto U.

$$U = \arg \max_{U^T U = I} \sum_{i=1}^{n} ||z_i||^2 = \arg \max_{U^T U = I} \sum_{i=1}^{n} ||U^T x_i||^2$$

$$(1) \iff (2)$$

HOW TO GET THE PRINCIPAL COMPONENTS ?





Source : Peter Bloem

How to solve the problem ?

SOLVING THE PROBLEM

Illustration with the first component

Problem :

$$u_1 = \arg \max_{U^T U = I} \frac{1}{N} \sum_{i=1}^{n} ||u^T x_i||^2 = \arg \max_{||u||=1} \frac{1}{N} ||Xu||^2$$

$$= \arg \max_{||u||=1} u^T (\frac{1}{N} X^T X) u$$

Solution :

u_1 : 1st eigenvector of the matrix $X^T X$ associated to the eigenvalue λ_1

$\frac{1}{N} X^T X$: covariance matrix of X

SOLVING THE PROBLEM

Global solution :

Overall the principal components are the eigenvectors of the matrix $X^T X$.

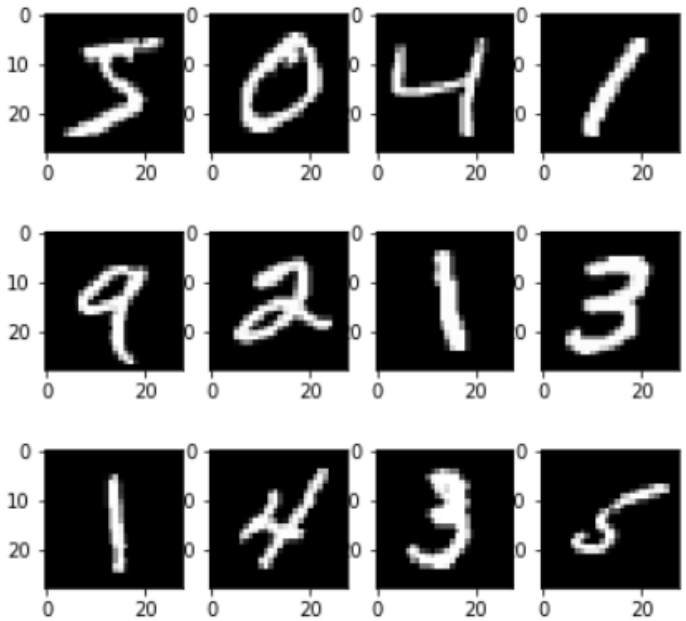
- u_1 : 1st eigenvector associated to the eigenvalue λ_1
- u_2 : 2nd eigenvector associated to the eigenvalue λ_2

...

- u_d : dth eigenvector associated to the eigenvalue λ_d

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d \geq 0$$

DIGITS APPLICATION



1 digit $= 28 \cdot 28$ pixels $= 784$ pixels

How to represent a single digit ?

$$x_i^{img} = \begin{pmatrix} x_{i,1} & \dots & x_{i,28} \\ x_{i,29} & \dots & x_{i,56} \\ & \dots & \\ x_{i,757} & \dots & x_{i,784} \end{pmatrix}$$

or

$$x_i^{usual} = (x_{i,1}, \cdots, x_{i,784})^T$$

DIGITS APPLICATION

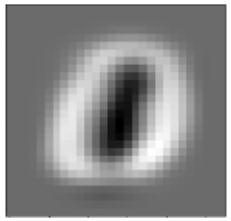
First component

Component visualization

$$U = \begin{pmatrix} | \\ u_1 \\ | \end{pmatrix} \in \mathbb{R}^{d \times 1}, d = 784$$

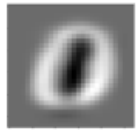
$$u_1^{usual} = (u_{1,1}, u_{1,2}, \cdots, u_{1,784})^T$$
$$= (7e-18, 1e-17, ..., 0)^T$$

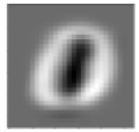
$$u_1^{img} = \begin{pmatrix} u_{1,1} & \dots & u_{1,28} \\ u_{1,1} & \dots & u_{1,56} \\ & \dots & \\ u_{1,757} & \dots & u_{1,784} \end{pmatrix}$$

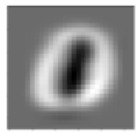


Reduction

$$z_i = U^T x_i$$

$Z_1 = U^T$   = 123.93

$Z_2 = U^T$   = 1011.71

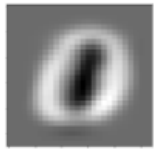
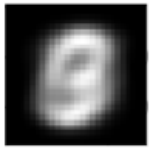
$Z_3 = U^T$   = -51.84

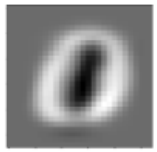
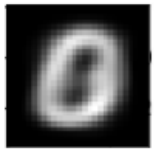
DIGITS APPLICATION

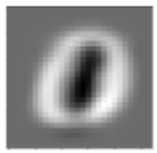
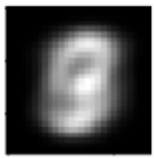
First component

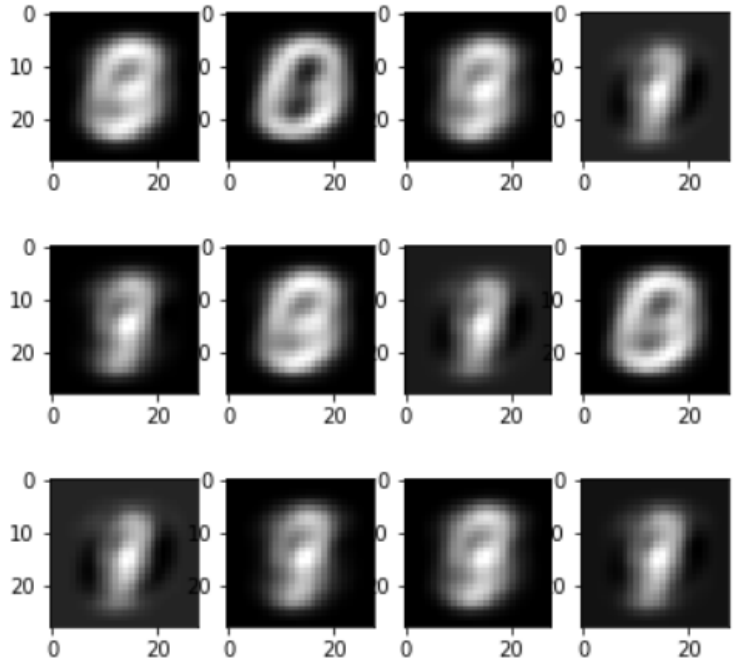
Reconstruction

$$x_i^{recons} = Uz_i$$

$x_1^{recons} =$  $z_1 =$ 

$x_2^{recons} =$  $z_2 =$ 

$x_3^{recons} =$  $z_3 =$ 



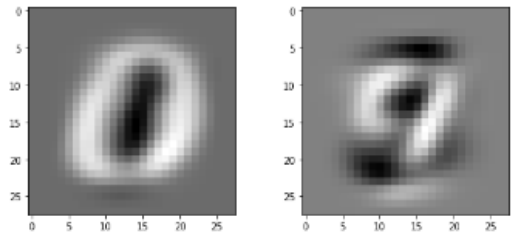
Can we distinguish the differences ?

DIGITS APPLICATION

2 components

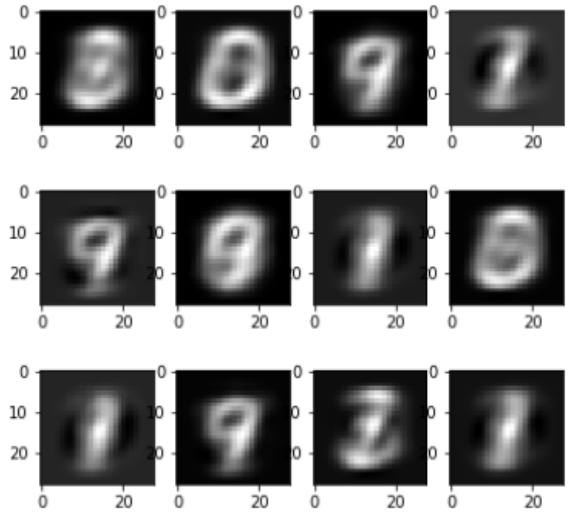
Components visualization

$Z_i = X_i^T$



Reconstruction

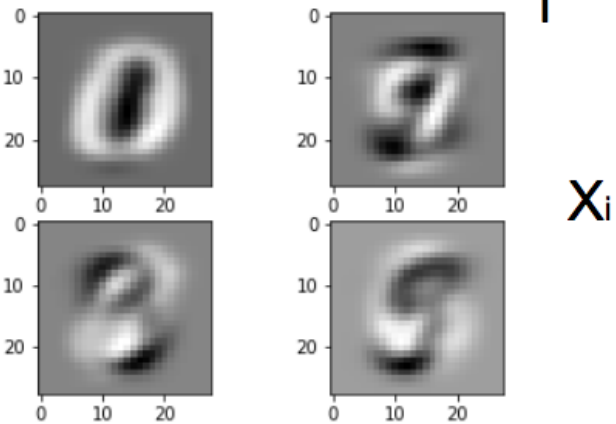
$$x_i^{recons} = Uz_i$$



DIGITS APPLICATION

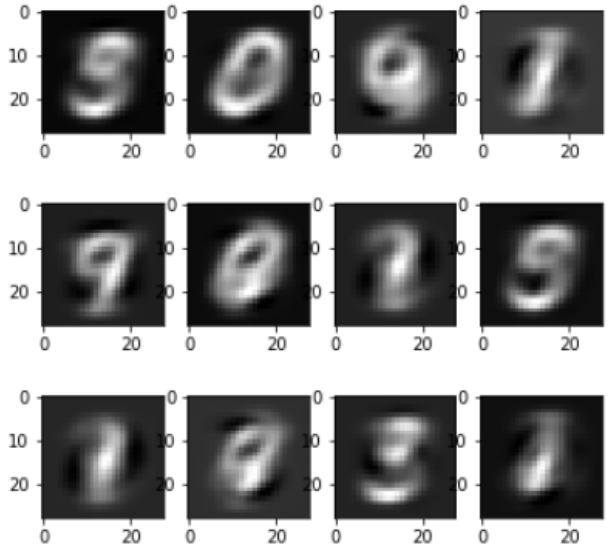
4 components

Components visualization

$Z_i =$  ^{T} X_i

Reconstruction

$$x_i^{recons} = Uz_i$$



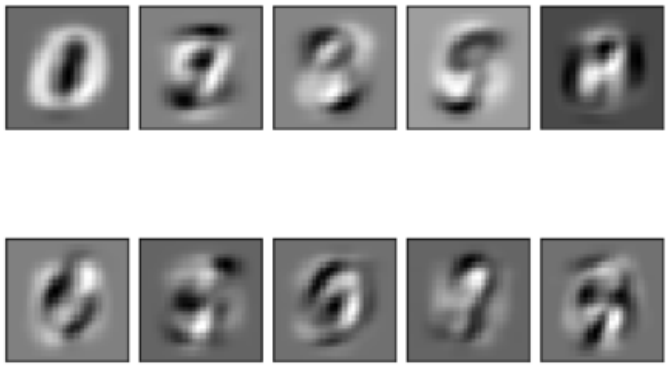
- At the very beginning we extract global variations (valid for all digits).
- The more we advance in the components, the more we capture granular variations (specific to each digit).

DIGITS APPLICATION

10 components

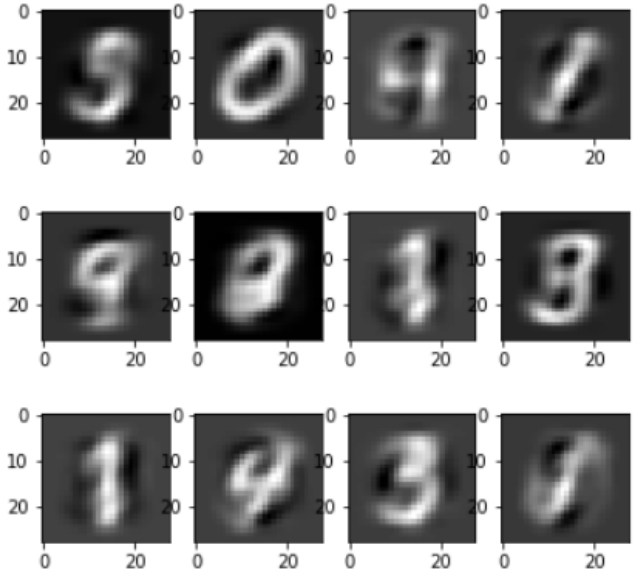
Components visualization

$Z_i = X_i^T$



Reconstruction

$$x_i^{recons} = Uz_i$$

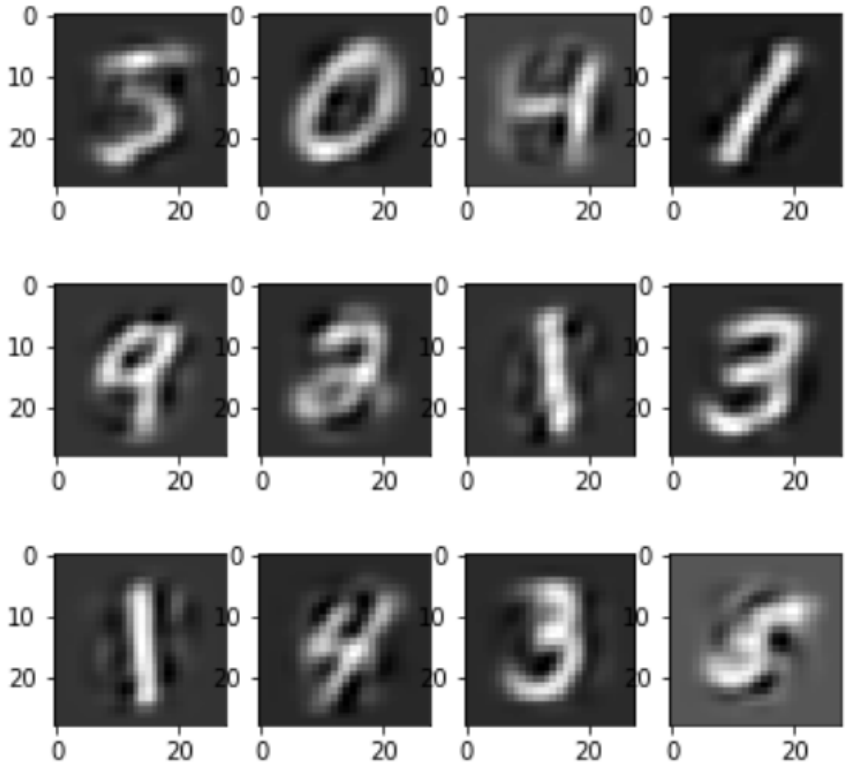


DIGITS APPLICATION

Reconstruction

$$x_i^{recons} = Uz_i$$

28 components



DIGITS APPLICATION

Reconstruction

$$x_i^{recons} = Uz_i$$

49 components

DIGITS APPLICATION

Reconstruction

$$x_i^{recons} = Uz_i$$

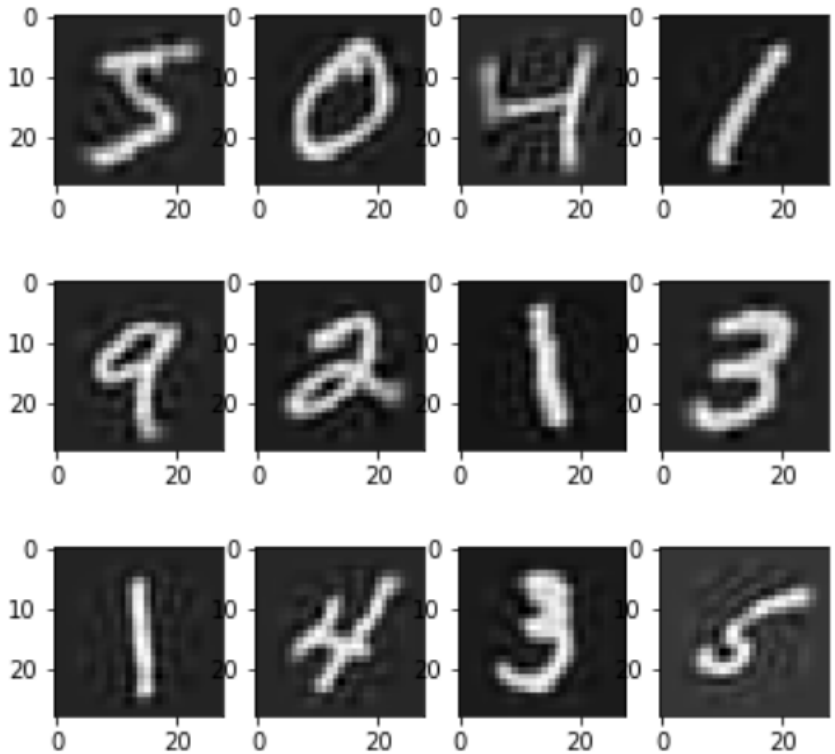
98 components

DIGITS APPLICATION

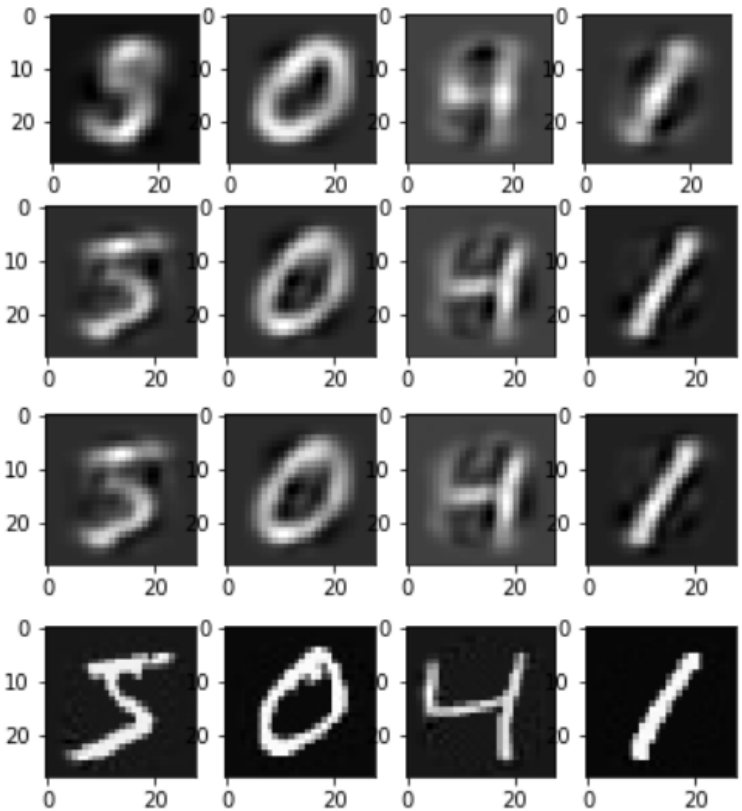
112 components

Reconstruction

$$x_i^{recons} = Uz_i$$



DIGITS APPLICATION



How do we choose the number of components ?

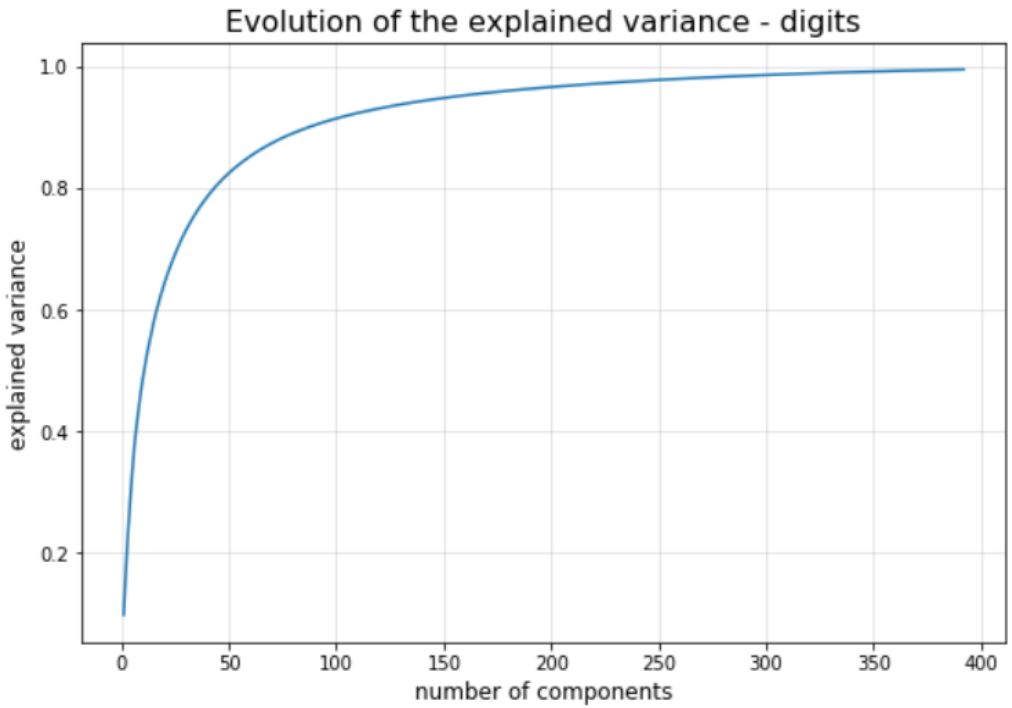
$\longrightarrow$ concept of explained variance

EXPLAINED VARIANCE

How much of the total variation in the original dataset is accounted for by each principal component in PCA ?

Proportion of explained variance

$$V_m = \frac{\sum_{j=1}^{m} \lambda_j}{\sum_{j=1}^{d} \lambda_j}$$



- Look at the spectrum of variance, pick K to capture most of the variance

"The Pareto principle, also known as the 80/20 rule."

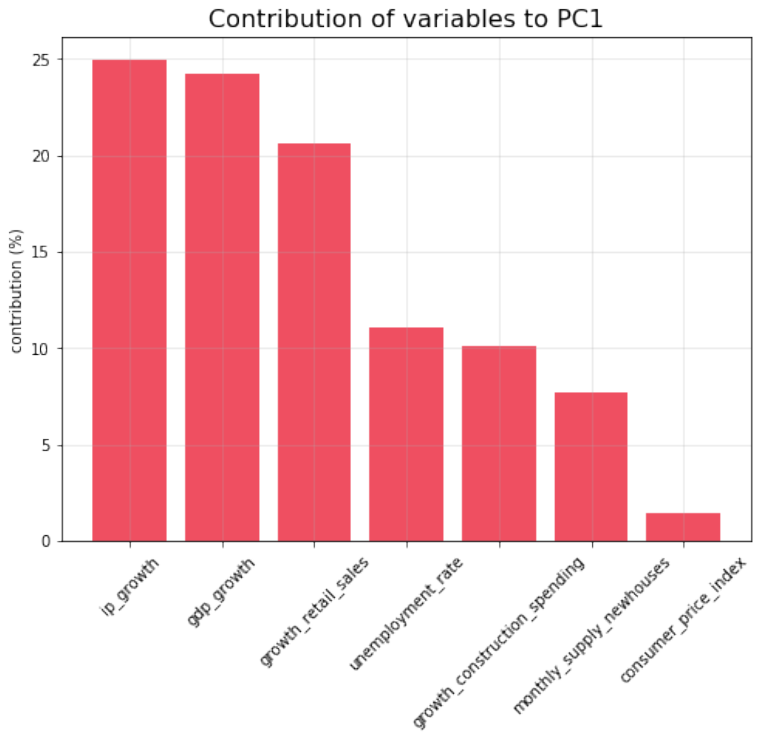
INTERPRETING VARIABLE CONTRIBUTION IN PCA

- **Variable contribution** is a measure of how much each variable contributes to the definition of a principal component.

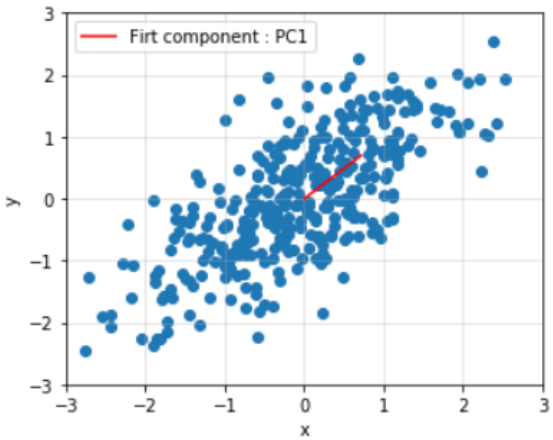
- The higher the contribution percentage, the more important the variable is in defining the component.

$$PC_1 = (PC_{1,1}, PC_{1,2}, \cdots, PC_{1,p})$$

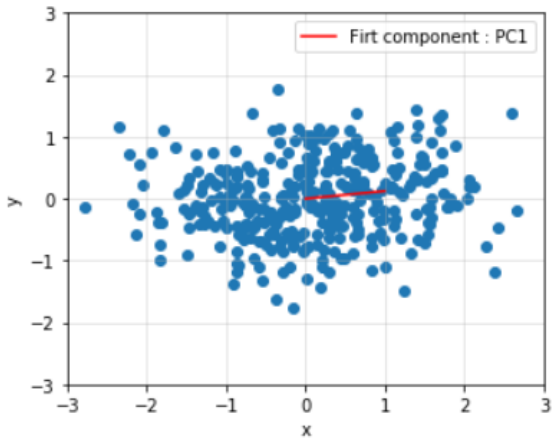
$$Cont_{PC_1} = \frac{1}{\sum_j PC_{1,j}^2} (PC_{1,1}^2, \cdots, PC_{1,p}^2)$$



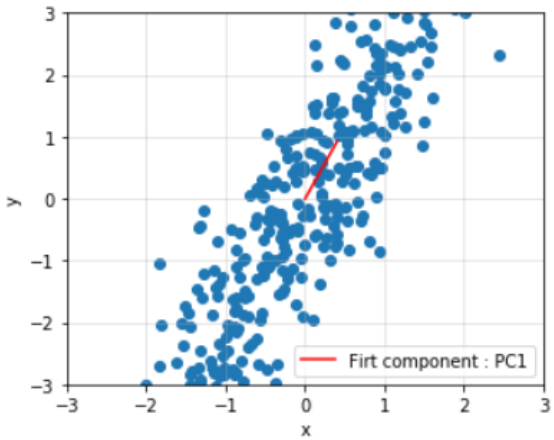
INTERPRETING VARIABLE CONTRIBUTION IN PCA



Contribution to PC1:
x : 54%
y : 46%



Contribution to PC1:
x : 94%
y : 4%



Contribution to PC1:
x : 16%
y : 84%

PCA PRACTICE

In practice :

```
n_components = 2
pca = PCA(n_components).fit(X)
U = pca.components_ # Access to principal components U
pc1 = U[0] # Access to first component
Z = pca.transform(X) # Projection
X_recons = pca.inverse_transform(Z) # Reconstruction
explained_variance = pca.explained_variance_ratio_ # Access to variances
cont_pc1 = pc1**2/np.sum(pc1**2) # contribution of variables
```

Contents

CLUSTERING WITH K-MEANS

Definition of clustering

- Process of grouping similar data points together

Definition of Kmeans

- K-means is a popular clustering algorithm
- Partitions data into **K clusters based on their similarity ($\longrightarrow$ distance)**.
- The middle of the cluster is represented by **centroids**

Intuition behind K-means algorithm

- K-means works by iteratively updating the cluster centroids until the variation within each cluster is minimized.

CENTROIDS IN KMEANS

Centroids

Centroids are the average of all the data points assigned to each cluster :

$$centroid_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

- Recalculated in each iteration of the algorithm.
- Initialized randomly.

INERTIA IN KMEANS

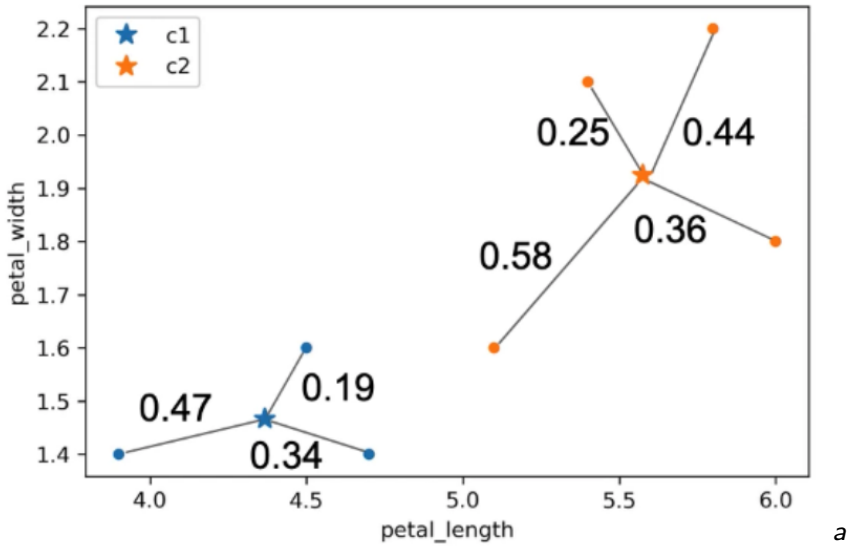
Inertia

- The sum of the squared distances of each point to its assigned centroid

$$inertia(C_i) = \sum_{x_i \in C_i} ||x_i - centroid_i||^2$$

$$inertia(C) = \sum_{i=1}^{K} inertia(C_i)$$

- It serves as the objective function of K-means clustering
- The algorithm seeks to minimize inertia.



a

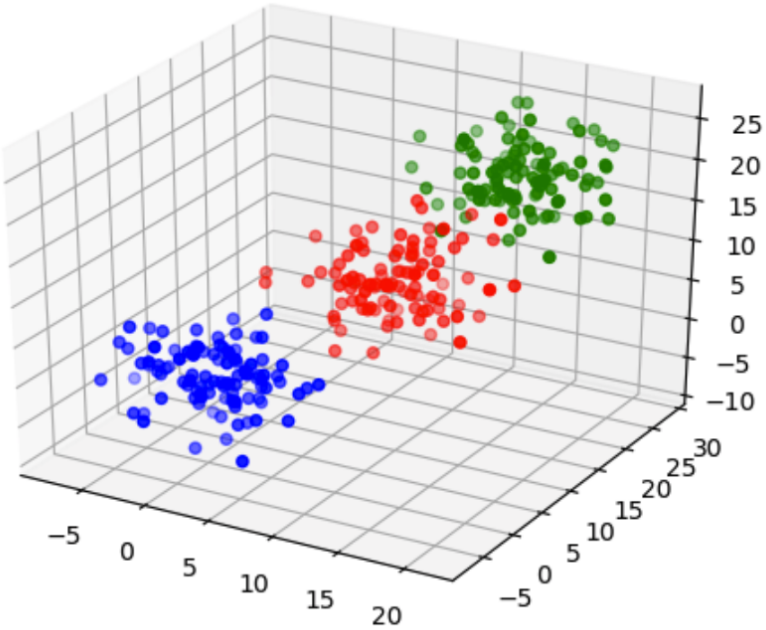
$$inertia(C_1) = 0.47^2 + 0.19^2 + 0.34^2 = 0.3726$$

$$inertia(C_2) = 0.58^2 + 0.36^2 + 0.44^2 + 0.25^2 = 0.7221$$

$$inertia(C) = inertia(C_1) + inertia(C_2) = 1.0947$$

a. *Hug J. (2019). Lecture 24 – Clustering.*

EXAMPLES WITH LIMITED NUMBERS OF VARIABLES

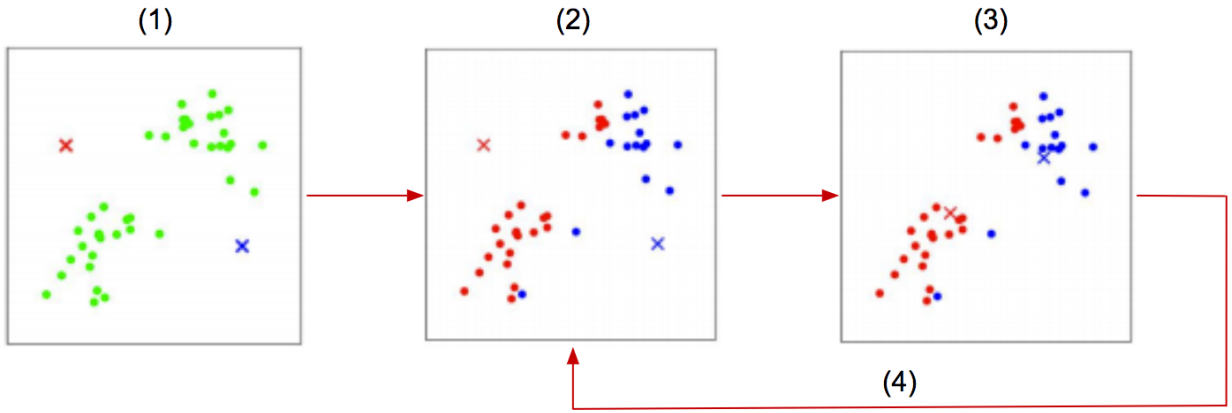


Interpretation :
K-means can provide insights into the underlying structure of the data, revealing patterns and relationships that were not previously visible.

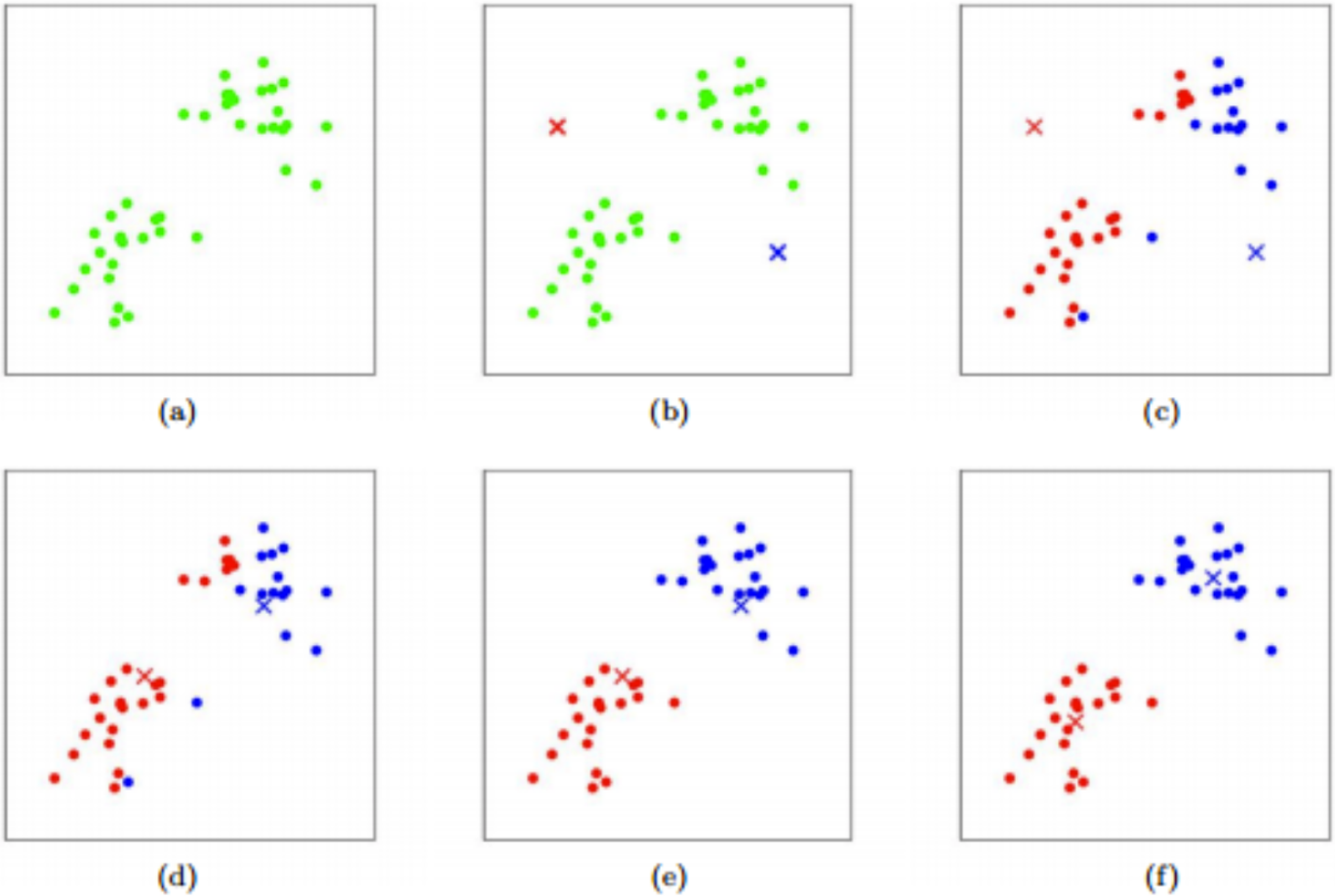
Limitations :
Examples with two or three variables may not reflect the complexity of real-world datasets, where many variables are involved and the data is high-dimensional.

KMEANS ALGORITHM

- **(1) Initialization** : Choose the number of clusters (k) and randomly assign k centroids to the data points in the dataset.
- **(2) Assigning data points to clusters** : For each data point in the dataset, calculate its distance to each centroid and assign it to the cluster corresponding to the nearest centroid.
- **(3) Updating centroids** : For each cluster, calculate the mean of all the data points assigned to it and update the centroid to this mean value.
- **(4) Repeat steps 2 and 3 until convergence** : Re-assign each data point to the nearest centroid and update the centroids until they no longer change or a stopping criterion is met.



KMEANS ALGORITHM IN ACTION



KMEANS ALGORITHM

Algorithm 1 Kmeans

Hyperparameters :
- d : distance
- k : number of clusters
- ϵ : stopping criterion

Training step : $\mathcal{D}_n = \{x_1, \cdots, x_i, \cdots, x_n\}$
- Step 1 : Initialize cluster centroids $\mu_1, \mu_2, \cdots, \mu_k$

- Step 2 : Assign data points to clusters

$$\forall i, C^{(i)} = \arg \min_{j \in \{1,k\}} ||x_i - \mu_{(j)}||^2$$

- Step 3 : Update centroids

$$\forall j, \mu_j = \frac{1}{|C_j|} \sum_{x_i \in |C_j|} x_i$$

- Step 4 : Repeat until convergence - at step m

$$|inertia_m - inertia_{m-1}| \leq \epsilon$$

Predicting step : with a new query point x_0
- Find the closest centroid μ_j and assign the associated cluster j.

KMEANS EVALUATION

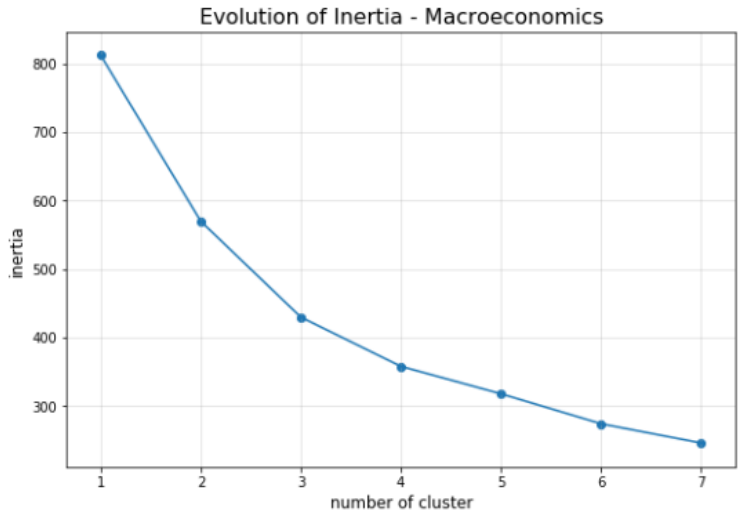
"Clustering is an unsupervised learning technique, so it is hard to evaluate the quality of the output of any given method."
- Page 534, Machine Learning : A Probabilistic Perspective, 2012. [1]

But we can use "evaluation metrics" that aim to measure the quality of the clustering by assessing how well-separated the clusters are and how compact the clusters are.

- **Inertia** : can be considered as a measure of **intraclass variance** in the data.

  $inertia(C) = \sum_{j=1}^{K} \sum_{x_i \in C_j} ||x_i - centroid_j||^2$

- Silhouette score, The Calinski-Harabasz index etc.



1. Murphy, K. P. (2012). Machine learning : a probabilistic perspective. MIT press.

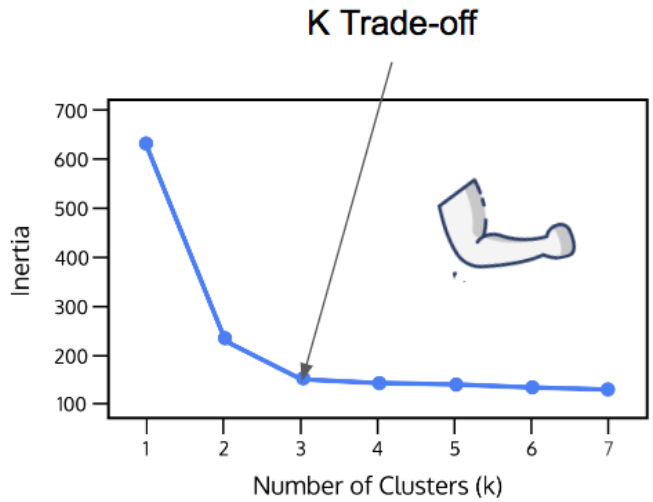
DETERMINING THE OPTIMAL NUMBER OF CLUSTERS

Choosing the optimal number of clusters :

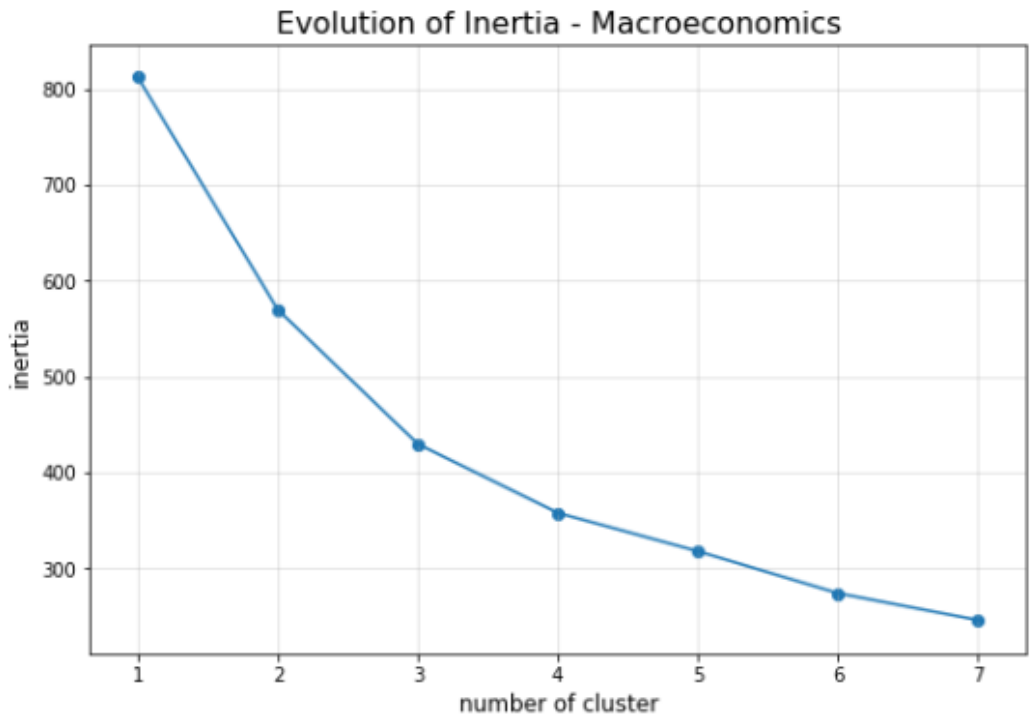
- Important to achieve meaningful results.
- But no theoretical method
- $\longrightarrow$ Empirical method : **"elbow method"**

Elbow method :

- Plot the within-cluster sum of squared errors (inertia) against the number of clusters.
- Find the "elbow" point where the decrease in inertia starts to level off.



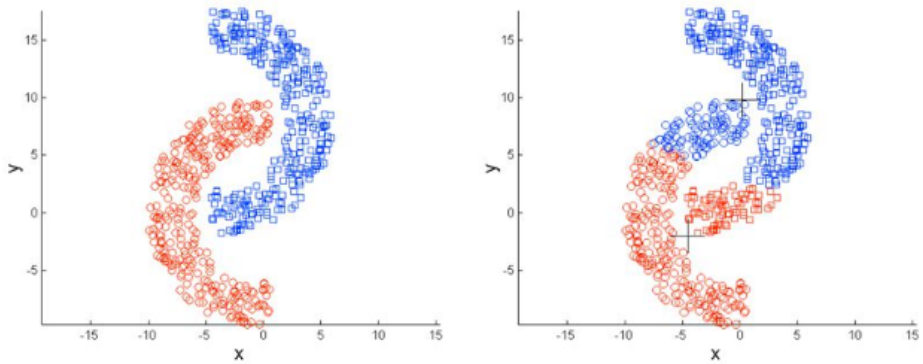
DETERMINING THE OPTIMAL NUMBER OF CLUSTERS



How can we use the graph to determine the optimal number of clusters ?

LIMITATIONS AND CHALLENGES

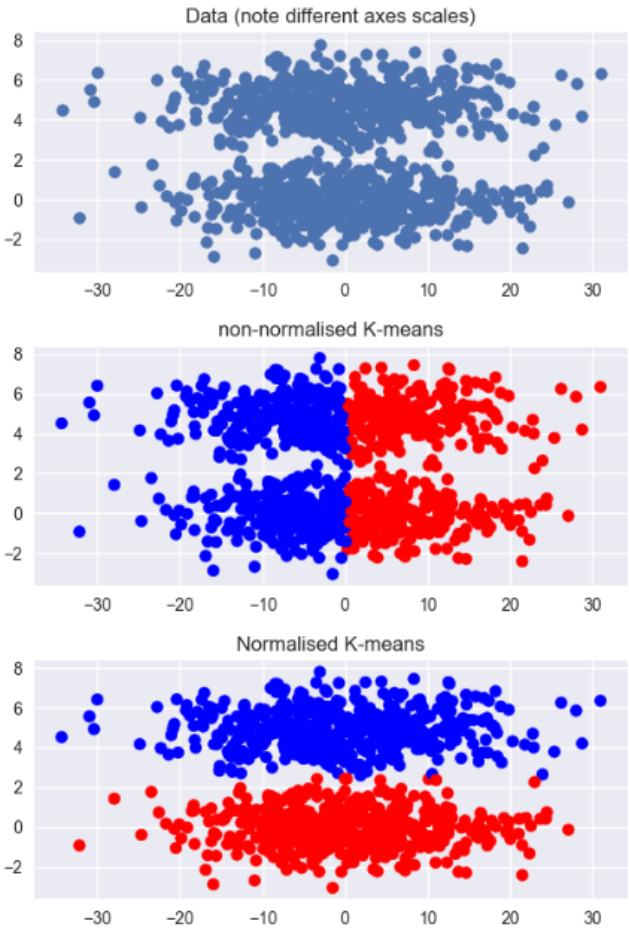
- **Sensitivity to initialization** : K-means is sensitive to the initial choice of centroids, which can lead to different clustering results.
- K-means assumes that **clusters have a circular shape**, which can lead to inaccurate clustering results when the true clusters have more complex shapes.
- **The curse of dimensionality** : As the number of variables in the dataset increases, K-means becomes less effective due to the increased sparsity of the data and the higher risk of overfitting.



DATA PREPARATION

- **Scaling and normalization of variables** : It is critical to ensure that variables are on a comparable scale and prevent variables with larger magnitudes from dominating the distance calculations.

- **Handling missing values and categorical variables** : K-means cannot handle missing values and categorical variables directly, and proper techniques (e.g. encoding) need to be applied to handle these data types.



K-MEANS PRACTICE

Scikit-Learn provides a user-friendly implementation of K-means.

In practice :

```
from sklearn.cluster import KMeans
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters)
kmeans.fit(X)

centroids = kmeans.cluster_centers_ # access to centroids
inertia = kmeans.inertia_ # access to inertia of the fitted model
```

APPLICATION TO THE MACROECONOMIC DATASET

In practice :

```
k_values = np.arange(1, 8, 1)
inertia = []
for k in k_values:
     kmeans = KMeans(n_clusters=k)
     kmeans.fit(X)
     inertia.append(kmeans.inertia_)
```