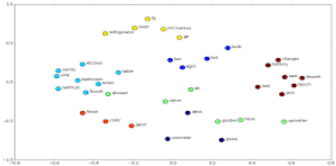
# Dimension reduction with PCA and t-SNE

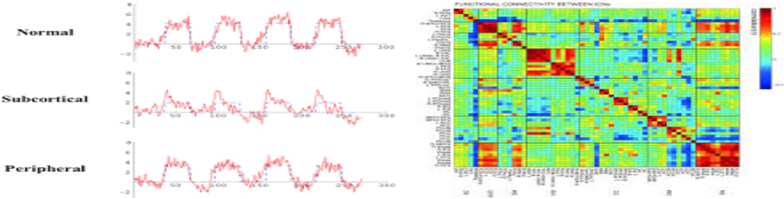
# High dimensional data in data analysis?

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ........ | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Apple | 0.9898 | 0.7865 | 0.5645 | 0.7509 | 0.4534 | 0.5467 | 0.6498 | 0.7613 | 0.8 |
| 2 | Banana | 0.4533 | 0.8644 | 0.1538 | 0.4313 | 0.3511 | 0.2422 | 0.2422 | 0.3553 | 0.2 |
| 3 | Cat | 0.8734 | 0.8363 | 0.4821 | 0.1378 | 0.2341 | 0.2122 | 0.6775 | 0.3432 | 0. |
| 4 | Dog | 0.9873 | 0.4836 | 0.1342 | 0.19564 | 0.2131 | 0.3433 | 0.2244 | 0.7453 | 0.5 |
| 5 | Eag | 0.9473 | 0.4836 | 0.4343 | 0.9211 | 0.1221 | 0.4634 | 0.7464 | 0.2424 | 0.5 |
| 6 | Google | 0.7634 | 0.4836 | 0.1313 | 0.1344 | 0.1232 | 0.6222 | 0.6564 | 0.3522 | 0.3 |
| 7 | Home | 0.8463 | 0.9732 | 0.4411 | 0.1333 | 0.6453 | 0.3435 | 0.3535 | 0.2442 | 0.3 |
| ...... | . | 0.8653 | 0.4835 | 0.1343 | 0.4421 | 0.7567 | 0.2424 | 0.5241 | 0.3221 | 0.3 |
| 00 | Zoo | 0.4736 | 0.9473 | 0.1453 | 0.1134 | 0.6564 | 0.1749 | 0.1892 | 0.1344 | 0.3 |



Words embeddings in NLP

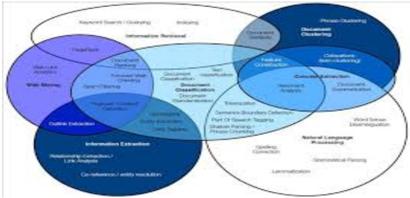
# High dimensional data in data analysis?

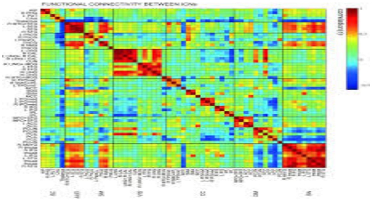



Brain activity

# High dimensional data in data analysis?

Challenges ?

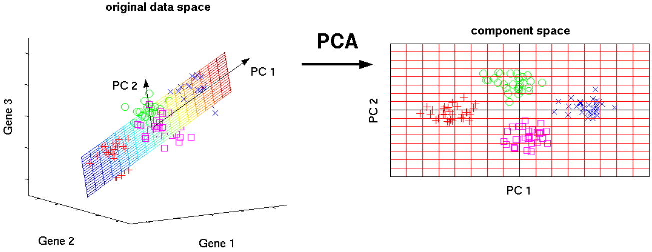


# High dimensional data in data analysis?

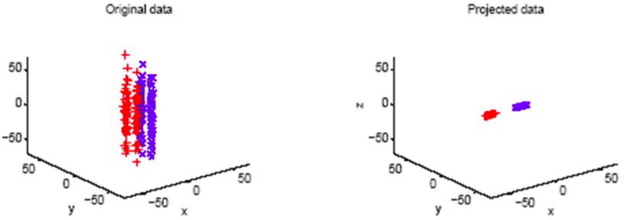
- Challenges ?
  - Visualize
  - Group in relevant clusters
- Difficult with high dimensional data!
- A classical dimension reduction approach Principal Component Analysis

# Dimension reduction?




Dimension reduction without loss of information?

# Dimension reduction




Dimension reduction without loss of information?

# Dimension reduction

**How can we reduce dimension to separate observations?**

Two examples in this lecture

- A linear approach : Principal Component Analysis (PCA)
- A non linear alternative : t-distributed stochastic neighbourhood embedding (t-SNE)

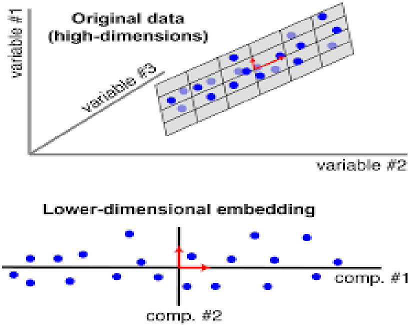
# Dimension reduction

## PCA vs t-SNE?

- Principal Component analysis (PCA)
  - preserves the global structure of data.
  - maps all the clusters as a whole
  - potential applications : noise filtering, feature extractions, stock market predictions, and gene data analysis.
- t-distributed stochastic neighbourhood embedding (t-SNE)
  - preserves the local structure of data
  - potential applications : music analysis, bioinformatics, and biomedical signal processing

# Principal Component Analysis

Principle : find a linear projection on a low-dimensional space




How can we find the low-dimensional space $H$?

# Principal Component Analysis

## Practical examples with Python

Let us consider a data set describing tree kinds of leafs coming from the website : `https://archive.ics.uci.edu/ml/datasets/Leaf`

### A toy example

- Leaf data set : describe tree kinds of leafs coming from the website : `https://archive.ics.uci.edu/ml/datasets/Leaf`
  Focus on the two following variables
  - Elongation : maximal normalized distance between a point of the leaf and its boundary
  - Isoperimetric factor : ratio between the area and the square of the perimeter of the leaf

# Principal Component Analysis

## Practical examples with Python

### Leaf Data Set

*Download*: Data Folder, Data Set Description

**Abstract**: This dataset consists in a collection of shape and texture features extracted from digital images of leaf specimens originating from a total of 40 different plant species.

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 340 | **Area:** | Computer |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 16 | **Date Donated** | 2014-02-24 |
| **Associated Tasks:** | Classification | **Missing Values?** | N/A | **Number of Web Hits:** | 88647 |

**Source:**

This dataset was created by Pedro F. B. Silva and AndrÃ© R. S. MarÃ§al using leaf specimens collected by Rubim Almeida da Silva at the Faculty of Science, University of Porto, Portugal.

**Data Set Information:**

For further details on this dataset and/or its attributes, please read the 'ReadMe.pdf' file included and/or consult the Master's Thesis 'Development of a System for Automatic Plant Species Recognition' available at [Web Link].

**Attribute Information:**

1. Class (Species)
2. Specimen Number
3. Eccentricity
4. Aspect Ratio
5. Elongation
6. Solidity
7. Stochastic Convexity
8. Isoperimetric Factor
9. Maximal Indentation Depth
10. Lobedness
11. Average Intensity
12. Average Contrast
13. Smoothness
14. Third moment
15. Uniformity
16. Entropy

# Principal Component Analysis

## Practical examples with Python

```
import numpy as np
leaf =
np.loadtxt('/home/marianne/Downloads/leaf.csv',
delimiter=',')
import pandas as pd
df_leaf=pd.DataFrame(np.array([leaf[:,4],leaf[:,7]]).T,colu
'Isoperimetric factor'])
df_leaf
```

# Principal Component Analysis

## Practical examples with Python

| | Elongation | Isoperimetric factor |
|---|---|---|
| **0** | 0.32396 | 0.835920 |
| **1** | 0.36116 | 0.798670 |
| **2** | 0.38998 | 0.808120 |
| **3** | 0.35376 | 0.816970 |
| **4** | 0.44462 | 0.754930 |
| **...** | ... | ... |
| **335** | 0.81725 | 0.125230 |
| **336** | 0.75319 | 0.136860 |
| **337** | 0.78147 | 0.135030 |
| **338** | 0.71532 | 0.157470 |
| **339** | 0.85409 | 0.078376 |

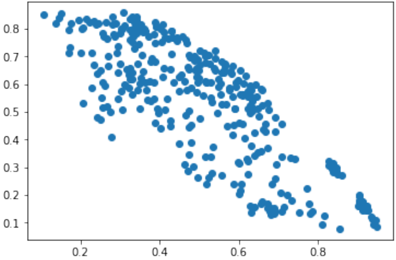
# Principal Component Analysis

## Practical examples with Python

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.scatter(leaf[:,4],leaf[:,7])
plt.show()
```

# Principal Component Analysis

## Practical examples with Python




The Leaf data set

# Principal Component Analysis

## Practical examples with Python

### Some questions

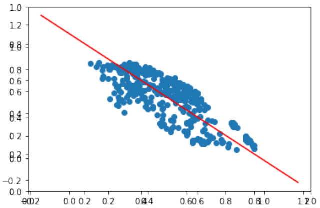
- How can we summarize properly using only one variable the information in this data set?
- Which variable allows to separate in the best possible way the data?
- Can we find an orientation along which the variance of the data is much higher ?

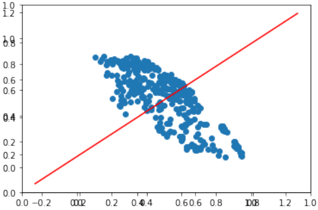
# Principal Component Analysis

## Practical examples with Python

Several possibilities....

...the axis of the figure on the left seems to be the best one!





# Principal Component Analysis

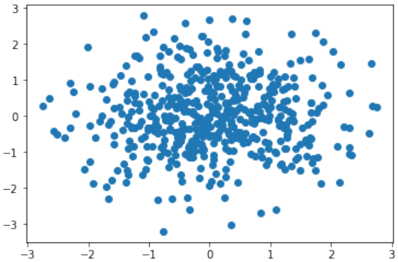
## Practical examples with Python

We try with a synthetic dataset!

```
rndn = np.random.randn(500,2)
fig, ax = plt.subplots()
ax.scatter(rndn[:,0],rndn[:,1])
plt.show()
```

# Principal Component Analysis

## Practical examples with Python

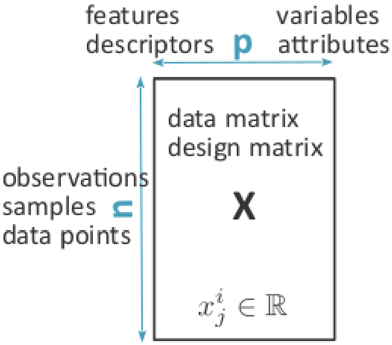



Not always possible to find an axis separating properly the data!

# Principal Component Analysis

## Principle

How summarize information in a large dataset ($n$ large) living in an high dimensional space ($p$ large)?



# Principal Component Analysis

## Principle

### Principal Component Analysis : how does it work?

- Denote $X$ the $n \times p$ data matrix or design matrix
- We want to find specific directions maximizing the variability of the data which is summarized in the covariance matrix $X^T X$
- The $k$ dimensional space $H$ that we are looking for is generated by the $k$ eigenvectors $u_i$, $i = 1, \cdots, k$ associated to the $k$ largest eigenvalues $\lambda_i$ of the matrix $X^T X$

# Principal Component Analysis

## Principle

### Principal component analysis : how does it work?

- The eigenvectors $u_1, \cdots, u_k$ are called the $k$ principal components.
- The eigenvalues $\lambda_1 \geq \cdots \geq \lambda_k$ are the explained variance ratio corresponding to each principal component
- By definition, the $k$ top principal components contain higher variance from the data.
- PCA can then be used as filtering, by selecting only the top significant PCs

# Principal Component Analysis

## Principle

### Principal component analysis : how does it work?

Several possible choices for the matrix $X$ :

- General PCA : the raw data matrix $X = R$
- Centered PCA: the centered data matrix. the matrix $X^T X$ is then the matrix of empirical covariances
- Normed PCA : the normed and centered data matrix. The matrix $X^T X$ is then the matrix of empirical correlations

# Principal Component Analysis

## Principle

### Principal component analysis : how does it work?

- In general $n \gg p$ (number of observations $\gg$ number of initial variables)
- It is the reason why we deal with the matrix $X^T X$ with dimension $p \times p$ rather than $XX^T$ with dimension $n \times n$
- These two analysis can both be deduced from the Singular Value Decomposition of $X$

# Principal Component Analysis

## Practical examples with Python

We import the data from the website : `https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data`

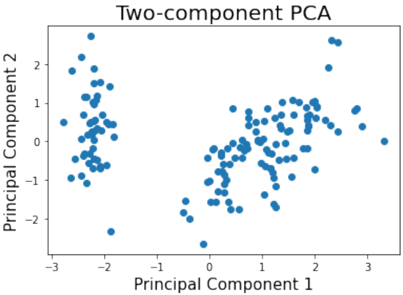
| | sepal length | sepal width | petal length | petal width | target |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 5 columns

# Principal Component Analysis

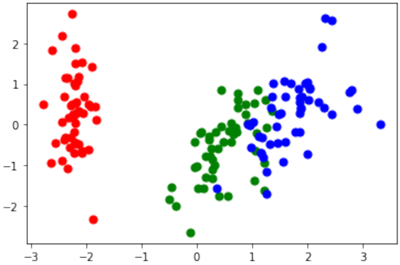
## Practical examples with Python

Application of a two components PCA and visualization



# Principal Component Analysis

## Link with different species?



# Principal Component Analysis

## Pro and cons of PCA

### Main advantages of PCA

- Simple to implement, no tuning
- Highly interpretable. We can find decide on how much variance to preserve using eigenvalues.

### Main drawbacks of PCA

- It is a global transform which may not preserve local structure (clusters)
- It is sensitive to outliers

An alternative : t-distributed stochastic neighbourhood embedding (t-SNE)

# t-distributed stochastic neighbourhood embedding (t-SNE)

More on t-SNE : `https://lvdmaaten.github.io/tsne/`

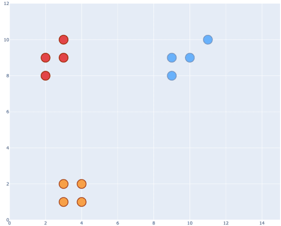
## Principle of SNE (Hinton 2002)

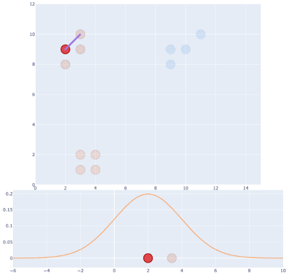
- Core idea : define an embedding from a high dimensional space to a low dimensional one, so that neighborhood identities are preserved.
- Similarity in the high dimensional space of two observations $x_j$ to $x_i$ is conditional probability $p_{j|i}$ that $x_i$ would pick $x_j$ as its nearest neighbor
- This conditional probability $p_{\cdot|i}$ is a Gaussian and depends on the (known) relative positions of the observations in the original space

# t-distributed stochastic neighbourhood embedding (t-SNE)

More on t-SNE : `https://lvdmaaten.github.io/tsne/`

Definition of a probability $p_{\cdot|i}$ associated to each observation $x_i$





# t-distributed stochastic neighbourhood embedding (t-SNE)

More on t-SNE : `https://lvdmaaten.github.io/tsne/`

## The SNE mapping (Hinton 2002)?

- $y_i$ : counterpart of $x_i$ in the low dimensional space, $q_{j|i}$ : conditional probability that $y_i$ would pick $y_j$ as its nearest neighbor
- If the map points $y_i$ and $y_j$ correctly model the similarity between the high-dimensional data points $x_i$ and $x_j$, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal, or at least close to each other
- Goal of SNE : find a low-dimensional data representation that minimizes the mismatch between $p_{j|i}$ and $q_{j|i}$ minimizing

$$\sum_i KL(p_{\cdot|i}, q_{\cdot|i}) \text{ (KL Kullback Divergence)}$$

# t-distributed stochastic neighbourhood embedding (t-SNE)

More on t-SNE : `https://lvdmaaten.github.io/tsne/`

- Leads to an optimization problem quite difficult to solve (in particular non symmetric)
- Introduction of t-SNE. Two main differences with SNE
  - uses a symmetrized version of the SNE cost function
  - uses a Student-t distribution rather than a Gaussian to compute the similarity between two points in the low-dimensional space

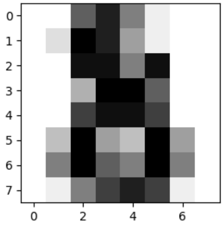
# t-distributed stochastic neighbourhood embedding (t-SNE)

## Practical examples with Python : comparison of PCA and t-SNE on the dataset digits

### The Digit Dataset

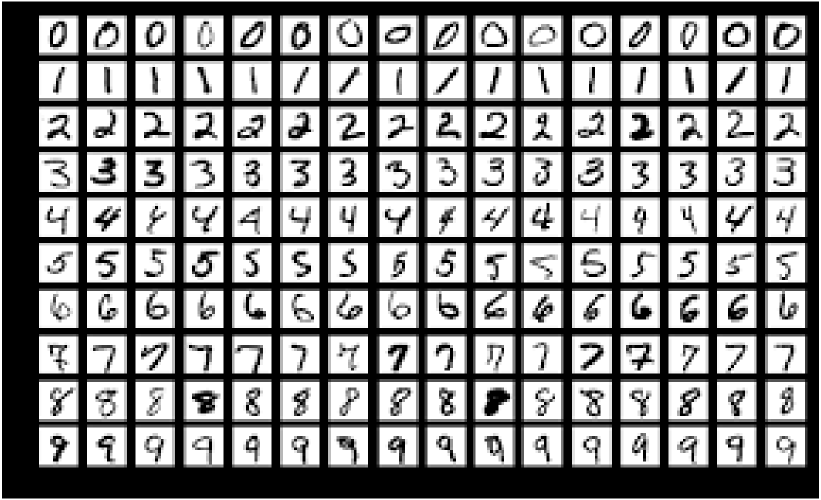
This dataset is made up of 1797 8x8 images. Each image, like the one shown below, is of a hand-written digit. In order to utilize an 8x8 figure like this, we'd have to first transform it into a feature vector with length 64.

See here for more information about this dataset.



# t-distributed stochastic neighbourhood embedding (t-SNE)

## Practical examples with Python : comparison of PCA and t-SNE on the dataset digits



# t-distributed stochastic neighbourhood embedding (t-SNE)

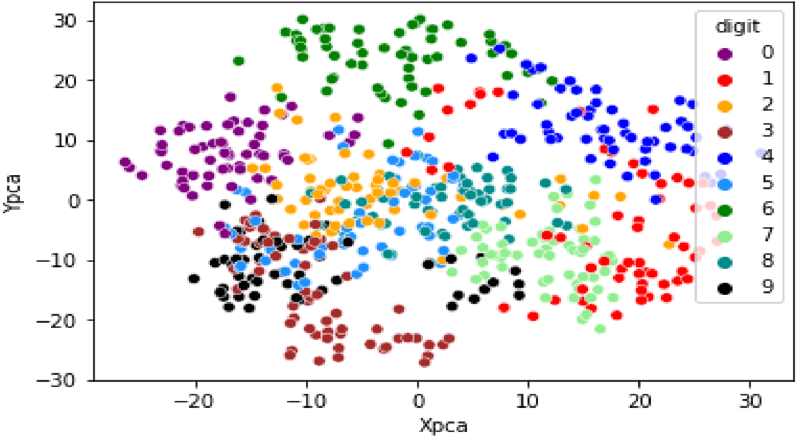
## Practical examples with Python : comparison of PCA and t-SNE on the dataset digits

We now compare

- Two dimensional PCA
- t-SNE embedding in a two-dimensional space

# t-distributed stochastic neighbourhood embedding (t-SNE)

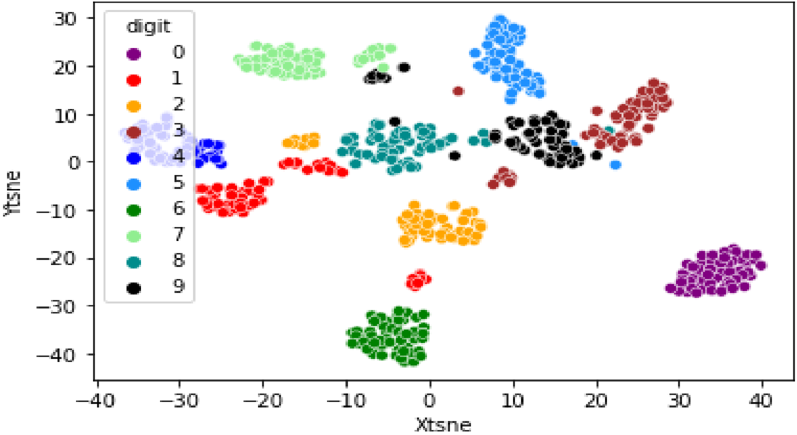
## Practical examples with Python : comparison of PCA and t-SNE on the dataset digits




The PCA answer

# t-distributed stochastic neighbourhood embedding (t-SNE)

## Practical examples with Python : comparison of PCA and t-SNE on the dataset digits




The t-SNE answer

# t-distributed stochastic neighbourhood embedding (t-SNE)

## Pro and cons of t-SNE

### Main advantages of t-SNE

- It tries to preserve the local structure(cluster) of data.
- It is one of the best dimensionality reduction technique
- It can handle outliers.

### Main drawbacks of t-SNE

- It is not deterministic and iterative so each time it runs, it could produce a different result.
- It is long to run compared to PCA
- It involves hyperparameters to tune.