Introduction to ML and NLP

Day 3

Sophie Robert-Hayek

University of Lorraine

Py4SHS 2023

Bibliography

- *Artificial Intelligence: A Modern Approach*, Russel and Norvig, Global Edition, 2014.
- *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python First Edition*, Hobson Lane, Hannes Hapke, Cole Howard, *Manning publishing*, 2019.
- Scikit-Learn documentation: `https://scikit-learn.org/stable/index.html`
- NLTK documentation: `https://www.nltk.org/`
- SpaCy documentation: `https://spacy.io/`

Reminders: Datasets and variables

Outline

Datasets

Question

Can anyone remind me what is a variable ?

Datasets

Question

Can anyone remind me what is a variable ?

Question

Can anyone remind me what is a feature ?

Datasets

Question

Can anyone remind me what is a variable ?

Question

Can anyone remind me what is a feature ?

Question

Can anyone reming me what is a dataset ?

Variable types

Question

Can anyone list the different types of variables that can be encountered in datasets ?

Variable types

Question

Can anyone list the different types of variables that can be encountered in datasets ?

Question

Can anyone list the different indicators that can be computed per variable type?

An introduction to Machine Learning

Outline

Why build datasets ?

Yesterday, we studied **structured datasets** to **describe** them.

Why build datasets ?

Yesterday, we studied **structured datasets** to **describe** them.
Today, we want to **get deeper in data analysis** in order to:

- **Derive** some new knowledge on the data;
- Use this data to **generate** new knowledge.

Why build datasets ?

Yesterday, we studied **structured datasets** to **describe** them.
Today, we want to **get deeper in data analysis** in order to:

- **Derive** some new knowledge on the data;
- Use this data to **generate** new knowledge.

We want the machine to **learn** on our dataset in order to recognize patterns, extract meaningful insights, and make informed decisions.

What is Machine Learning ?

What is Machine Learning ?

Machine Learning algorithms

Algorithms able to **learn** and **adapt** without following explicit instructions by **drawing inferences from patterns in data**.

Given a **training** dataset, Machine Learning* algorithms are able to **find patterns in data** to **predict** or **infer** information on new data.

What is Machine Learning ?

Machine Learning algorithms

Algorithms able to **learn** and **adapt** without following explicit instructions by **drawing inferences from patterns in data**.

Given a **training** dataset, Machine Learning* algorithms are able to **find patterns in data** to **predict** or **infer** information on new data.

Question

Can you give me some examples of Machine Learning models you know ?

What is Machine Learning ?

Problems can usually be divided into two main types:

What is Machine Learning ?

Problems can usually be divided into two main types:

What is Machine Learning ?

Problems can usually be divided into two main types:

- **Supervised learning***: the algorithm **should learn** from " example data" to predict the value for **unseen data**.
    - **Classification* problems**: a **class** (**categorical variable***) is predicted
    - **Regression* problems**: a **metric** (**numerical variable***) is predicted

What is Machine Learning ?

Problems can usually be divided into two main types:

- **Supervised learning***: the algorithm **should learn** from " example data" to predict the value for **unseen data**.
  - **Classification* problems**: a **class** (**categorical variable***) is predicted
  - **Regression* problems**: a **metric** (**numerical variable***) is predicted
- **Unsupervised learning***: the algorithm **should find some patterns in the data** to provide a better understanding of the data.

What is Machine Learning ?

Problems can usually be divided into two main types:

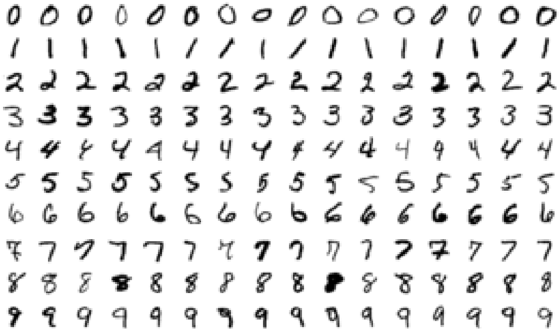
- **Supervised learning***: the algorithm **should learn** from " example data" to predict the value for **unseen data**.
    - **Classification* problems**: a **class** (**categorical variable***) is predicted
    - **Regression* problems**: a **metric** (**numerical variable***) is predicted
- **Unsupervised learning***: the algorithm **should find some patterns in the data** to provide a better understanding of the data.

Today, we will use **unsupervised learning**, and tomorrow, **supervised**.

Example of classification: handwritten recognition

Handwritten number recognition (OCR)

Given pixel repartition, learn to match handwritten numbers with their true value.



Example of regression: prediction of manuscript dating

Manuscript dating

Given linguistic features and manuscript physical characteristics, predict the age of the manuscript.

Example of clustering: finding sources within text

Infer different sources within text

Given the stylometry features of the first book of Genesis, identify if there are different sources across the text.

Can you infer the type of problem ?

Given a set of literary work written by Rowley and Sharkespeare, identify if the *Birth of Merlin* was written by Shakespeare or by Rowley.

Can you infer the type of problem ?

Given a very large database of literary works, identify books with common themes for easier storage.

Can you infer the type of problem ?

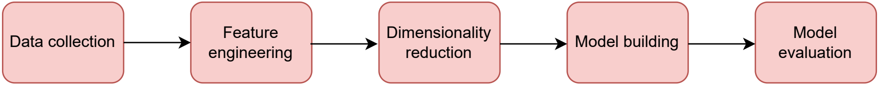
Given a hand-written manuscript, numerize its content.

Can you infer the type of problem ?

Given a database of works claimed to be by Augustine, identify if the *Sermones ad fratres in eremo* can be considered to be written by him.

Steps of a Machine Learning pipeline

Most pipelines can be summarized into 5 steps:



Step 1: Data collection

Data collection

Data collection* consists in collecting **manually** or **automatically** a set of information regarding the problem to solve.

Step 1: Data collection

Data collection

Data collection* consists in collecting **manually** or **automatically** a set of information regarding the problem to solve.

- **Data is the foundation of machine learning**.
- High-quality data is essential for training accurate and robust models: ***poor data leads to poor results***.

Step 1: Data collection

Data can be either:

- **Structured Data**: Well-organized data in rows and columns (e.g., SQL database, flat files ...).
- **Unstructured Data**: No predefined structure (e.g., text, images, audio, video).
- **Semi-Structured Data**: Some structure, but not as rigid as structured data (e.g., JSON, XML, NoSQL database ...).

Step 2: Feature engineering

Features

Features are **the input variables** that machine learning models use to make predictions, to encode the relevant information from the data.

Step 2: Feature engineering

Features

Features are **the input variables** that machine learning models use to make predictions, to encode the relevant information from the data.

Raw data is usually not suitable for direct use:

- Because it can be **unstructured** (text, images, categories ...).
- Because it lacks **information** to capture patterns and relationships effectively.

Step 2: Feature engineering

Examples of feature engineering include:

- **Feature Extraction**: Transforming unstructured data into structured data.

Step 2: Feature engineering

Examples of feature engineering include:

- **Feature Extraction**: Transforming unstructured data into structured data.
- **Feature Scaling**: Normalizing features to ensure consistent scales.

Step 2: Feature engineering

Examples of feature engineering include:

- **Feature Extraction**: Transforming unstructured data into structured data.
- **Feature Scaling**: Normalizing features to ensure consistent scales.
- **Encoding Categorical Data**: Transforming categorical variables into numerical form.

Step 2: Feature engineering

Examples of feature engineering include:

- **Feature Extraction**: Transforming unstructured data into structured data.
- **Feature Scaling**: Normalizing features to ensure consistent scales.
- **Encoding Categorical Data**: Transforming categorical variables into numerical form.
- **Creating Interaction Features**: Combining features to capture interactions, either using domain based information or automatic methods.

Step 2: Feature engineering

Examples of feature engineering include:

- **Feature Extraction**: Transforming unstructured data into structured data.
- **Feature Scaling**: Normalizing features to ensure consistent scales.
- **Encoding Categorical Data**: Transforming categorical variables into numerical form.
- **Creating Interaction Features**: Combining features to capture interactions, either using domain based information or automatic methods.

Question

Feature engineering is **THE** challenge of NLP.
Does anyone have an idea on how we can structure text ?

Why do we need to reduce dimension ?

Question

Why do you think we need to reduce dimensions ?

Why do we need to reduce dimension ?

Question

Why do you think we need to reduce dimensions ?

- Computation time;

Why do we need to reduce dimension ?

Question

Why do you think we need to reduce dimensions ?

- Computation time;
- Easier data visualization;

Why do we need to reduce dimension ?

Question

Why do you think we need to reduce dimensions ?

- Computation time;
- Easier data visualization;
- Possible unrelated features acting as noise;

Why do we need to reduce dimension ?

Question

Why do you think we need to reduce dimensions ?

- Computation time;
- Easier data visualization;
- Possible unrelated features acting as noise;
- Possible correlated features that do not bring any new information to solve the task;

Why do we need to reduce dimension ?

Question

Why do you think we need to reduce dimensions ?

- Computation time;
- Easier data visualization;
- Possible unrelated features acting as noise;
- Possible correlated features that do not bring any new information to solve the task;
- **The curse of dimensionality**.

Possible approaches

Question

What is in your opinion possible approaches to reduce the number of features ?

Possible approaches

Question

What is in your opinion possible approaches to reduce the number of features ?

- **Removing** some features (forward/backward selection ...)

Possible approaches

Question

What is in your opinion possible approaches to reduce the number of features ?

- **Removing** some features (forward/backward selection ...)
- **Projecting the features** into a lower dimensional space (PCA, tSNE, UMAP, NN embeddings ...)

Today, we will work with PCA which I will explain during the lab session.

Model fitting/training

Model training

Training a model on a dataset consist in finding the parameters of an algorithm that optimize the prediction power of this algorithm on this input dataset.

Workflow usually consists in:

- Selecting a class of model.
- Fitting the model on the dataset.
- Evaluating the quality of the model on unseen data.

Supervised models

Can you give examples of class of supervised models ?

Supervised models

Can you give examples of class of supervised models ?

Example of supervised models:

- K-nearest neighbors
- Neural Networks
- Decision trees and random forests
- Support Vector Machines ...

Unsupervised models

Can you give examples of class of unsupervised models ?

Unsupervised models

Can you give examples of class of unsupervised models ?

Example of supervised models:

- K-means
- DBScan
- Agglomerative clusterings ...

The k-means algorithm

Today, we will use a specific unsupervised learning algorithm called **k-means**.

The k-means algorithm

Today, we will use a specific unsupervised learning algorithm called **k-means**.

K-Means algorithm

The k-means algorithm* (*MacQueen, 1967*) is a clustering algorithm that partitions the space into k cluster by minimizing the *within-cluster variance*.

The k-means algorithm

Today, we will use a specific unsupervised learning algorithm called **k-means**.

K-Means algorithm

The k-means algorithm* (*MacQueen, 1967*) is a clustering algorithm that partitions the space into k cluster by minimizing the *within-cluster variance*.

Given a set of individuals described by their features $(X_1, \ldots, X_n)$ find k sets to partition the data into by minimzing *the within cluster variance*.

$$\sum_{i=1}^{k} \sum_{X \in S_i} = ||X - \mu_i||^2$$

with:

$$\mu_i = \frac{1}{|S_i|} \sum X$$

The k-means algorithm

In practice, problem is NP-hard, so we rely on **Lloyd's iterative algorithm**:

The k-means algorithm

In practice, problem is NP-hard, so we rely on **Lloyd's iterative algorithm**:
Given a set of k means $m_1^{(1)}, \ldots, m_k^{(1)}$, iteratively perform two steps:

The k-means algorithm

In practice, problem is NP-hard, so we rely on **Lloyd's iterative algorithm**:
Given a set of k means $m_1^{(1)}, \ldots, m_k^{(1)}$, iteratively perform two steps:

1. **Assignment step**: Assign each observation to the cluster with the nearest mean using the **Euclidean distance**.

The k-means algorithm

In practice, problem is NP-hard, so we rely on **Lloyd's iterative algorithm**:
Given a set of k means $m_1^{(1)}, \ldots, m_k^{(1)}$, iteratively perform two steps:

1. **Assignment step**: Assign each observation to the cluster with the nearest mean using the **Euclidean distance**.
2. **Update step**: Recalculate the mean for each cluster.

The k-means algorithm

In practice, problem is NP-hard, so we rely on **Lloyd's iterative algorithm**:
Given a set of k means $m_1^{(1)}, \ldots, m_k^{(1)}$, iteratively perform two steps:

1. **Assignment step**: Assign each observation to the cluster with the nearest mean using the **Euclidean distance**.
2. **Update step**: Recalculate the mean for each cluster.

Run steps until assignment do not change.

The k-means algorithm

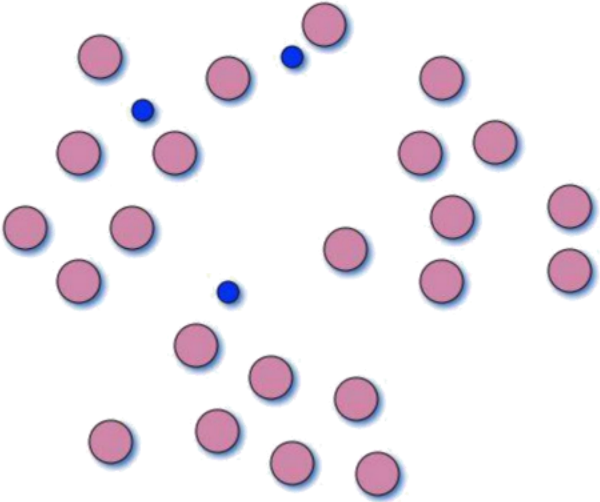
In practice, problem is NP-hard, so we rely on **Lloyd's iterative algorithm**:
Given a set of k means $m_1^{(1)}, \ldots, m_k^{(1)}$, iteratively perform two steps:

1. **Assignment step**: Assign each observation to the cluster with the nearest mean using the **Euclidean distance**.
2. **Update step**: Recalculate the mean for each cluster.

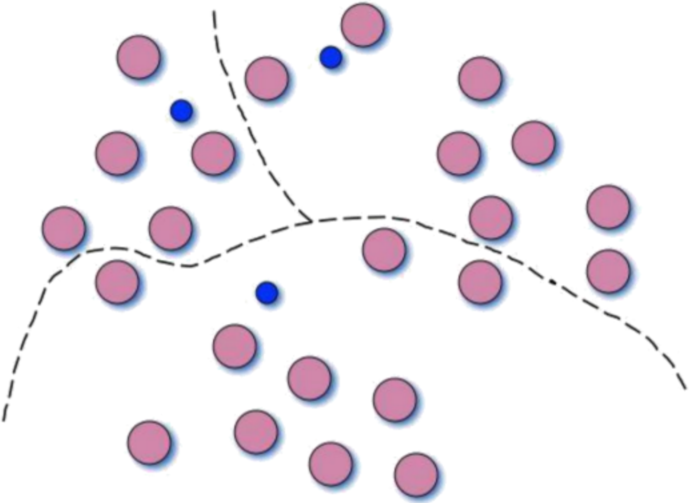
Run steps until assignment do not change.

There is no garantee to find the optimum (but efficient in practice).

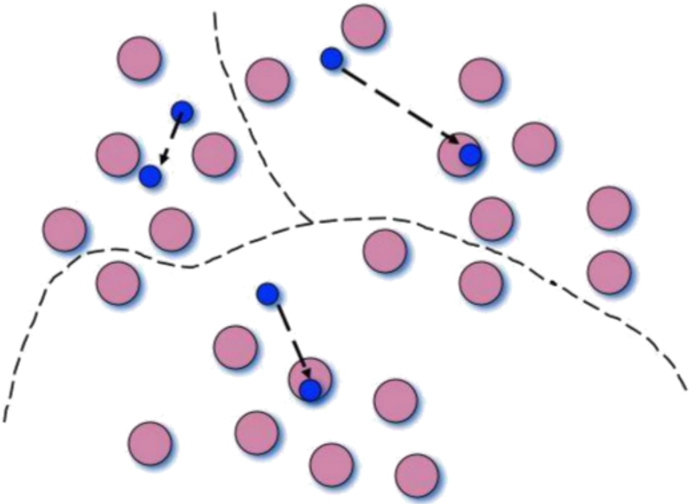
Initialization



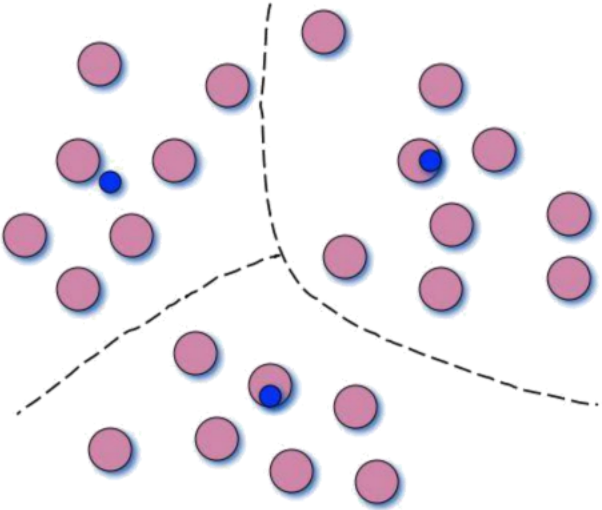
Assign each individual to a cluster



Compute new medoids



Repeat until stable



Hyperparameters optimization

Hyperparameters

Hyperparameters* are parameters that are not learned during training, but that affect the learning process.

Hyperparameters optimization

Hyperparameters

Hyperparameters* are parameters that are not learned during training, but that affect the learning process.

Examples of hyperparameters:

- The number of layers of the neural network
- The homogeneity measure in decision trees
- The number of clusters in the k-means algorithm

Strategies for Hyperparameter Optimization

- **Grid Search**: Exhaustively search a predefined set of hyperparameter combinations.
- **Random Search**: Randomly sample from a predefined distribution of hyperparameters.
- **Bayesian Optimization**: Use probability distributions to model the objective function and select promising hyperparameters.
- **Genetic Algorithms**: Evolve a population of hyperparameter combinations to find optimal values.

Step 5: Model evaluation

After fitting a Machine Learning model, its quality needs to be **evaluated**, on **seen** and **unseen** data.

Step 5: Model evaluation

After fitting a Machine Learning model, its quality needs to be **evaluated**, on **seen** and **unseen** data.

Model evaluation

Model evaluation consists in providing objective metrics able to quantify how well an algorithm perform on seen and unseen samples.

Step 5: Model evaluation

After fitting a Machine Learning model, its quality needs to be **evaluated**, on **seen** and **unseen** data.

Model evaluation

Model evaluation consists in providing objective metrics able to quantify how well an algorithm perform on seen and unseen samples.

Possible metrics include:

- For supervised learning: accuracy, precision, recall ...
- For unsupervised learning: silhouette score, Davies Boulin ...
- For regression: Mean Square Error ...

Tomorrow, we will learn more in detail what metric can be used in the case of supervised learning.

Evaluating k-means algorithms

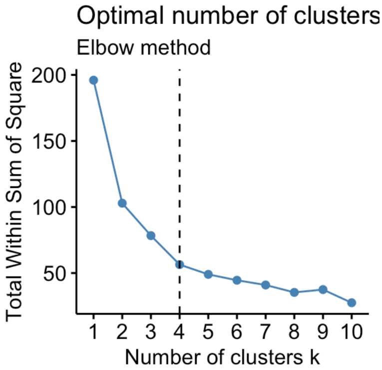
Elbow method

An elbow plot is a visual method by plotting the *within cluster variance* against the number of clusters and selecting the number of clusters before the curve flattens.

Evaluating k-means algorithms

Elbow method

An elbow plot is a visual method by plotting the *within cluster variance* against the number of clusters and selecting the number of clusters before the curve flattens.



Introduction to Natural Language Processing

Outline

Definition

Natural Language Processing

Natural Language Processing* (NLP) aims to enable computers to **comprehend, interpret, and interact with human language in a manner that is both meaningful and contextually appropriate**.

Definition

Natural Language Processing

Natural Language Processing* (NLP) aims to enable computers to **comprehend, interpret, and interact with human language in a manner that is both meaningful and contextually appropriate**.

It is an interdisciplinary field of study at the intersection of:

- linguistics
- artificial intelligence
- computer science

Major sub-fields

- **Language Analysis**: syntactic analysis, semantic role labeling, and entity recognition, which help identify the relationships between words, phrases, and concepts within a sentence.

Major sub-fields

- **Language Analysis**: syntactic analysis, semantic role labeling, and entity recognition, which help identify the relationships between words, phrases, and concepts within a sentence.
- **Language Generation**: generation of human-like language through machine learning models, such as text completion, summarization, text generation (think chatGPT !)

Major sub-fields

- **Language Analysis**: syntactic analysis, semantic role labeling, and entity recognition, which help identify the relationships between words, phrases, and concepts within a sentence.
- **Language Generation**: generation of human-like language through machine learning models, such as text completion, summarization, text generation (think chatGPT !)
- **Speech Recognition**: converting spoken language into written text (speech recognition) and converting text into spoken words (speech synthesis).

Major sub-fields

- **Language Analysis**: syntactic analysis, semantic role labeling, and entity recognition, which help identify the relationships between words, phrases, and concepts within a sentence.
- **Language Generation**: generation of human-like language through machine learning models, such as text completion, summarization, text generation (think chatGPT !)
- **Speech Recognition**: converting spoken language into written text (speech recognition) and converting text into spoken words (speech synthesis).
- **Machine translation**: conversion of text or speech from one language to another.

Example of applications in digital humanities

Can you give some examples of NLP based projects in digital humanities ?

Example of applications in digital humanities

Can you give some examples of NLP based projects in digital humanities ?

- Stylometry analysis for authorship detection.
- Ordering according to topics large datasets of documents.
- Finding similarity across authors.
- Analyzing scribal behavior through stemmatology.

Reference tools and bibliography

- Spacy: https://spacy.io/
- NLTK: https://www.nltk.org/
- GenSim: https://github.com/RaRe-Technologies/gensim
- Scikit-Learn: https://scikit-learn.org/stable/

Textual embeddings

Let's consider this dataset:

ID 1	I love Machine Learning
ID 2	I like Machine Learning
ID 3	I hate Machine Learning

Textual embeddings

Let's consider this dataset:

ID 1	I love Machine Learning
ID 2	I like Machine Learning
ID 3	I hate Machine Learning

Is this structured or unstructured data ?

Textual embeddings

Why can't we apply directly statistical analysis and Machine Learning algorithms to this data ?

Textual embeddings

Why can't we apply directly statistical analysis and Machine Learning algorithms to this data ?

... **Most algorithms require *quantitative data***... ! We need to project the textual data into a **numerical space**.

Text embedding

Text embedding

An embedding is a mapping of a non-quantitative variable to a vector of continuous numbers.
A **text embedding** is the transformation of textual data into a vector of continuous numbers.

Text embedding

Two possible approaches:

- **Frequency based**: the coordinates of the text represents the frequency or normalized frequency of the words found in the dataset.
- **Neural-network based**: the coordinates are automatically learned through a neural network.

Term Frequency (TF)

Term Frequency (TF)

Term Frequency* (TF) quantifies the importance of a term in a corpus, as the ratio of the number of occurrences of term t in a document d:

$$TF(t, d) = \frac{\text{Number of occurrences of term } t \text{ in document } d}{\text{Total number of terms in document } d}$$

Term Frequency (TF)

Term Frequency (TF)

Term Frequency* (TF) quantifies the importance of a term in a corpus, as the ratio of the number of occurrences of term t in a document d:

$$TF(t,d) = \frac{\text{Number of occurrences of term } t \text{ in document } d}{\text{Total number of terms in document } d}$$

Term Frequency (TF) measures how frequently a term t appears in document d, to **quantify the importance** of t in d.

Term frequency

Question

Build the term frequency matrix of the following corpora:

ID 1	I love Machine Learning
ID 2	I like Machine Learning
ID 3	I hate Machine Learning

Inverse Document Frequency

Inverse Document Frequency (IDF)

Inverse Document Frequency (IDF) quantifies the importance of term t in the entire collection of document, as the ratio of the total number of documents to the number of documents containing term t:

$$IDF(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right)$$

IDF measures the informativeness of term t across the entire document collection.

Inverse Document Frequency (IDF)

Question

- If t is present in every document, what is IDF equals to ?

Inverse Document Frequency (IDF)

Question

- If t is present in every document, what is IDF equals to ?
- If t is present in half documents, what is IDF equals to ?

Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF combines two metrics: Term Frequency (TF) and Inverse Document Frequency (IDF).

Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) identifies terms that are **frequent within a specific document** and **rare across the entire document collection**:

$$\mathit{TF\text{-}IDF}(t, d) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t)$$

High TF-IDF values indicates that a term is important in a particular document but relatively rare in the overall collection: **it gives more weight to relevant terms and reduces the impact of common words**.

Term frequency

Question

Build the term frequency matrix of the following corpora:

ID 1	I love Machine Learning
ID 2	I like Machine Learning
ID 3	I hate Machine Learning

Text embedding using neural networks

Sentence/Document Embeddings:

- **Sentence or document embeddings** are continuous vector representations of entire sentences or documents;
- These models **utilize recurrent neural networks (RNNs) or transformer-based architectures** to capture the semantic meaning of variable-length texts;
- Examples of such model include s-BERT and Doc2Vec.

You will learn more about the practicalities of Deep Learning in tomorrow's session ! **Bonus questions in today's lab will include Doc2Vec**.

Data cleaning

Given the previous embeddings, what do you see is a limitation of using "*raw*" words ?

Data cleaning

Given the previous embeddings, what do you see is a limitation of using "*raw*" words ?

In the previous example, *liking* and *like* are projected differently ...To mitigate this problem, one can use:

- Lemming
- Stemming
- Removal of stop words

Lemming and stemming

- **Lemmatization** and **Stemming** are techniques used to reduce words to their base or root form.
- They are essential preprocessing steps to **normalize and standardize textual data** before further analysis.
- Both methods aim to reduce inflected or derived words **to a common base form**, but they have different approaches and outcomes.

Stemming

Stemming

Stemming* is the process of removing suffixes or prefixes from words to **obtain the root or base form** of a word, called the **stem**, according to a heuristic rule.

Historical $\rightarrow$ *Histori*
History $\rightarrow$ *Histori*

Stemming

Advantages:

- Very fast to compute
- Does not require any knowledge of context or grammar

Stemming

Advantages:

- Very fast to compute
- Does not require any knowledge of context or grammar

Drawbacks:

- Does not create meaningful word
- Can stem similarly words that are different (*jumps* $\rightarrow$ *jump* and *jumper* $\rightarrow$ *jump*)

Lemming

Lemming

Lemming* reduces words to their **dictionary form**, called lemma. It requires lexical knowledge, context understanding, and morphological analysis to correctly identify the lemma.

jumps $\rightarrow$ verb $\rightarrow$ *jump*
jumper $\rightarrow$ noun $\rightarrow$ *jumper*

POS tagging

POS tagging

POS tagging consists in assigning grammatical categories representing the syntactic roles of words in a sentence (nouns, verbs, adjectives, adverbs, etc.).

POS tagging can be done:

- Using a **rule-based approach**: POS follows syntactic rules, which is prone to mistake.

POS tagging

POS tagging

POS tagging consists in assigning grammatical categories representing the syntactic roles of words in a sentence (nouns, verbs, adjectives, adverbs, etc.).

POS tagging can be done:

- Using a **rule-based approach**: POS follows syntactic rules, which is prone to mistake.
- Using **statistical approach** (HMM, RNN, etc): use machine learning algorithms to predict POS tags based on features like word context and neighboring tags.

Lemming

Advantages:

- More precise information

Drawbacks

- Computationally heavy
- Not available in every language (such as ancient languages)

Stop words removal

Even though IDF "naturally" cleans up frequent words, manually removing stop words speeds up the cleaning process.

Stop words removal

Stop words are common words that occur frequently in a language but usually do not carry significant meaning or information (such as "the", "a", "an", "in", "of", "and", "is", "it", etc).

Possible approaches include:

- **Stop word lists:** Remove predefined lists of stop words.
- **Frequency-based removal:** Remove words with a term-frequency above a threshold.
- **Contextual analysis:** Use Machine Learning methods to remove context-specific stop words.

Full pipeline example

Do the following exercise:

- Remove stop words (stop word list: *and*)
- Perform lemming
- Compute tf-idf

ID 1	Machine Learning is fun and interesting !
ID 2	Machine Learning interests me.
ID 3	Machine Learning is fun and I like it.

Questions ?

Questions ?

Lab session: clustering of Shakespeare's plays

Outline

Problem

Given a dataset containing a set of Shakespeare's play:

- Give the most frequent words globally and per play.
- Project the books into a lower dimensional space to understand similarity in terms of topic.
- Use k-means clustering to group books together.

Required packages

- Sklearn (must-have main Python library!)
- Spacy
- Pandas
- Matplotlib
- Seaborn

Task

You need to install the required packages using `pip`.

Scikit-learn

Sklearn

Scikit-learn, or sklearn, is a **must-know** machine learning library in Python, which provides a wide range of tools for various machine learning tasks, including classification, regression, clustering, and more.

Scikit-learn

Sklearn

Scikit-learn, or sklearn, is a **must-know** machine learning library in Python, which provides a wide range of tools for various machine learning tasks, including classification, regression, clustering, and more.

It provides:

- An easy to use standard API.
- Most Machine Learning algorithms.

Scikit-learn API: supervised learning

Simple Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Load data and split into training/testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a linear regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)
```

Scikit-learn API: unsupervised learning

K-Means Clustering

```python
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Generate sample data
X, _ = make_blobs(n_samples=300, centers=3, random_state=42)

# Create a K-Means model with 3 clusters
model = KMeans(n_clusters=3)

# Fit the model to the data
model.fit(X)

# Get the cluster assignments and centroids
labels = model.labels_
centroids = model.cluster_centers_
```

Scikit-learn API: PCA

Principal Component Analysis

```
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Create a PCA model with 2 components
pca = PCA(n_components=2)

# Fit and transform the data
X_pca = pca.fit_transform(X)
```

Principal Component Analysis

Principal Component Analysis

Principal Component analysis is a feature projection method that consists in finding a **new coordinate system as a linear combination of the input features** to project the data: this system is orthogonal and a linear combination of the features that **maximizes the variance**.

Principal Component Analysis

Principal Component Analysis

Principal Component analysis is a feature projection method that consists in finding a **new coordinate system as a linear combination of the input features** to project the data: this system is orthogonal and a linear combination of the features that **maximizes the variance**.

The goal of PCA is to:

Principal Component Analysis

Principal Component Analysis

Principal Component analysis is a feature projection method that consists in finding a **new coordinate system as a linear combination of the input features** to project the data: this system is orthogonal and a linear combination of the features that **maximizes the variance**.

The goal of PCA is to:

- Find new axis more relevant to represent the data

Principal Component Analysis

Principal Component Analysis

Principal Component analysis is a feature projection method that consists in finding a **new coordinate system as a linear combination of the input features** to project the data: this system is orthogonal and a linear combination of the features that **maximizes the variance**.

The goal of PCA is to:

- Find new axis more relevant to represent the data
- Give us the importance of each axis

Principal Component Analysis

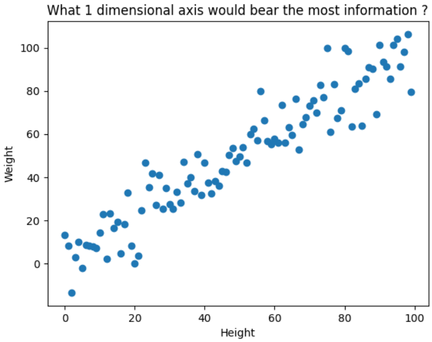
Principal Component Analysis

Principal Component analysis is a feature projection method that consists in finding a **new coordinate system as a linear combination of the input features** to project the data: this system is orthogonal and a linear combination of the features that **maximizes the variance**.

The goal of PCA is to:

- Find new axis more relevant to represent the data
- Give us the importance of each axis
- Remove unimportant axis and reduce dimension

Principle

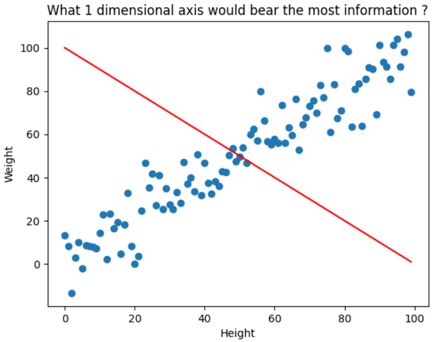


Question

What would be an adequate axis to project the data on to reduce to a single axis ?

Principle

Possibility: $y = -x$

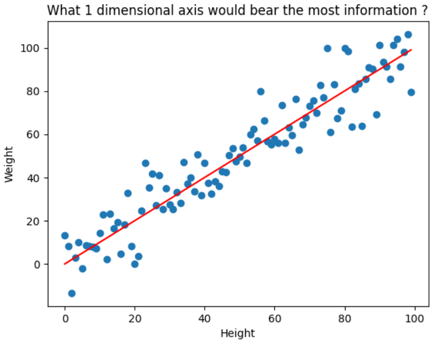


Question

What would a projection on this new axis look like ?

Principle

Possibility: $y = x$



Question

What would a projection on this new axis look like ?

Principle

We are looking for new axis that **maximize the variance** and are **uncorrelated**.

Principle

We are looking for new axis that **maximize the variance** and are **uncorrelated**.

It can be shown that these **principal components** are eigenvectors of the data's:

Principle

We are looking for new axis that **maximize the variance** and are **uncorrelated**.

It can be shown that these **principal components** are eigenvectors of the data's:

- Covariance matrix (**Centered PCA**)

Principle

We are looking for new axis that **maximize the variance** and are **uncorrelated**.

It can be shown that these **principal components** are eigenvectors of the data's:

- Covariance matrix (**Centered PCA**)
- Correlation matrix (**Normed PCA**)

Example dataset

The **Iris dataset** D:

sepal length (cm)	sepal width (cm)	petal length (cm)
5.1	3.5	1.4
4.9	3.0	1.4
4.7	3.2	1.3
4.6	3.1	1.5
5.0	3.6	1.4
5.4	3.9	1.7
4.6	3.4	1.4
5.0	3.4	1.5
4.4	2.9	1.4
4.9	3.1	1.5

Standardize matrix

For **centered PCA**, center matrix.
For **normed PCA**, standardize matrix (remove mean and divide by standard error): Z.

Compute the correlation matrix

Compute correlation matrix C.

With Z the standardized matrix and n the number of individuals within the dataset,
$C = \frac{1}{n} Z^t Z$

	Sepal length	width	Petal length
Sepal length	1.00	0.79	0.60
width	0.79	1.00	0.52
Petal length	0.60	0.52	1.00

Find eigenvalues and eigenvectors

Eigen values (sorted) of the correlation matrix are:
$[2.27, 0.51, 0.20]$

Eigen vectors matrix P (sorted by eigen values) is:

Feature	**Component 0**	**Component 1**	**Component 2**
S. length	0.61	-0.26	0.75
Width	0.59	-0.48	-0.65
P. length	0.53	0.84	-0.14

Project matrix into new space

Multiply the standardized matrix Z by the eigenvectors to have the matrix Z^* in the new projected space $Z^* = ZP$: **this is the projection of the individuals in the new feature space**.

ID	Component 0	Component 1	Component 2
0	0.66	-0.95	0.30
1	-0.80	0.07	0.87
2	-1.35	-0.90	0.02
3	-0.74	1.00	-0.31
4	0.64	-1.02	-0.20
5	3.68	0.57	-0.20
6	-0.65	-0.31	-0.83
7	0.75	0.13	0.11
8	-2.11	0.70	-0.26
9	-0.08	0.72	0.51

Select number of axis

Axis importance

The value of each eigenvalue is the **importance of the axis** and is used to select the number of axis to keep as a percentage of the explained variance.

Select number of axis

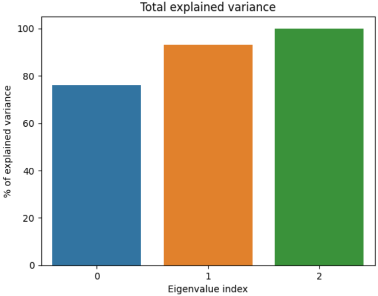
Axis importance

The value of each eigenvalue is the **importance of the axis** and is used to select the number of axis to keep as a percentage of the explained variance.

Usually, compute the cumulated sum of $\frac{\lambda_i}{\sum_{i=0}^{n} \lambda_i}$ for the i-th eigenvalue λ_i: the selected value is the **total explained variance of the dataset**.

Select number of axis

Here the cumulated sum is: $[0.76, 0.93, 1]$



We select **two axis**.

Final projection

We have our final projection for each individual within the reduced space:

ID	Component 0	Component 1
0	0.66	-0.95
1	-0.80	0.07
2	-1.35	-0.90
3	-0.74	1.00
4	0.64	-1.02
5	3.68	0.57
6	-0.65	-0.31
7	0.75	0.13
8	-2.11	0.70
9	-0.08	0.72

Interpret results

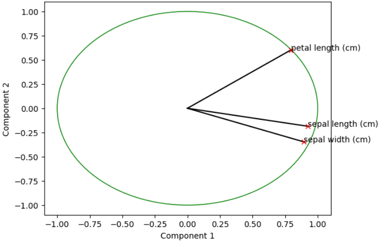
A possible interpretation is the plot of the **correlation circle**.

Interpret results

A possible interpretation is the plot of the **correlation circle**.

Correlation circle

The correlation circle consists in computing **the correlation of the original features with the new components** and deducing from it the contribution of each variable to the axis.



Advantages and drawbacks

Advantages

Advantages and drawbacks

Advantages

- Simple to implement

Advantages and drawbacks

Advantages

- Simple to implement
- Simple to evaluate data loss because of transformation

Advantages and drawbacks

Advantages

- Simple to implement
- Simple to evaluate data loss because of transformation

Drawbacks

- Features are transformed and lose interpretability of some results

Advantages and drawbacks

Advantages

- Simple to implement
- Simple to evaluate data loss because of transformation

Drawbacks

- Features are transformed and lose interpretability of some results
- Sensitive to outliers