# SI221/MICAS911 Practical assignment #6: $k$-Nearest Neighbors and $k$-Means Clustering- October 2021

## Instructions

Read all the instructions below carefully before you start working on the assignment.

- Please submit the source code and a pdf report of your work. You can use the Jupyter notebook instead, by submitting the ipynb file. Each file must have as title: *TP_kNN_kMeans_Student1_Student2*.
- Late assignments will not be corrected.
- You must do this assignment in groups of 2. Please submit no more than one submission per group.
- You must implement all algorithms from scratch.
- Code that does not work will not be considered.
- Send your work to: `arthur.louchart@telecom-paris.fr`

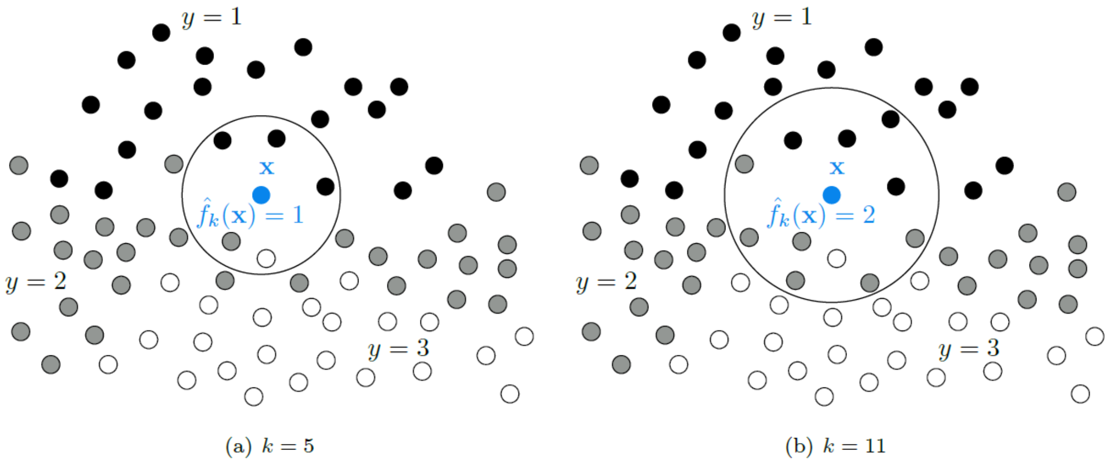
## Practical assignment objectives

- To implement the $k$-Nearest Neighbors algorithm from scratch in order to classify and predict data.
- To implement the $k$-Means algorithm from scratch in order to cluster data.

## Working with the $k$-NN algorithm

The $k$-Nearest-Neighbor algorithm (or, shortly, $k$-NN) is a supervised algorithm to handle a classification problem with any number of labels. Its principle is particularly simple: for each new sample $\mathbf{x}$, the set $S_k(\mathbf{x})$ of its $k$-nearest neighbors among the $n$ learning samples is determined, according to a chosen distance metric (Euclidean distance is commonly used). The integer parameter $k$ must be chosen in the range $[1, n]$, of course. The class assigned to this new sample $\mathbf{x}$ is then the majority class in the set $S_k(\mathbf{x})$. Figure 1 depicts how the method works for the case of three classes.

Figure 1: How the $k$-NN works for $k = 5$ (a) and $k = 11$ (b), respectively. In this example, data belongs to three classes, having label $y = 1$, $y = 2$ or $y = 3$. The estimated class/label is denoted as $\hat{f}_k(\mathbf{x})$.



(a) $k = 5$

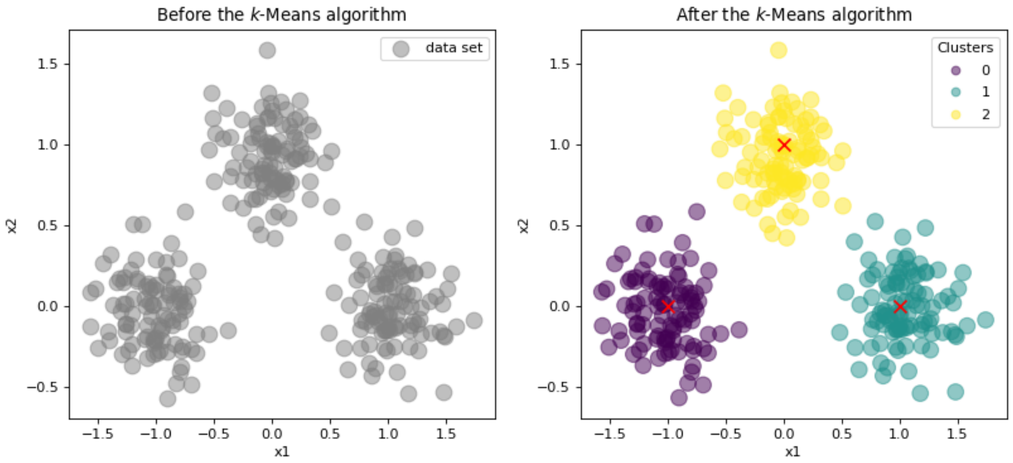
(b) $k = 11$

# Working with $k$-Means Clustering

The $k$-Means Clustering is one of the most popular machine learning algorithms for cluster analysis in data mining. The $k$-Means algorithm is an unsupervised learning algorithm, i.e. no need for training data, which aims to look at the hidden patterns between the data and group them into clusters, so part of data that has some shared properties will fall into a cluster or a similar group. Figure 2 depicts how the method works for the case of three clusters.

Given a data set of samples and a chosen distance metric, the goal is to group the samples into $k$ clusters. The algorithm starts by initializing $k$ cluster centers (or centroids), which can be randomly chosen, from some prior knowledge or by some initialization method. Compute the distance from each sample to all centroids and assign it to the nearest cluster centroid. Adjust the centroid of each cluster by taking the average of all the data points that belong to the corresponding cluster. Repeat these steps until convergence is achieved, or up to a maximum number of iterations, or when the difference between the old and the new centroids is negligible.

Figure 2: How the $k$-Means works for three clusters ($k = 3$).



# Getting started!

## 1 Toy data set

In this part we will implement the $k$-NN algorithm for classification and the $k$-Means algorithm for clustering.

Consider a data set $\{(\mathbf{x}(n), y(n))\}_{n=1}^{300}$ consisting of 300 points $\mathbf{x}(n) = (x_1(n), x_2(n)) \in \mathbb{R}^2$ and their labels $y(n) \in \{0, 1, 2\}$ generated as follows. The first 100 points have label $y(n)=0$ and are generated according to a Gaussian distribution $\mathbf{x}(n) \sim \mathcal{N}([-1, 0], \sigma^2\mathbf{I})$, the next 100 points have label $y(n)=1$ and are generated according to a Gaussian distribution $\sim \mathcal{N}([1, 0], \sigma^2\mathbf{I})$, and the remaining 100 points have label $y(n) = 2$ and are generated according to a Gaussian distribution $\sim \mathcal{N}([0, 1], \sigma^2\mathbf{I})$. Split the data set into a training and test set, containing respectively 75% and 25% of the data set.

### $k$-NN algorithm for classification

1. Generate one data set with $\sigma^2 = 0.10$. Classify the test data set using the $k$-NN algorithm with respect to the Euclidean distance for $k \in \{1, 2, 5, 10\}$. In order to visualize all data set and the corresponding label, plot all generated points in a coordinate plane and the estimated decision boundary for each $k$, similarly to Fig. 3. Comment on what happens with the decision boundary when $k$ increases.
*Hint: Use matplotlib.pyplot.countourf to plot the decision boundary.*

2. Fix the variance $\sigma^2 = 0.10$ and $k = 1$. Run the algorithm over 50 randomly generated data sets (training and test data set), compute the average error rate of the test data set and its standard deviation. Repeat it with $\sigma^2 \in \{0.15, 0.20, 0.25\}$. Plot the average error rate of the test data set *versus* the variance $\sigma^2$, use error bars to

represent the standard deviation. Comment.

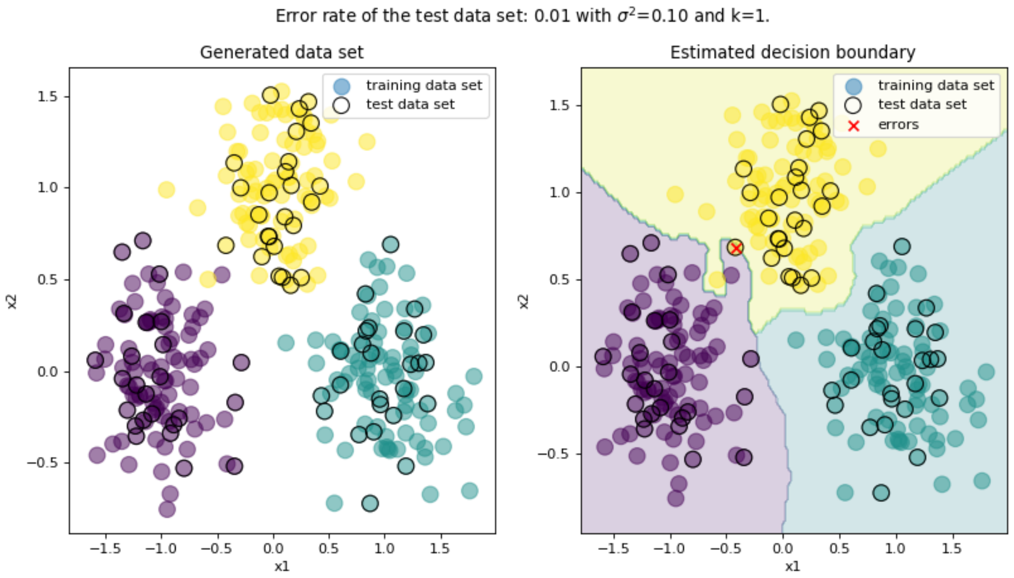


Figure 3: Estimated decision boundary.

## $k$-Means algorithm for clustering

In this part we do not need to split the data set into training and test data set and we will not use the knowledge about the input labels ($y(n)$ is no longer available).

1. Generate the data set with $\sigma^2 = 0.15$. Group the data set into three clusters ($k = 3$) using the $k$-Means algorithm with respect to the Euclidean distance. Repeat it with $k \in \{2, 4, 6\}$. Visualize all data set and the centroids obtained at the convergence for each $k$. Comment.

2. Generate the data set with $\sigma^2 \in \{0.10, 0.25, 0.50\}$. For each $\sigma^2$, group the data set into three clusters ($k = 3$) using the $k$-Means algorithm with respect to the Euclidean distance. Visualize all data set and the centroids obtained at the convergence, compare them with the centroids that generated the input data, i.e. $\{(-1, 0), (1, 0), (0, 1)\}$ for each $\sigma^2$. Comment.

## 2 *k*-NN regression: Szeged-weather data-set

In this part we will apply the k-NN algorithm for prediction.

Let's use the Szeged-weather data set that can be downloaded in `https://www.kaggle.com/budincsevity/szeged-weather/data`. We want to predict the apparent temperature given humidity and temperature. Apparent temperature is a notion of temperature that reflects human perception.

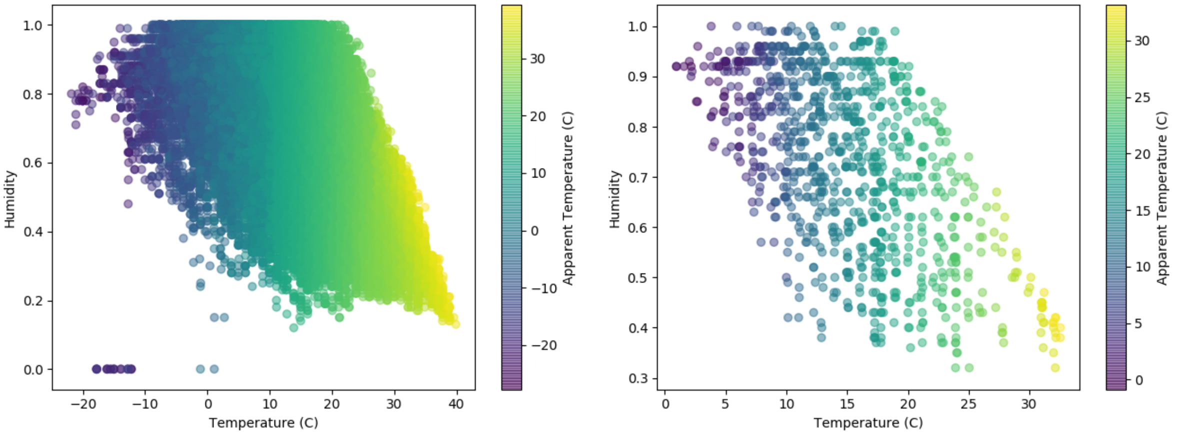
**Short description:** The Szeged-weather data-set is a daily/hourly summary for Szeged, Hungary area, between 2006 and 2016, in terms of temperature, humidity, apparent temperature, pressure, wind speed, among other measurements.

**Data Preparation:** For simplicity we will consider only three attributes: apparent temperature, humidity, and temperature, and only the first 2000 samples of the data set. Permute the order of the 2000 samples uniformly at random, and split the data set into 5 partitions of the data set (folds). Each fold is then used once as a test while the 4 remaining folds form the training set.
*Hint: Use sklearn.model_selection.KFold to split data.*

1. Visualize all data set in terms of temperature (x-axis), humidity (y-axis), and apparent temperature (color). You should obtain a figure similar to Fig. 4a.

2. Consider the first 2000 samples of the data set, you should get Fig. 4b. The prediction, given by the $k$-NN algorithm, is computed by taking into account the average of the values of $k$ nearest neighbors. Predict the apparent temperature given humidity and temperature using the $k$-NN algorithm with respect to the Euclidean distance for $k = 1$. Repeat five times, using each fold as test at a time. Compute the mean squared error of the test data set and its standard deviation.

3. Repeat the previous item for $k \in \{3, 5, 7, 10, 15\}$. Plot the mean squared error of the test data set *versus* the parameter $k$, use error bars to represent the standard deviation. Comment.



(a)

(b)

Figure 4: Visualization of three attributes of the Szeged-weather data set: (a) shows all data set and (b) shows only the first 2000 samples of the data set.