ECG Signal Classification with the k-NN Algorithm

Objective

Implement and use the k-NN algorithm for classification of various signals.

Theoretical aspects

ML decision with multiple samples in Gaussian noise

In a detection problem with Gaussian noise, we have seen in the lectures that decision with the Maximum Likelihood criterion comes down to **choosing the smallest distance**:

$$|r(t_0) - s_0(t_0)| \underset{H_0}{\overset{H_1}{\gtrless}} |r(t_0) - s_1(t_0)|$$

What happens when we have **multiple samples**?

1. We have vectors of samples: $\mathbf{r}, \mathbf{s_0}, \mathbf{s_1}$ (bold font = it is a vector)
2. The distance between any two vectors $\mathbf{a}$ and $\mathbf{b}$ is the **Euclidean distance**:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots (a_n - b_n)^2}$$

3. We use the same distance-based rule:

$$d(\mathbf{r}, \mathbf{s_0}) \underset{H_0}{\overset{H_1}{\gtrless}} d(\mathbf{r}, \mathbf{s_1})$$

   The smallest distance wins.

The k-NN algorithm

Suppose we have a set of **training signals** whose classes are known beforehand. For example:

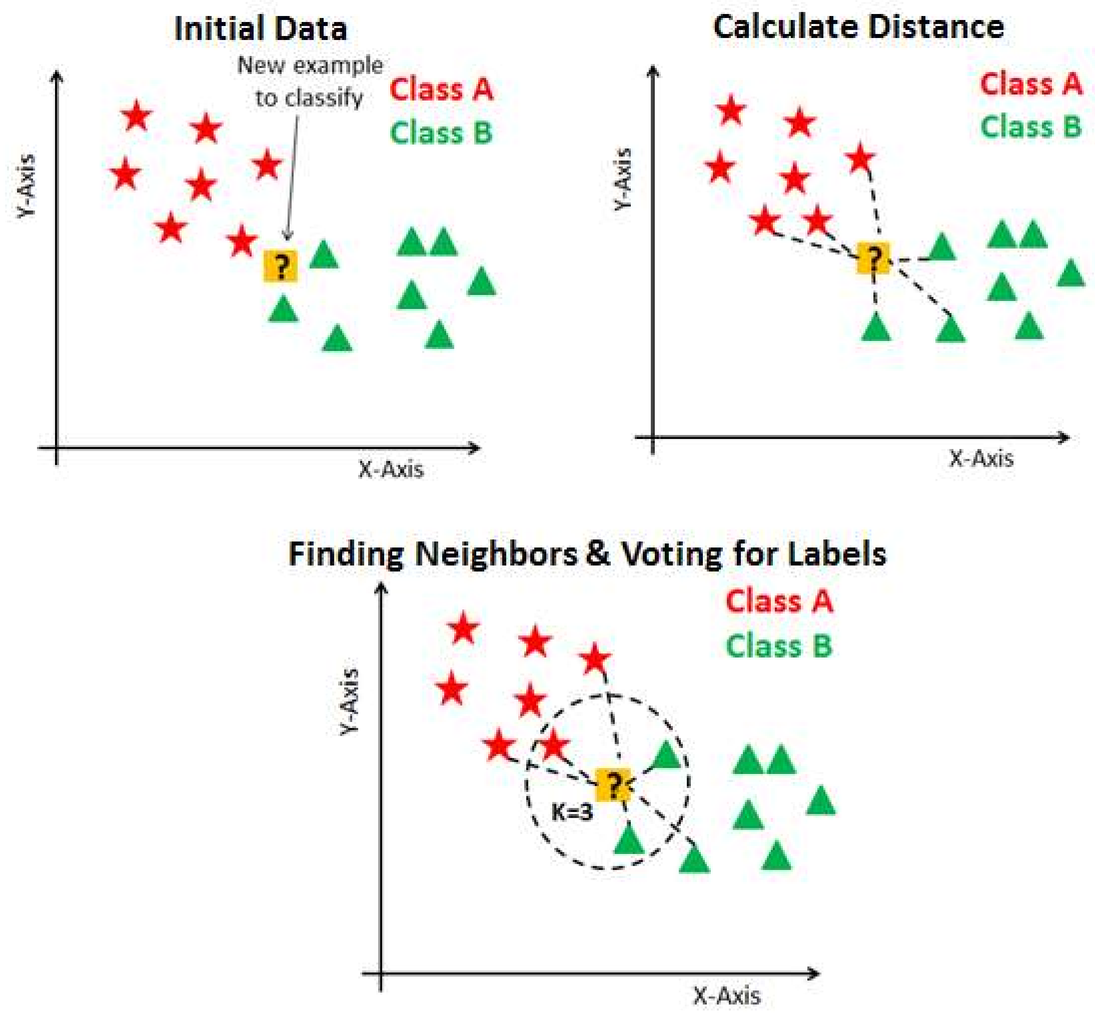
- 100 signals of class A (e.g. ECG heart signals from healthy persons)
- 100 signals of class B (e.g. ECG heart signals from ill persons)
- maybe more classes

We have a new signal X. We need to decide to which class it belongs (class A, or class B, etc).

We can use **the k-NN algorithm**:

1. Compute the distances from X to all the signals in the training set
2. Choose the **closest k neighbors**, take the class of the majority of them (e.g. majority voting). Decide that this is the class of X.

A visual illustration is below:



(image from "KNN Classification using Scikit-learn", Avinash Navlani, https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn)

Datasets organization

Usually, we have at our disposal a large class of signals whose classes are known. The data is randomly split into:

- a **training set**: this data is used for the majority voting
- a **test set**: used only for **evaluation** of the algorithm performance. This data should never be used for training (the algorithm should never have seen this data before the testing).
- (optional) a **cross-validation set**: a subset of the training set, used to determine which values of k work best

The datasets are obtained by randomly splitting all the signals available at the beginning. Common sizes of the datasets should be around 70% for the training set, 15% for the cross-validation set, 15% for the testing set.

Data for this laboratory

In this laboratory we will use ECG signal data from the MIT-BIH Arrhythmia database.

(image from https://archive.physionet.org/physiobank/database/mitdb/)

The excerpt provided for this lab contains electrocardiographic (ECG) signals from **4 classes**, with **120 signals per class**. The 4 classes are:

1. 1 class with ECG from healthy persons
2. 3 classes with 3 different types of arrhythmia (irregular/abnormal heart beat)

The ECG signals provided here are preprocessed:

- all signals are segmented in **segments** corresponding to **one heart beat**
- the signals are resized to **fixed length** 256 samples
- the signals are resized so that the peak R wave is located at the center of the signal
- the continuous component of all signals has been removed
- the signals have been normalized to norm equal to 1

The signals are randomly split into two sets:

- training set: `ECG_train.mat` , 400 signals = 4 classes $\times$ 100 signals each
- test set: `ECG_test.mat` , 80 signals = 4 classes $\times$ 20 signals each

Exercises

1. Load the data files `'ECG_train.mat and 'ECG_test.mat` . Explore the dataset:

   - display 3 signals from each class contained in the training set. Try to figure out some visual differences.
   - display the first signal from the test dataset. Try to determine visually to what class it belongs to.

1. Implement a function `[class] = myKNN(signal, k, trainset)` for performing k-NN classification of a signal:

    - the function takes as input an unclassified signal `signal`, the parameter value `k`, and the training set matrix `trainset`
    - the function computes the Euclidean distance between `signal` and each vector from the training set
    - the output `class` is defined by the majority of the k nearest neighbours of the signal

1. Call the function `myKNN` for each signal from the testing set and compare the classification results against the ground truth. Use different values for k: $k=1$, then $k=5$, then $k=15$. Compute the **confusion matrix** A, where A_{ij} = how many signals of class i are classified by our algorithm as being in class j.

1. Repeat exercise 3, this time adding a variable amount of gaussian noise to the test signals. How does the performance change?

1. Repeat exercise 4., this time adding a DC component to the test signals. How does the performance change?

Final questions

1. How does the confusion matrix look like in the ideal case? (perfect classification)

2. Is there a problem in case the classes are imbalanced? (different number of signals for the classes)