Signal Classification with the k-NN Algorithm

Laboratory 4, DEDP

Objective

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

Theoretical aspects

The k-NN algorithm

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

- 10 images of class A (e.g. images of cats)
- 10 images of class B (e.g. images of dogs)
- …

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

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).

Datasets organization

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. They sizes of the datasets should be around:

- 60% of all data for the training set
- 20% of all data for the cross-validation set
- 20% of all data for the in the testing set

Exercises

1. Load the data file 'face_dataset.mat'. Explore the dataset:
   - display 5 images from the dataset
   - print the image sizes
2. Split the dataset as follows:
   - 80% of images of each class as the `training set`
   - 20% of images of each class as the `test set`
   - save the datasets as different files `trainset.mat` and `testset.mat`
3. Implement a function `[class] = myKNN(image, k)` for performing k-NN classification of an image:
   - the function takes as input an image `image`
   - the function loads the training set from `trainset.mat`
   - the function computes the Euclidean distance between `image` and each image from the training set
   - the output `class` is defined by the majority of the k nearest neighbours of the image
4. Call the function `myKNN` for each image from the testing set and compare the classification results against the ground truth.
   Print the *confusion matrix*: A_{ij} = percentage of each images of class i which are classified by our algorithm as being in class j.

Final questions

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