Artificial Inteligence - LAB 4

ISEP – December 1, 2020

Instructions: Prepare a report including the source code and the results. Send your report by e-mail to your Instructor class.

Remark: This lab will be done using Python 3.6.

The objective of this lab is to program and test two classification algorithms, very simple but very effective: the K-Nearest Neighbor (KNN) algorithm and the Classifier Bayesian Naive (CBN). We are studying here only the simplest versions of these algorithms. For this lab we will need to import sklearn and numpy. The tests can be done on sklearn's predefined data that comes with their class labels (target), for example:

```
iris = datasets.load_iris ()
X = iris.data
Y = iris.target
```

A Introduction – Analysing Fiher's Iris with the K-Means algorithm

1. Open the file *iris.data* using the command **datasets.load** with the correct parameters.
2. The last column in your data contains the labels matching with the Iris specie to which each data belongs. Remove these labels from the main set and store them in another vector.
3. Use the command PCA command of sklearn to do a Principal Component Analysis on your data. Then use the following lines to retrieve the dataset projected on the two first principal components.

You should now have 2 different data sets, the original one stored in a first variable (e.g. "*data*"), and the same data projected on 2 components stored in the variable X.

4. Use the K-Means algorithm of sklearn on your data X to obtain a partition with 3 clusters and visualize your results.
   - Use the command **cl=kmeans(*data*,*nbclusters*)**.
   - Visualize the result using **plot($\cdots$)** with the right parameters.
   - Plot the centroids using the following command centers with the right parameters.
5. Repeat question 4) several times. What happens ? Comment.

6. Use the command **plot(⋯)** to project the labels that you stored in a separate vector in question 2). Compare these results with the partitions from your K-Means experiments. Comment.

7. Start again questions 4) to 6) using the original data ("*data*") instead of the projected ones. How different are the results ? Explain the pros and cons of using the projected data or the original ones.

B Nearest neighbor

The Nearest Neighbor algorithm is a very simple classification algorithm which is based on the following principle: the class of each test data (to be classified) must be the class of the closest (most similar) data among the training data. List of useful functions:

```
- metrics.pairwise.euclidean_distances: calculates distances between data.
- argsort: returns the indices of the ordered vector
- argmin, argmax: returns the indices of the minimum/maximum values
- neighbors.KNeighborsClassifier:  K Nearest Neighbors alg. of sklearn
```

1. Create a TNN function (X, Y) which takes X data and labels as input Y and which returns a label, for each data, predicted from the nearest neighbor of this data. Here we take each data, one by one, as data test and we consider all others as learning data. That we allows to test the power of our algorithm according to a validation method by cross validation of leave one out type.

2. The TNN function calculates a predicted label for each data. Change the function to calculate and return the prediction error: i.e. the percentage badly predicted labels

3. Test on Iris data.

4. Test the function of the K Closest Neighbors of sklearn (with here K = 1). Are the results different? Test with other values of K.

5. BONUS: Modify the TNN function so that it takes as input a number K of neighbors (instead of 1). The predicted class will then be the majority class among the K neighbors.

C Naive Bayesian classifier

The Naive Bayesian Classifier algorithm is a classification algorithm based on calculating the probability of belonging to each class. That is to say that the test data (to be classified) will be assigned to the most likely class. The probability of belonging to each class is calculated from the learning data as follows:

$$class(x) = \arg\max_{\omega_k} \{\Pi_i P(w_i/\omega_k) P(\omega_k)\} \quad (1)$$

Here $P(\omega_k)$ is the a priori probability of belonging to the class k. in other words its the probability of obtaining a data of class k if we draw a data at random. $P(x_i/\omega_k)$ is the probability that a data x has the value x_i for the variable i, if we know its class ω_k. Here we will calculate this probability by calculating the distance between the data x_i and each barycenter of the classes (i.e. the class average), divided by the sum distances between this data and each barycenter.

List of useful functions:

- `mean, sum: calculate the mean and the sum of a list of values.`
- `unique: returns the list of list values, without repeating values.`
- `asarray: transforms a list into a vector.`
- `vector.prod: makes the product of the values of a vector.`
- `naive_bayes.GaussianNB: the naive Bayesian Classifier from sklearn.`

1. Create a CBN (X, Y) function that takes X data and labels as input Y and which returns a label, for each data, predicted from class la more likely according to equation (1). Here again, we take each data, one by one, as test data and we consider all data as training data. It is advisable to first calculate the barycentres and the a priori probabilities $P(\omega_k)$ for each class, then calculate the conditional probabilities $P(x_i/\omega_k)$ for each class and each variable.

2. The CBN function calculates a predicted label for each data. Change the function to calculate and return the prediction error: i.e. the percentage of badly predicted labels. Test on Iris data.

3. Test the function of the Naive Bayesian Classifier included in sklearn. This function uses a Gaussian distribution instead of the barycenter distances. The results are they different?

4. BONUS: Modify the CBN function so that it uses a Gaussian distribution for probability laws instead of simple distance to the barycenter.