ROB311 - TD1 - Apprentissage pour la Robotique

Araujo Belén, Victor
De Carvalho Ferreira Sula, Julia

September 2019

1 KNN-Algorithm

Write a python program to implement K-NN to work with several datasets. Please verify with the 2 datasets below. Calculate also the confusion matrix, the accuracy, and plot the data.

The K-NN algorithm stands for the k-nearest neighbor. This machine learning method aims to classify, for example, x_{test} assigning it the most frequent label among its k closest neighboring samples x_{train}. To determine the closest neighbors a distance is calculated between the sample to be classified and the ones the algorithm has access to.

1.1 Functions TD1.py

Used libraries: numpy, random, matplotlib.pyplot, statistics, math, and mplot3d.

1.1.1 loadfile(filename, class_index)

This function is responsible for loading the data files and dividing them into features and labels.

Opening the file was achieved by using the open() function and setting its parameter to *"r"* to open it in reading mode.

While analyzing the file *"breast-cancer-wisconsin.data"*, line breaks "*n*" were noticed, therefore during the formatting of the file this had to be taking into consideration.

The main steps for dividing the files into X and Y values (data information and label information, respectively), were :

- Dividing the files into lines
- Splitting the lines (", " was the separator)
- Appending in Y the label files erasing "*n*".The *"class_index"* variable allows indicating in which position of the line the label is.
- Appending in X all features

In the case of the *"breast-cancer-wisconsin.data"*, the ID number is not interesting for the K-NN calculation process, as they are chosen by the Doctor or the Laboratory and have no relation with the tumors. Consequently, in this case, the ID information was removed by the *"pop()"* function.

1.1.2 split_data(x,y,train_fraction)

To apply a machine learning method, two data sets are needed, one will be used in the training to refine the classification parameters and one will be used in the testing to verify the accuracy of the method. This function receives the features(x) and the labels(y) and divides it in the subsets train and test. The parameter train_fraction information represents the percentage of data dedicated to the training process. To fairly chose the data division the *"randint"* function is used to select a random index and, hence, a random line to be part of one of the subsets. The method *pop()* was used to make sure that the two subsets were disjoint.

1.1.3 Distance Calculation

There are different distances definitions. In this code, four types of distance's calculation methods were implemented as shown in the equation bellows.

Comparing their results, it was shown that distance definition calculated may strongly affect the prediction's accuracy.

Euclidian distance:

$$d_{q,p} = \sqrt{(\sum_{i=1}^{n})(q_1 - p_1)^2 + (q_2 - p_2)^2 \dots (q_n - p_n)^2} \tag{1}$$

Manhattan distance:

$$d_{q,p} = |q_1 - p_1| + |q_2 - p_2| + \cdots + |q_n - p_n| \tag{2}$$

Chebyshev distance:

$$D_{Chebyshev} = max_i(|x_i - y_i|) \tag{3}$$

Hamming distance:

$$\begin{array}{c} D_H = \sum_{i=1}^{n} |x_i - y_i| \\ x = y \longrightarrow D = 0 \\ x \neq y \longrightarrow D = 1 \end{array} \tag{4}$$

During the distance calculation, it was noticed that the *"breast-cancer-wisconsin.data"* had certain fields which were not filled (they were set to *"?"*). To not disregard the whole line, it was chosen to consider that *"?"* could be any value, consequently, its distance to any value would be zero.

1.1.4 kNN(x_train, y_train, x_test, k, distance)

The function kNN implements the algorithm. It compares the distances between the features of the train data set with the test data set and then assigns the label of the most similar train entry to the test entry. This label is the classification prediction saved as the *"y_pred"* vector.

This function is decomposed in:

- Distance data calculation: the distance is calculated and saved in a vector distances.
- Sorting distance: *"argsort"* function is used to get the *"k"* shorter distances index between the train and the test data
- Voting: *"mode()"* function allows to determinate the most probable label.
- Saving: predicted labels are saved

1.1.5 confusion_calcul(y_test, y_pred, class1, class2)

As its name indicates, this function calculates the confusion matrix of the prediction. The *”class1 and class2”* variables define the classes that will be considered and the predicted value of *y_pred* is compared with the real labels, then the matrix values are affected according to the next example:

	A_{pred}	B_{pred}
A_{test}	A_{true}	A_{false}
B_{test}	B_{false}	B_{true}

1.1.6 accuracy_calcul(y_test, y_pred)

In order to determinate the accuracy of the prediction, we compare the *y_test y_pred* data, if the value is the same, the local counter *accuracy* is augmented. In the end, the total is divided by the data size to obtain the final accuracy fraction.

1.1.7 choose_property(property1)

This is a function that has been created in order to improve the way to show results. It allows to put the right axes labels automatically, and makes the code easier for the user, as we only have to know the features name, not its position.

1.1.8 plot_data(x, y, property1, property2)

In order to plot the data for the *”breast-cancer-wisconsin.data”* , it was chosen to plot features versus features, since it was not possible to plot the 9 features together. To choose the property to be plotted it suffice to write in natural language the wanted features.

1.1.9 plot_data3d

In order to plot the data for the *”haberman.data”* , as there only were 3 features a 3d graph was chosen.

1.1.10 distance_comparison

The different distance definition, as stated previously, influence the algorithm prediction, therefore, as to find out the best distance choice for each data set this function was established.

The KNN algorithm is run for each distance and the one with better accuracy output the predicted labels.

1.1.11 plot_accuracy

To easily compare the distance calculation, this function was created to output a graph bar with the accuracy of each prediction according to the distance used.

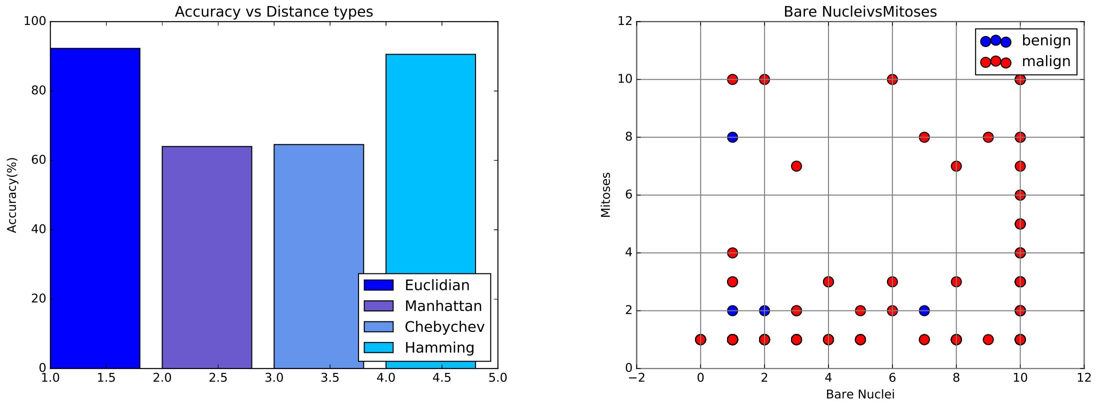
1.2 Results

The algorithms in the two data set were tested using 50% of the data to the training stage and 50% to the testing and considering k=5 neighbors.

1.2.1 breast-cancer-wisconsin.data

After applying the KNN algorithm to the breast-cancer-wisconsin it was clear that the Euclidian Distance provided better results, as the average accuracy achieved was of 95%. The accuracy according to each distance used in the KNN algorithm can be seen in figure 1(a)

Furthermore, plotting the graph of different features, such as in figure 1(b), offers a little intuition in how the prediction was made. For example, in figure 1(b) high Bare Nuclei (higher than 9) normally indicates that the tumor will be malign, the same could be said for high Mitoses (higher than 9).



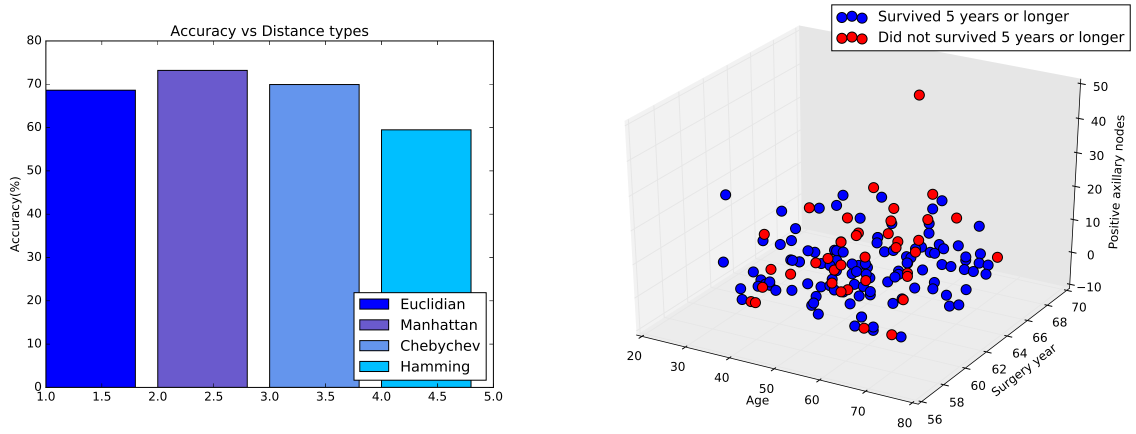
(a) Accuracy Comparison

(b) Comparison of features Bare Nuclei vs Mitoses

1.2.2 haberman.data

In the Haberman database, the accuracy scores were lower and the distance definition providing the best accuracy was the Manhattan (around 72-74% of accuracy) as shown in figure 1(c).

This behavior could have two explanation, the first one is we have access to fewer features to predict who survived or not. Secondly, as shown in figure 1(d), it seems as the cases of survival or not can have similar features (considering only the available features). Therefore, the KNN algorithm is rendered less effective as features distances do not entail in class change.



(c) Accuracy Comparison

(d) Features Comparison

Conclusion

KNN's algorithm has shown acceptable results in the class prediction performed. One of KNN's perks is that the data division can take any form, that is to say, that the division boundary could have any format as it does not make any assumption on data distribution. However, as seen in the Haberman's data set case KNN's depends on the local structure of the date. Its assumption is same class sample will be similar or close, but that's will greatly depend on the features chosen to represent the data.

Furthermore, KNN does not offer a way to handle missing data, for example, in the case where instead of value there was a "?" the value had to be somewhat disregarded in the distance calculation. Whereas other machine learning methods offer better solutions.