Applied Inductive Learning
Project 1 - Classification algorithms

October 2015

In this first project, we ask you to write several Python scripts to answer the different questions below. One separate script is required for each of the three questions. Make sure that your experiments are reproducible (e.g., by fixing manually random seeds). Add a brief report (pdf format, 5 pages max.) giving your observations and conclusions.

Each project must be done by group of 2 students. Send a zip archive with the pdf report and python scripts before October 25, 23:59 GMT+2 to a.joly@ulg.ac.be with subject “[ELEN0062][project-1]last-name1-last-name2” where last-name1 and last-name2 are the names of the members of the group in alphabetical order.

1 Decision tree (dt.py)

With the provided function make_data, generate a sample set (X, y) of 2000 samples. The first 150 samples will be used as training set and the remaining ones will be used as testing set (X_test, y_test).

1. Build a decision tree model (sklearn.tree.DecisionTreeClassifier) on X_train, y_train. Compare visually\(^1\) the prediction of the model on X_test to the ground truth y_test. Comment your observations.

2. Observe visually the effect of the max_depth parameter on the decision frontier.

3. Evaluate the accuracy on the learning and testing sets for various values of the max_depth parameter and plot the corresponding error curve.

4. Use a ten-fold cross validation strategy and optimize the value of the max_depth parameter. What is the best score attained and best value of this parameter? Justify your answer.

2 K-nearest neighbors (knn.py)

We are going now to study a distance based method the k-nearest neighbors algorithm. Given a set of \(n\) training samples \((x^i, y^i) \in (X \times Y))_{i=1}^{n}\) and a distance measure \(d\), an unseen sample with value in the feature space \(x'\) is assigned a prediction through the following procedure:

1. First, compute the distance \(d(x^i, x')\) between the training samples and \(x'\).

2. Secondly, search for the \(k\) samples in the training set which have the smallest distance to the sample \(x'\) in the feature space.

3. Thirdly, compute the proportion of samples of each class proportion of each class among the \(k\)-nearest neighbors.

\(^1\)i.e., represent the decision frontier obtained by the estimator in the input space. To do that, you can use the function plot_boundary from plot.py that allows you to plot the decision surface of an estimator and a scatter plot of a set of points.
4. Lastly, compute the final prediction by predicting the class with the highest probability. This is equivalent to a majority vote of the \( k \) nearest neighbors.

For the following questions, re-use the same training and testing sets as in question 1.

1. Implement your own k-nearest neighbors estimator according to the above description and following the scikit-learn convention (http://scikit-learn.org/dev/developers/). The parameters of the algorithm should be \( n\_neighbors \), the number of considered neighbors. The distance measure to implement is the euclidean distance. *Suggestion: Fill in the class whose template is given in knn.py.*

2. Build a one-nearest neighbor model (sklearn.neighbors.KNeighborsClassifier) on \( X\_train \), \( y\_train \). Compare visually the prediction of the model on \( X\_test \) to the ground truth \( y\_test \). Comment your observations.

3. Observe visually the effect of the \( n\_neighbors \) parameter on the decision frontier.

4. Evaluate the accuracy on the learning and testing sets for various values of the \( n\_neighbors \) parameter and plot the corresponding error curve.

5. Use the same cross-validation strategy that you use previously and optimize the value of the \( n\_neighbors \) parameter. What is the best score attained and best value of this parameter? Justify your answer.

### 3 Linear model classifier (linear_model.py)

One way (among many) to train a linear (binary) classifier is to train a linear regression model to a new output \( y \) set to 1 for the examples of one class and to -1 for the examples of the other class. A prediction is then obtained for a new example by comparing its predicted numerical output \( y \) with 0. This method is implemented in the scikit-learn RidgeClassifier function which fits an ordinary least square model.

Re-use the same training and testing sets as in question 1 to answer the following questions:

1. Build an ordinary least square model (sklearn.linear_model.RidgeClassifier with \( alpha=0 \)) on \( X\_train \), \( y\_train \). Compare visually the prediction of the model on \( X\_test \) to the ground truth \( y\_test \). Comment your observations.

2. Imagine and implement a way to modify the input data in order to improve the performance of the linear methods for this specific problem. Describe precisely the transformation that you have implemented. Compare visually the predictions of the model on \( X\_test \) to the ground truth \( y\_test \). Comment your observations.

3. Evaluate the accuracy of the model with and without the proposed transformation.