ELEN0062 - Introduction to machine learning
Project 1 - Classification algorithms

September 2019

The goal of this first assignment is to get accustomed to the basics of machine learning and concepts such as under- and over-fitting. We ask you to write several Python 3 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.

The assignment must be carried out by group of *two students* and submitted as a `tar.gz` file on Montefiore's submission plateform (`http://submit.montefiore.ulg.ac.be`) before *October 20, 23:59 GMT+2.*

Note that attention will be paid to how you present your results. Careful thoughts in particular – but not limited to – should be given when it comes to plots.

Files

You are given several files, among which are `data.py` and `plot.py`. The first one generates binary classification datasets with two real input variables. In the following, you will work on two datasets generated by `make_data1` and `make_data2`, respectively. You can generate datasets of 2000 samples. The first 150 will be used as training set and the remaining ones as testing set.

The second file contains a function which depicts the dataset together with the decision boundary of a trained classifier.

The other files must be completed and archived together with the report.

1 Decision tree (`dt.py`)

In this section, we will studying decision tree models (see the `DecisionTreeClassifier` class from `sklearn.tree`). More specifically, we will observe how model complexity impacts the classification boundary. To do so, we will build several decision tree models with `max_depth` values of $1, 2, 4, 8$ and `None` (which corresponds to an unconstrained depth). Answer the following questions in your report.

1. Observe how the decision boundary is affected by tree depth:
   (a) illustrate and explain the decision boundary for each depth;
   (b) discuss when the model is clearly underfitting/overfitting and detail your evidence for each claim;
   (c) explain why the model seems more confident when the depth is unconstrained.
2. Report the average test set accuracies (over five generations of the dataset) along with the standard deviations for each depth. Briefly comment on them.

2 K-nearest neighbors (`knn.py`)

In this section, we will be studying nearest neighbors models (see the `KNeighborsClassifier` class from `sklearn.neighbors`). More specifically, we will observe how model complexity impacts the classification boundary. To do so, we will build several nearest neighbor models with `n_neighbors` values of $1, 5, 10, 100, 150$ and 1200. Answer the following questions in your report.

1. Observe how the decision boundary is affected by the number of neighbors:
   (a) illustrate the decision boundary for each value of `n_neighbors`.
   (b) comment on the evolution of the decision boundary with respect to the number of neighbors.
2. Use a ten-fold cross validation strategy to optimize the value of the `n_neighbors` parameter and obtain an unbiased estimate of the test:
   (a) explain your methodology;
   (b) report the score you obtain and the optimal value of `n_neighbors`. Do they corroborate your decision boundary-based intuition? Justify your answer.

3 Naive Bayes classifier (`naive_bayes.py`)

In this section, we will be studying the Naive Bayes (NB) classifier model. NB is a probability-based method which computes its predictions according to the posterior probability:

$$\hat{f}(\mathbf{x}) = \text{argmax}_y \, Pr(y|x_1, \ldots, x_p), \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_p)$ denotes the p input variables. To compute this posterior, the NB classifier postulates that the input variables are conditionally independent given the output. More formally,

$$P(\mathcal{X}_i|\mathcal{Y}, \mathcal{Z}) = P(\mathcal{X}_i|\mathcal{Y}) \quad \forall \mathcal{Z} \in \mathcal{P}(\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots X_p\}) \tag{2}$$

where $\mathcal{P}(S)$ designate the power set of S. Given this hypothesis, the predictions of the NB classifier can be obtained as

$$\hat{f}(\mathbf{x}) = \text{argmax}_y \, Pr(y) \prod_{i=1}^{p} Pr(x_i|y) \tag{3}$$

To compute (3), the prior (*i.e.* $Pr(y)$) and the likelihood (*i.e.* $Pr(x_i|y)$) probabilities must be estimated. For discrete variables – which is the case of $\mathcal{Y}$, one chooses the corresponding ratio over the learning set. For continuous variables, an additional assumption must be made with respect to the underlying distribution. One common choice is to presume Gaussianity:

$$Pr(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}\right) \tag{4}$$

where the population's mean μ_i and variance σ_i^2 are estimated by the sample mean and variance.

Despite its simplicity, this learning algorithm has proven itself quite useful in several areas and in text-based problems in particular.

1. Show that (3) is equivalent to (1) under the NB independence assumption.
2. Implement your own NB estimator according to the above description and following the scikit-learn convention (`http://scikit-learn.org/dev/developers/`). *Suggestion: Fill in the class whose template is given in `naive_bayes.py`.*
3. Compute the testing set accuracy on both datasets and interpret the results.