ELEN0062 - Introduction to machine learning
Project 1 - Classification algorithms

September 30th, 2022

In this first project, you will get accustomed with some classical machine learning algorithms and concepts, such as under- and over-fitting. For each algorithm, we ask you to deliver a separate Python script. Make sure that your experiments are reproducible (e.g., by manually fixing random seeds). Add a *brief* report (pdf format, roughly 4 pages without the figures) giving your observations and conclusions. Each project must be carried out by groups of *two* to *three students* and submitted as a `tar.gz` file on Montefiore's submission platform (`http://submit.montefiore.ulg.ac.be`) before *October 23, 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. More precisely, the examples are sampled from two Gaussian distribution.

In the following, you will mainly work with `make_dataset2` (see Figure 1), where one gaussian is circular and the other is elliptic. `make_dataset1`, a dataset where two identical Gaussians are shifted, will only be required for question 3.

You can generate datasets of 1 500 samples. The first 1 200 will be used as training set and the remaining ones as testing set.

The second file contains a function which depicts the decision boundary of a trained classifier. Note that you should use a dataset independent of the training set to visualize the boundary.

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

1 Decision tree (`dt.py`)

In this section, we will study 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`. For this question, you should work on datasets generated by `make_dataset2`. Answer the following questions in your report.

1. Observe how the decision boundary is affected by tree complexity:
   (a) illustrate and explain the decision boundary for each hyperparameter value;

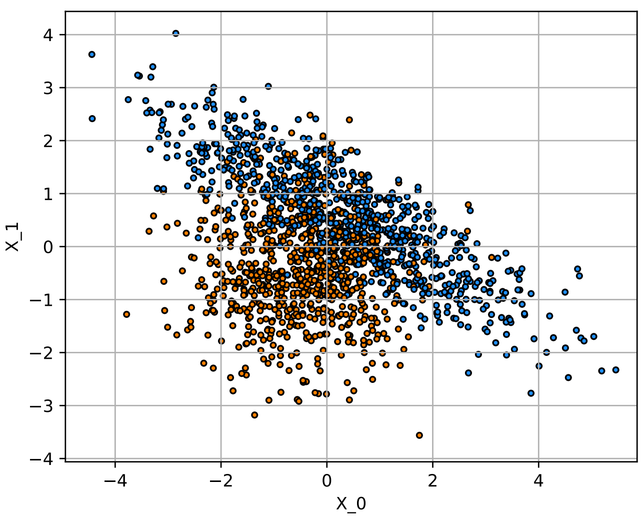

Figure 1: Second dataset.

(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 `max_depth` is the largest.

2. Report the average test set accuracies over five generations of the dataset, along with the standard deviations, for each value of `max_depth`. Briefly comment on them.

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

In this section, we will study 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 $1, 5, 25, 125, 625$ and 1200. For this question, you should work on datasets generated by `make_dataset2`. 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. Report the average test set accuracies over five generations of the dataset, along with the standard deviations, for each value of `n_neighbors`. Briefly comment on them.

3 Quadratic/Linear discriminant analysis (`qda.py`)

Quadratic and linear discriminant analyses are two classification methods (not covered in the theoretical lectures) that are based on the assumption that input vectors from each class are distributed according to a multivariate Gaussian distribution. Class conditional density functions $f_k(x)$ (with $k = 1, \dots, K$) are then written as follows:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \tag{1}$$

where $x \in \mathbb{R}^p$ is the feature vector and μ_k and Σ_k are respectively the mean and covariance matrix corresponding to class k. Applying Bayes' theorem yields that the probability of an example x belonging to class k is given by:

$$P(y = k|x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \tag{2}$$

where $\pi_k = P(y = k)$ $(k = 1, \dots, K)$ is the prior probability of class k. The final predicted class, that minimizes the error rate, is then the most probable class at x given by $\text{argmax}_k\, P(y = k|x)$.

Linear discriminant analysis (LDA) assumes that the covariance matrices are equal for each class, i.e., $\Sigma_k = \Sigma \quad \forall k$. This property is called homoscedasticity. On the other hand, quadratic discriminant analysis (QDA) makes no such assumption.

Learning a QDA/LDA model requires to estimate the prior probabilities π_k $(k = 1, \dots, K)$, the means of each class μ_k $(k = 1, \dots, K)$ and the different class conditional covariance matrices Σ_k for QDA and the common covariance matrix Σ for LDA. These can be inferred from the data by the usual maximum likelihood estimators. Contrary to k-nearest neighbors and decision trees, QDA/LDA does not have any hyper-parameters to tune (but relies on the Gaussian and, for LDA, homoscedasticity assumptions).

We ask you to implement your own QDA/LDA estimator according to the above description and following the scikit-learn convention [1] by filling in the `qda.py` file. Both algorithms will be implemented in the same functions, with the choice between QDA and LDA determined by a single binary hyper-parameter, denoted `lda`, that can be either `True` or `False`. The implementation can be restricted to the binary classification case.

[1] `http://scikit-learn.org/dev/developers/`

Answer the following questions in your report:

1. In the two classes case, show mathematically that the decision boundary of QDA is quadratic and that the decision boundary of LDA is linear (hence the names of the two methods).

2. Illustrate the decision boundary of both methods on the second dataset. Briefly comment on the results.

3. Report the average accuracy over five generations of the dataset along with the standard deviation for *both* methods and *both* datasets. Comment on these results by comparing the two methods.

4 Method comparison

We would like now to compare all four methods on the two datasets. Since decision trees and k-nearest neighbors depend on a hyper-parameter, we need to find a way to tune it. You can re-use the hyper-parameter values from the previous questions, paying attention to the maximum number of neighbors that can be used by your protocol.

Answer the following questions in your report:

1. Explain a way to tune the value of `max_depth` and `n_neighbors` on the learning set.

2. Implement this method and use it to compute the average test set accuracies over five generations of the dataset, along with the standard deviations, for the tuned decision tree and K-nearest neighbors methods on both datasets.

3. Compare these results with those obtained at question 3.3. Discuss the ranking of the methods on the two datasets.