ELEN0062 - Introduction to machine learning
Project 1 - Classification algorithms

October 4th, 2023

In this first project, you will get accustomed with some classical machine learning algorithms and concepts, such as under and overfitting. 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 6 pages without the figures) giving your observations and conclusions. Each project must be carried out by groups of *two* to *three students* and submitted to Gradescope[1] before *October 25, 23:59 GMT+2*. There will be two projects to submit to: one for your Python scripts and one for your report. 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. We provide a LaTeX template on the projects web page and recommend using it, although it is not compulsory.

Files

You are given several files, among which are `data.py` and `plot.py`. The first one generates a binary classification dataset with two real input variables. More precisely, you will work with (a modified version of) the `make_moons` dataset from scikit-learn (see Figure 1).

You can generate datasets of 1 500 samples. The first 1 200 will be used as the learning set and the remaining ones as the test 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 learning set to visualize the boundary.

Only modify the other files, which you must archive before submitting.

1 Decision tree (`dt.py`)

In this section, we will study decision tree models (see `DecisionTreeClassifier` from `sklearn.tree`). More specifically, we will observe how the model's complexity affects the classification boundary. To do so, we will build several decision tree models with `max_depth` values of $1, 2, 4, 8$, and `None`. 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 hyper-parameter value.
   (b) Discuss when the model is clearly underfitting/overfitting and detail your evidence for each claim.

[1] `https://www.gradescope.com`. The course entry code is JK485B. Please register with your official ULiège email address.

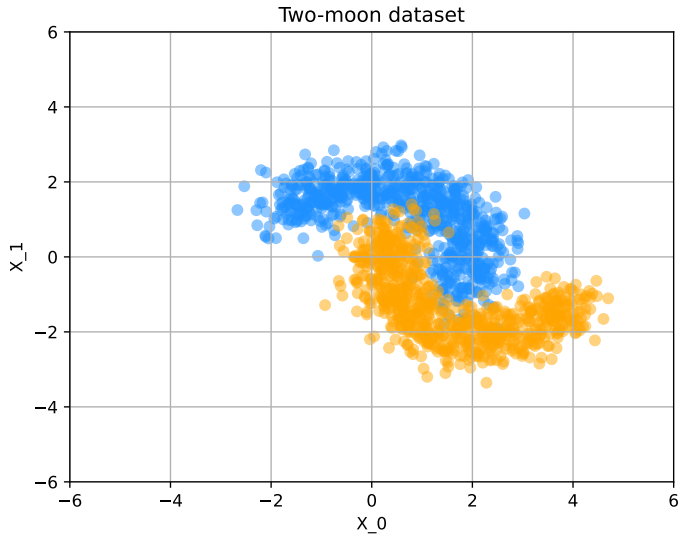

Figure 1: Example two-moon dataset (`make_moons` in scikit-learn).

(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. 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 Naive Bayes classifier (`nb.py`)

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

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

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, we assume that:

$$p\left(X_i|Y, S\right) = p(X_i|Y) \quad \forall S \subseteq \{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots X_p\}. \tag{2}$$

Given this hypothesis, the predictions of the NB classifier can be obtained as

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

To compute (3), the prior class probabilities $P(y)$ and the likelihood densities $p(x_i|y)$ must be estimated. $P(y)$ can be estimated from class frequencies in the learning set. Several methods exist to estimate the densities $p(x_i|y)$. In this project, we propose to make the assumption that these densities are Gaussian:

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_{i,y}^2}} \exp\left(-\frac{(x_i - \mu_{i,y})^2}{2\sigma_{i,y}^2}\right) \tag{4}$$

where the class-dependent mean $\mu_{i,y}$ and variance $\sigma_{i,y}^2$ can be estimated, separately for each input variable, by the sample mean and variance, conditioned on the class.

Despite its simplicity, this learning algorithm has proven itself quite useful in several areas, such as text classification for example.

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* ***`nb.py`***.

3. Using the same sizes for the learning and test sets (i.e., respectively 1200 and 300), show the decision boundary of the Naive Bayes classifier. Comment on the shape of this boundary. Try to explain the observed shape mathematically.

4. Report the average test set accuracy over five generations of the dataset, along with the standard deviation. Interpret the result.

4 Method comparison (`mc.py`)

We would like now to compare the three methods more systematically. 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 ranges of hyper-parameters values explored in 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` using only 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.

3. Compare these results with those obtained at question 3.4. Discuss the ranking of the methods on the dataset.