ELEN0062 - Introduction to machine learning
Project 1 - Classification algorithms

October 11th, 2017

In this first project, you will glean a first experience with some machine learning algorithms. More specifically, you will experiment with the decision tree and nearest neighbor algorithms as well as implement a simple linear classification algorithm. In addition, the project will discuss the problem of unbalanced datasets.

For each algorithm, we ask you to deliver a separate Python script. Make sure that your experiments are reproducible (e.g., by fixing manually random seeds). Add a *brief* report (pdf format, 3 pages maximum not counting the figures) giving your observations and conclusions.

Each project must be done by group of *two students* and submitted as a `tar.gz` file on Montefiore's submission platform (`http://submit.montefiore.ulg.ac.be`) before *October 31, 23:59 GMT+2.* Concatenate your sXXXXXX ids as group name.

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 circular gaussian distributions with the same covariance matrices, centered at $(+1.5, +1.5)$ for the negative class and $(-1.5, -1.5)$ for the positive class. In the following, you will work with `make_unbalanced_dataset` (see Figure 1), where the negative class is three times more present. You can generate datasets of 3000 samples. The first 1000 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.

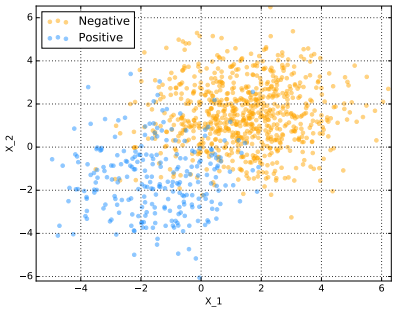

Figure 1: Unbalanced dataset.

1 Decision tree (`dt.py`)

In this section, we will study decision tree models (the `DecisionTreeClassifier` class from `sklearn.tree`). More specifically, we will observe how the model complexity impacts the classification task. To do so, we will build several decision tree models with `max_depth` values of $1, 2, 4, 6, 8$ 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 study nearest neighbors models (the `KNeighborsClassifier` class from `sklearn.neighbors`). More specifically, we will observe how the model complexity impacts the classification task. To do so, we will build several nearest neighbor models with `n_neighbors` values of $1, 5, 50, 100$ and 500. 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. Give an explanation for what you observe.
2. Use a ten-fold cross validation strategy to optimize the value of the `n_neighbors` parameter:
   (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.

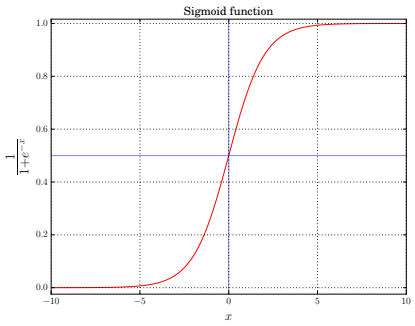

Figure 2: Sigmoid function

3 Logistic regression (`logistic_regression.py`)

The misleadingly named logistic regression is actually a classification technique, and, more precisely, a linear method. That is, the prediction of the model follows the form:

$$\hat{y} = w_0 + \mathbf{w}^T \mathbf{x} \tag{1}$$

In order to derive probabilities from this model, the output is run through a sigmoid function (Figure 2), which squishes its input to the unit range. The final form of the model is thus:

$$P(Y = 1|\mathbf{x_i}, w_0\mathbf{w}) = \frac{1}{1 + \exp(-w_0 - \mathbf{w}^T \mathbf{x}_i)} \tag{2}$$

Note that we assume that the positive class is labeled 1 and the negative class is labeled 0.

Consequently, learning a logistic regression model means learning the vector $\theta = (w_0, \mathbf{w})^T$ of parameters. To do so, one usually finds θ^* that minimizes the negative log-likelood loss function over the training set defined as:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log P(Y = y_i|\mathbf{x}_i, \theta) \tag{3}$$

In this project, we will learn the θ parameter in the simplest possible way: with a batch-mode vanilla gradient descent. The idea is to iteratively update the parameter θ according to the following rule:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\theta) \tag{4}$$

where η is an hyper-parameter called the learning rate, which is introduced to prevent overfitting.

In the present case, the gradient of the loss function is given by

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[P(Y = 1|\mathbf{x}_i, \theta) - y_i\right] \mathbf{x}'_i, \tag{5}$$

where $\mathbf{x}'_i = (1, x_i^{(1)}, x_i^{(2)})^T$ is the ith feature vector to which a constant 1 has been introduced as first component.

Implement your own logistic regression estimator according to the above description and following the scikit-learn convention (`http://scikit-learn.org/dev/developers/`). The estimator should have two hyper-parameters: the learning rate η and the number of gradient descent iterations to perform.
Suggestion: Fill in the template given in `logistic_regression.py`.

Answer the following question in your report.

1. Show that Equation 5 is indeed the gradient of the negative log-likelihood (Equation 3).

2. How do you propose to set the initial value of θ?

3. Illustrate the decision boundary on the dataset and briefly comment on the results. Specify the values of the hyper-parameters.

4. Report the average accuracy (over five generations of the dataset) along with the standard deviation. Specify the hyper-parameters you used.

5. Study the effect of the number of iterations (for a fixed but relevant learning rate) on both the decision boundary and the error.

bonus In real world applications, the data used to learn a model do not always follow the exact same distribution as the one used subsequently (*i.e.* when the model is in production). Let us consider the following scenario. You have been given a logistic regression model trained on an unbalanced dataset but you discover that the unbalancedness is due to a bias during the collection of the data. In fact, the phenomenon you are modeling is balanced class-wise. How can you adapt your model without retraining it so that it generalizes better in the balanced setting in terms of log-likelihood?