SFML Project

Prof. dr. Pieter Libin and Santiago Amaya

March 31, 2023

1 Introduction

For the course *Statistical Foundations of Machine Learning* students will be marked on a project submitted no later than Thursday, 1 June, 2023.

What is expected is an interactive research report in the form of a *Python Jupyter Notebook* addressing the assignment. It should be a **research** report, so you should state **three research questions** that you answer in the paper. You should clearly describe the research questions, and motivate why they are important, and how you answer these questions. It should be an **interactive** report, so that the reader can easily reproduce your experiments. Python Jupyter notebooks provide a way to combine text, figures and Python code to run and report the experiments. We expect that you perform at least some experiments on 2- or 3-dimensional data points such that you can visualize the dataset and the decision boundary or regression function in your report. For more information on this, the blog "Easily visualize Scikit-learn models decision boundaries" shows how such visualisations can be done.

The notebook has to be self-contained[1], and the **reader should be able to understand the report without consulting other sources**, and to **run the notebook as it is**. This means that the datasets have to be included or are imported directly from the website where the dataset originates from.

2 Submission Modalities

You submit your project on Canvas. The **strict deadline** is Thursday, 1 June, 2023 by midnight. Make sure to add the names of your group member names and the master program in which they are enrolled, to your Python notebook.

You will work together in groups of three individuals, **please report your group composition at the start of the WPO** on Tuesday, 18 April, 2023. During this WPO, we will also answer any questions you might have on the project. If you can't find a teamate, contact Prof. Pieter Libin via e-mail to find a match (pieter.libin@vub.be).

[1]The blog Learning Seattle's Work Habits from Bicycle Counts gives an example of a self-contained notebook.

3 Algorithms and datasets

You will submit **one Jupyter notebook** addressing three research questions. Make sure to mention the title of the project, the group's names and the master program(s) in which the group members are enrolled. To address your research questions, use at least two datasets, a synthetic and a real world one, to be used for training, validation and testing purposes. You will consider both a classifcation and regression problem, for each problem, you will explore two different machine learning algorithms.

Algorithms

You can use any ML-algorithm. It is not necessary that the algorithm was discussed during the classes or covered in the course material. Also you do not have to implement the algorithms yourself. You can use a library like *scikit-learn*. It is important that you give a **brief explanation of the ML-algorithm** and the most important tunable parameters. In Appendix, an example explanation of the perceptron learning algorithm is given, to provide you with an example.

Datasets

1. *synthetic* dataset: You can use the python module *scipy.stats* for that purpose. You need to ensure that you have control about the data generation process, in order to setup meaningful experiments. The easiest way to do this, is to generate the data yourself.

2. *real-world* dataset: You can find them at Kaggle Datasets the UCI ML repository at UCI, etc. Top Sources for ML Datasets gives more information.

.

4 Research question examples

You are asked to apply ML-algorithms for classification and regression. Here, we give some examples of straightforward research questions:

1. How do the ML-algorithms compare in terms of in-sample error and out of sample error on the data sets used.

2. What is the impact of mislabeled training examples/outliers[2] on the performance?

[2] A training example is mislabeled when it is put in the wrong class when labeling the training set.

3. What is the impact of class unbalances on the performance? In many cases we have about the same number of examples of each class. In some applications this is not the case, e.g. credit card fraud detection, intrusion detection, etc.

4. What happens when classes overlap? How does that affect the performance? Which transformations to the feature space work best?

5. In case of SVM, what is the impact of class overlap, class unbalance and mislabeled data on the number and location of support vectors?

Note that synthetic datasets are most useful to investigate the latter questions. You can control the percentage of mislabeled data, the class unbalance and the class overlap.

5 Evaluation

Your project will be evaluated based on the Python notebook you submit and the 15 minute presentation you give at a time slot during the exam period. Your work will be evaluated based on the originality and motivation of the research questions, the rigour with which you setup your experiments, the quality of your report and visualisations, and the description of the machine learning algorithms you used. You will also be evaluated on your presentation, taking into account the clarity, correctness of your presentation, and your ability to answer our questions after the presentation.

Appendix

A PLA - example description

We explain the perceptron learning algorithm (PLA).
Let

$$\{(\mathbf{x}_n, y_n) : \mathbf{x}_n \in \mathbb{R}^{d+1} \quad and \quad y_n \in \{-1, 1\}\}$$

be the training set, $n = 1, \cdots, N$. The *Perceptron Learning Algorithm* (PLA) is a classification algorithm that searches the hypothesis space

$$\mathcal{H}_{PLA} = \{h \in \mathcal{H} : h(x) = sign(\mathbf{w}^T\mathbf{x})\}$$

for a hypothesis $g(\mathbf{x})$ that classifies all $\mathbf{x}_n$ correctly. Note that we use the dummy feature $x_0 = 1$. The real features are $x_1, \cdots, x_d$. As a result the hyperplane $h(\mathbf{w})$ goes through the origin $\mathbf{0}$ and is completely characterized by the weight vector $\mathbf{w}$ orthogonal to that hyperplane.

The final hypothesis $g(\mathbf{x})$ is obtained by iteratively updating $\mathbf{w}$ until all training examples are classified correctly. When the training set is *linear separable* this is always possible.

PLA proceeds as follows. First, initialize the weight vector: $\mathbf{w} \leftarrow \mathbf{w}_0$. Next, select a misclassified point $\mathbf{x}_k$ and update the weight vector

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_k \mathbf{x}_k$$

until all examples are classified correctly, the parameter $\eta > 0$ is the learning rate. The number of iterations needed depends on the learning rate $\eta > 0$. The parameter η can be tuned using validation.

A variant called the pocket algorithm is used when the training set is not linear separable. The number of iterations N is determined beforehand and the best hypothesis evaluated during the run is returned. The best hypothesis minimizes the number of misclassifications.