SFML-Projects

Bernard Manderick

1 Introduction

Important Note: Only **2** (instead of 3 as stated before) ML-algorithms for classification and regression are **required**. However, if you applied more algorithms that will be taken into account.

For the course *Statistical Foundations of Machine Learning* students will be marked on 2 projects submitted as Python Jupyter notebooks not later than Sunday, 7 June, 2020.

In this document I repeat *previous* information about the projects, Section 3. I provide *additional* information and I *answer questions* that I got. Please note that this document contains links, shown in blue, to additional information.

Basically, what is expected is an interactive research report in the form of *Python Jupyter Notebooks* addressing the assignments. *Research* meaning that you state a research question that you answer in the paper. It should be very clear *what* is the research question, *why* is it important, and *how* you answer that question. *Interactive* meaning that the reader can run your experiments. Python Jupyter notebooks provide a way to combine text, figures and code to run the experiments.

The notebooks have to be self-contained[1]. 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 even better they are imported directly from the website where you got the data sets.

The rest of this document is organized as follows. First, I repeat the information about the ML-assignments. Second, I review the submission modalities.

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

Third, I give some examples of research questions and descriptions of ML-algorithms. Finally, I answer some questions like

- Do we have to implement the ML-algorithms ourselves?
- Which ML-algorithms can we use?
- What data sets do we use and where can we get them?
- When and how to submit the projects?
- Do we have to visualize data and hypotheses?

2 Submission Modalities

You submit your projects in 1 mail to *Bernard.Manderick@vub.be*. The **deadline** is *Sunday, 7 June by midnight* for both VUB and ULB-students.

Use as *email subject*[2]

VUB-students *SFML: Project submission*.

ULB-students *INFO-F422: Project submission*.

Make sure to add your name (or names if you are working in a group) and the master program in which you are enrolled.

3 ML-projects

First I discuss the default option for one person followed by the same option for a group and an option that thesis students can choose if they interested.

Default option: You do two projects, one about classification and one about regression. You will submit one Jupyter notebook per project. Don't forget to mention the title of the project, your name(s) and master program(s) in which you are enrolled. For each project you use two datasets, a synthetic and a real-world one, to be used for training, validation and testing purposes. For both classification and regression you use at least **2** different ML-algorithms.

Default option for a group: You can work together with one or more people but then the work presented should be twice that of the default option for one person of you are working in a group of two, etc.

[2] Erasmus students are enrolled at either the ULB or VUB and use the corresponding subject.

Option for thesis students: If you are working on a master or PhD thesis then you can submit a ML project related to your thesis instead of the default option. However, you have to discuss this with me beforehand.

ML-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*. In any case it is import that you give a brief explanation of the ML-algorithm and the most important tunable parameters. In Section 4 example explanations of the perceptron learning algorithm and the support vector machine are given.

Data Sets

1. *synthetic* dataset: You can use the python module *scipy.stats* for that purpose. It is preferable but not necessary that you generate the synthetic data sets yourselves. The most important thing is that you can control the data generation process. If you generate the data yourself you have that control.

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.

Questions 1–4 in Section 5 also address algorithms and datasets.

4 Examples

We give some examples

- research questions
- explanations, one for the linear perceptron and another for the support vector machine, and
- description of use of an ML-library

4.1 Research Questions

You are asked to apply ML-algorithms for classification and regression. A straightforward research question is

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

But you can go further than that. Some examples are

2. What is the impact of mislabeled training examples/outliers[3] on the performance?

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?

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. Last but not least, when your data points are 2- or 3- dimensional you can visualize the dataset and the decision boundary or regression function.

4.2 ML-algorithms

4.2.1 Linear Perceptron Algorithm (*PLA*)

Let

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

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

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

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.

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

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

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_k \mathbf{x}_k \tag{1}$$

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.

4.2.2 Support Vector Machine

Let

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

be the training set, $n = 1, \cdots, N$. The hypothesis set of the basic *support vector machine* or *SVM* is

$$\mathcal{H}_{SVM} = \{h \in \mathcal{H} : h(\mathbf{x}) = sign(\mathbf{w}^\top \mathbf{x} + b)\}$$

where $\mathbf{w}$ is the weight vector *orthogonal* to the hyperplane $h(\mathbf{x}) = 0$ and b is the *intercept*, the intersection of the hyperplane with the y-axis.

The basic *SVM* a classification algorithm that assumes that the data are linearly separable. It returns the maximum margin hyperplane. This is the solution to the constrained optimization problem [1]:
Minimize the *objective function*

$$\frac{1}{2}\mathbf{w}^\top \mathbf{w} \tag{2}$$

subject to the *constraints*

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 0 \text{ for } n = 1, \cdots, N \tag{3}$$

This is an example of a *quadratic program* (QP). The objective function is quadratic and the constraints are linear.

The weight vector and the intercept of the final hypothesis $g(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ are given by

$$\mathbf{w} \quad = \quad \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n \tag{4}$$

$$y_s(\mathbf{w}^\top \mathbf{x}_s + b) \quad = \quad 1 \tag{5}$$

where $\mathbf{x}_s$ is any of the support vectors.

The coefficients $\alpha_n \geq 0$ in Eq. (5) are returned by the QP-solver. Most $\alpha_n = 0$. When $\alpha_n > 0$, the corresponding $\mathbf{x}_n$ is called a *support vector*. Eq. (5) gives the constraint to be satisfied by the corresponding support vector $\mathbf{x}_s$ and gives the intercept b.

The basic *SVM* can be extended in 2 ways.

A first way is to apply a *non-linear transformation* $\Phi : X \to Z : \mathbf{x} \to \mathbf{z} = \Phi(\mathbf{x})$ from the input space to the feature space. The maximum margin separation is done in Z.

SVM uses the *kernel trick* to accomplish this. A kernel function $K : X \times X \to \mathbb{R}$ is positive definite: $K(\mathbf{x}, \mathbf{x}^{'}) > 0$ for all non-zero vectors $\mathbf{x}, \mathbf{x}^{'} \in X$. A kernel K induces a transformation Φ_K to a feature space Z_K but both are left implicit.

The polynomial kernel

$$K_{poly}(\mathbf{x}, \mathbf{x}^{'}) = \left(\mathbf{x}^\top \mathbf{x}^{'} + 1\right)^d$$

where d is a positive integer, and the radial basis kernel

$$K_{rb}(\mathbf{x}, \mathbf{x}^{'}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}^{'}\|^2\right)$$

where $\gamma > 0$ is a real constant, are examples.

The final hypothesis when using kernel K is

$$g(\mathbf{x}) = sign(\sum_{\alpha_n > 0} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b) \tag{6}$$

where $b = y_s - \sum_{\alpha_n > 0} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}_s)$ for any of the support vectors $\mathbf{x}_s$, i.e. $\alpha_s > 0$.

Note that although the maximum margin separation takes place in the feature space Z_K all computations in Eq. (6) are done in the input space X.

The second extension is to allow misclassifications. Each $\mathbf{x}_n$ has an associated slack variable $\xi_n \geq 0$ that measures the degree of misclassification: if $\xi_n > 0$, $\mathbf{x}$ is misclassified, otherwise $\mathbf{x}$ is classified correctly and $\xi_n = 0$. The original optimization problem defined by Eqs.(2)-(3) becomes:

Minimize the *objective function*

$$\frac{1}{2}\mathbf{w}^\top \mathbf{w} + C \sum_{n=1}^{N} \xi_n \tag{7}$$

subject to the *constraints*

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \quad \geq \quad 1 - \xi_n \tag{8}$$

$$\xi_n \geq 0 \tag{9}$$

for $n = 1, \cdots, N$.

The extra term in Eq. (7) is the product of the *box constraint* $C > 0$, a user defined parameter, and the total amount of misclassification. When $C = 0$ no misclassifications are allowed.

This time the *QP*-solver returns $\alpha_n, n = 1, \cdots, N$, such that $0 \leq \alpha_n \leq C$. Compared to the basic *SVM*, the extra condition is that $\alpha_n \leq C$.

4.2.3 Running the Support Vector Machine

We used *SVM* from the *skilearn*-module, cf. SVM from scikit-learn. In order to run *SVM* we have to specify the kernel function K and the box constraint C.

SVM take as input two arrays: the data matrix X and the class labels $\mathbf{y}$.

Learning is done as follows:

$>$ *from sklearn import svm*

$>$ *clf = svm.SVC()*

$>$ *clf.fit(X, $\mathbf{y}$)*

After the learning process, the model can then be used to predict y_{new} for $\mathbf{x}_{new}$

$>$ *clf.predict($\mathbf{x}_{new}$)*

5 Questions

Below you will find a number of questions that have been asked by students.

ML-Algorithms

Q: Can we use ML-algorithms not seen during the course?

A: Yes, you can use any algorithm you are interested in. But you have to give a short description of the algorithm used and the parameters that affect its performance, cf. Examples 4.2.1 and 4.2.2.

ML-Implementation

Q: Do we have to implement the ML-algorithms ourselves?

A: No, you can use any library that you like. But explain how that library has to be installed of it is not part of the *Python Ecosystem for Machine Learning*. The library *sklearn* is part, others like *Theano* and *PyTorch* are not. Also explain the use, cf. Examples 4.2.3.

Synthetic Data Sets The purpose is

Q: Do we have to generate the synthetic data sets ourselves?

A: Preferably. It is important that you can control the data generation process. You can use other means as long as you have control. One can also generate test datasets with scikit-learn but I have no idea how much control that you have.

Real-World Data Sets

Q: Where can we find such data sets?

A: Kaggle, UCI ML Repository, Google provide data sets. For an overview, cf. Top Sources for ML Datasets. You can also use other sources.

Visualization

Q: Do we have to visualize the decision boundary of a classification algorithm or the regression function? sufficient?

A: Yes, in case of synthetic datasets. If your data points are 2- or 3-dimensional this is not difficult and it gives you insight in the performance of the ML-algorithm. The blog *Easily visualize Scikit-learn models decision boundaries* show how that can be done.

Submission The **deadline** is *Sunday, 7 June, by midnight* for both VUB and ULB-students.

Q: How do we submit our projects?

A: You submit your projects in 1 mail to *Bernard.Manderick@vub.be* before the deadline. You use as *email subject*[4]

VUB-students *SFML: Project submission*.

ULB-students *INFO-F422: Project submission*.

[4] Erasmus students are enrolled at either the ULB or VUB and use the corresponding subject.

References

[1] Stephen Boyd and Lieven Vandenberghe *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004