Machine Learning: Project 1 Report

Simon Perriard, Benoit Knuchel, Hédi Sassi
Department of Computer Science, EPFL, Switzerland

***Abstract*—For this project we were tasked with designing and testing diverse machine learning algorithms and train them to recognise Higg's Boson. We explored different models and training methods and came up with a solution reaching 82.8% accuracy.**

I. Introduction

The goal of this project was to get familiar with basic classification and regression techniques and apply them to a real-world problem: detecting Higg's Boson after a collision between two particles in CERN's LHC. We implemented the algorithm in python and used the numpy library. This project is related to the following aicrowd challenge : https://www.aicrowd.com/challenges/epfl-machine-learning-higgs-2019 . The predictions were verified by submitting a csv file containing the predictions on aicrowd.

II. Data Cleaning

Before training any model on the data, we first had to clean them. We noticed that there were lot's of null values (-999) and that they were appearing in certain columns depending on the "PRI_jet_num" value. We figured out that we could split the data into four different tables depending on the field "PRI_jet_num", taking values in the set $\{0, 1, 2, 3\}$. For each of these table, we trained a different model. This is justified by the CERN's experiment description that says that this field denotes the number of jets of particles that were observed during the collision and thus, depending on these jets, certain measures were null (field value at -999). Therefore, we made the decision to split the data according to these jet numbers and drop the columns that were filled with null values. This way, we limited the impact of these invalid values during the model's training.

Even after we removed the columns with the null values, some columns still had a small proportion of them. We decided to replace those missing values with the median of the corresponding columns. Thanks to that, we don't have to drop some of the lines and we can keep the maximum information possible.

III. Feature Expansion

We used polynomial feature expansion in order to improve our accuracy. We limited ourselves to polynomials of degrees 1 to 16 because we haven't seen any improvement after degree 16 (see Figure 1). We also tried to compute additional values as suggested in the data description given by CERN (Section A.3). These additional computations didn't give any results which means that the available data coupled to the polynomial features expansion already give enough information.

IV. Algorithms used

Even if this challenge is a classification problem, we use regression techniques (least-squares, ridge regression) as well as classification techniques (logistic regression, regularised logistic regression). We only used linear models for this project. In this section, we give some details about the different implementations.

A. Loss

We used the logistic loss and the MSE (mean squared error) loss functions for this project as they are convex functions that are easy to optimise. The MSE loss heavily benefits from the data cleaning since outliers have big impact on MSE loss.

B. Logistic regression

With this classification algorithm, we minimize the logistic loss using the gradient descent technique. We also wrote a regularized logistic regression that fights over-fitting by adding the squared norm of the weights vector to the logistic loss.

C. Least squares

Least squares computes a closed-form solution of our linear model. Since it tries to find an exact solution, it tends to over-fit. That is why we implemented a regularized version that penalizes heavy weights. We also have a least squares implementation that uses gradient descent and stochastic gradient descent to approximate the solution.

D. Cross-Validation

We tried different forms of cross-validations. Some are based on loss and others are based on accuracy. Normally the loss-based ones should give more information since they use continuous results (the loss is not approximated nor bounded) whereas the cross-validation based on accuracy uses discrete results (misclassified or not).
We also used different algorithms for the cross-validation. One is using a simple split of the training set into a training set and a validation set with respect to a given ratio. We also tried a modified version of the classical cross validation, that we called pseudo cross-validation, that directly outputs the

weight vector given by the model with the best parameters. Since it means training the model on a given part of that dataset, the training ratio has to be high (between 90 and 98%) for this model to be representative. The last form of cross validation that we implemented is the k-fold validation.

V. Results

Our models greatly benefited from the polynomial feature expansion. This allowed us to reach a better precision as the degree increased. However, as we can see on the graph, it is not useful to use degrees greater than 16. If we had increased the degrees further, we would have tended to over-fit.

We can see in the models comparisons that generally the pseudo cross-validation using testing accuracy gives better results than the other form of cross-validations and the pseudo cross-validation using the loss. This could be caused by the validation set being too small and causing the cross validation to be biased when using the loss. Using accuracy to determine the correctness of a model is less precise but outliers in the validation set don't have much impact on the overall accuracy. Also, by choosing a model based on its performance on a fraction of the training data can be relevant if the data are randomly sampled.

Model	Acc/Loss	Algo	Result
Logistic	Acc	pseudo CV	77.6
Logistic	Loss	pseudo CV	74.9
Log regul	Loss	pseudo CV	75.9
Log regul	Acc	pseudo CV	77.6
Least squares	Acc	pseudo CV	82.8
Least squares	Loss	pseudo CV	82.8
Least squares	Loss	4-fold	79.3
Least squares	Loss	classic CV	82.6
Ridge regression	Acc	pseudo CV	80.5
Ridge regression	Loss	classic CV	80.9
Ridge regression	Loss	4-fold	80.2

VI. Conclusion

We can see that the closed-form solution for least squares gives the best result using polynomial features expansion but the stochastic gradient descent least squared and gradient descent least squares give poor results (approximately 65% accuracy). We noticed that logistic regression tends to over-fit a bit and that may be why the regularized logistic regression performs a bit better. We also see that depending on the cross validation algorithm, we can have some changes in performance. Our pseudo cross-validation algorithm gives the best results for the closed-form least squares. It means that the test set as roughly the same distribution as the training set. If it was not the case, this pseudo cross-validation algorithm would had given bad results.

Overall, our best model reaches 82.8% of accuracy on the test dataset which is quite good. To reach a better accuracy, the use of different models (Neural net, SVM etc..) or a more in-depth feature expansion should be considered.

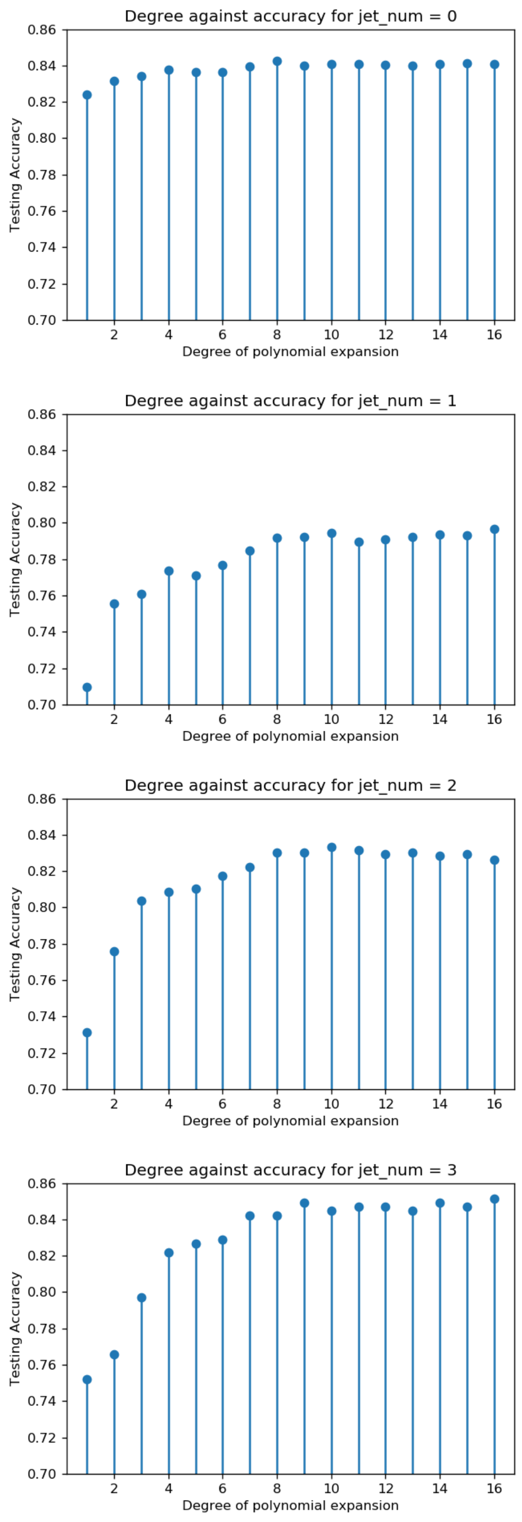

Figure 1: Cross-validation results for closed-form least square regression

References

project's github page is here (https://github.com/simon-perriard/ML_course_project1)