CS-433 Machine Learning - Project 1
Finding the Higgs Boson

Alix Jeannerot [1], Nicolas Lefebure [2], Samuel Dubuis [1]
[1]*School of Computer and Communication Sciences, EPF Lausanne, Switzerland*
[2]*School of Engineering, EPF Lausanne, Switzerland*

***Abstract*—This project presents a machine learning model to detect the presence of the Higg's boson in an experiment done by the CERN. Based on a data-set, we develop a ridge regression model to predict whether or not the production of a Higg's boson has been done.**

I. Introduction

Starting with a simple one-liner model of least squares we then clean the data-set and add some hyper-parameters to finally choose a ridge regression model. In the following section we show how we cleaned our data-set and expended our features. Then we will explain in details what our model is and how we choose our hyper-parameters. Finally we will compare our method to others and present some further improvement that could be made.

II. Chosen Model and Method

A. Cleaning the Data Set

When looking at the data-set we can clearly see that a lot of values are set to -999 meaning that there is no value for this parameter, the value is quite bothering as our model will try to take it into account whereas it should be ignored. However the full description of the project [1] can help us a little, indeed there exist a feature called **PRI_jet_num**. This variable represent the number of jet that were made but what is important is that it is the only one that is categorical as it take values in $\{0, 1, 2, 3\}$. Even more important, is that given the value of **PRI_jet_num** some other feature will be set to -999, it is therefore useful to split the data-set into four and remove some feature for each of the four. However the price to pay is have less data to train our models. This is summered in Table I

PRI_jet_num	Number of Features	Size of the Data Set
_	30	250000
0	18	99913
1	22	77544
2	29	50379
3	29	22164

TABLE I: Size of the feature and of the data set given the values of **PRI_jet_num**. _ represent the size before spliting

B. Choosing the Hyper-Parameters

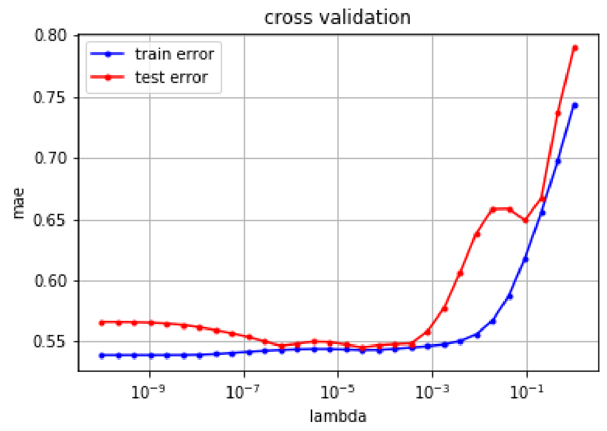
The least square method is too sensitive to overfitting in order to offer good predictions. The model of ridge regression is therefore a good improvement as the parameter λ will decrease the chance of choosing a overfitting weight vector. Furthermore to enrich our choice of model a polynomial expansion is done. We now have two hyper parameters that we need to tune to find the lowest mean absolute error (MAE). This is done by doing a 4-fold cross validation and iterating over several values and taking the parameters resulting in the lowest error. The chosen parameters are in Table II. The plots of the cross validation for the two first sub-dataset are visible in Figure 1, the two other plots are very similarly Figure 1(b), therefore they have been omitted. To choose them we have iterated over 30 different λ in the logspace between $1 \cdot 10^{-10}$ and 1 and for all degrees of expansion between 1 and 7 with the Numpy seed 5 for the randomness.

PRI_jet_num	λ	Degree of the polynomial expansion
0	$3.2903 \cdot 10^{-5}$	3
1	$1 \cdot 10^{-10}$	7
2	$1 \cdot 10^{-10}$	6
3	$1 \cdot 10^{-10}$	7

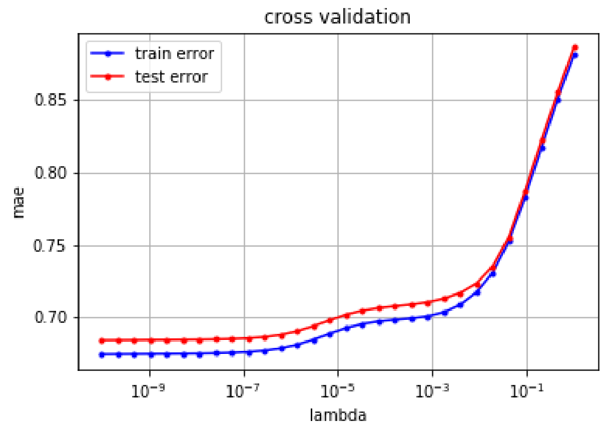
TABLE II: Parameters chosen to for the model

C. Outliers

As discussed before, there are values of the features in the data-set that are equal to -999. Part of them were rejected thanks to the splitting of the data-set. During the cross validation, we tried to minimise the effect of the remaining outliers by using the MAE instead of the mean squared error (MSE), knowing that MSE is not a good cost function when outliers are present. Indeed, the results were a little bit better: the score on AIcrowd rose up by 0.003. Looking again at the name of the features in the documentation we can realise that some, like **DER_mass_MMC** are defining a mass. As there no physical interpretation of a negative mass when one variable of these column is at -999 we can set it to 0. This reduced our loss during the cross validation and increased our score by 0.017.



(a) First sub-dataset, when **PRI_jet_num**$= 0$, degree $= 3$



(b) Second sub-dataset, when **PRI_jet_num**$= 1$, degree $= 7$

Fig. 1: Cross validation plots.

III. Results

The accuracy of our model based on optimisation of hyper-parameters for the ridge regression is 0.809. When using an MSI GL62M Laptop with a 2.5 Ghz Intel i5 Core CPU and with a 8GB of RAM, the training is done in less than twenty minutes. The Cross-validation denotes close values for mae, showing that the model should have a low bias and low variability. Thus we have a fast training method that can produce accurate models, which is useful if the data-set changes every now and then.

IV. Discussion

A. Comparison to other methods

As a result of our ridge regression, we also implemented a regularized logistic regression using a stochastic gradient descent. The SGD was set up with 50000 iterations, a decreasing step size (gamma was decreasing according to the iteration step, gamma = $\frac{1}{\sqrt{n}}$, allowing a closer approximation of the optimal solution since the step size needs to be reduced in order to be more accurate), a threshold over the rate of the loss function of 10^{-7} (to reduce computing time) and a mini-batch size of one. Similarly as for the rigid regression, we optimised the hyper-parameters (lambda and the polynomial degree expansion) using the cross-validation. The training was very long (lasted for several hours) and did not give much more interesting solutions (the best score on AIcrowd was 0.703). The long training time do not allow us to test various techniques to improve our predictions hence this is the main reason we did not investigate more with this model even though from a theoretical point of view it should be better than the one we selected.

B. Improvements

In general a deeper exploration of the data could be done in order to not only increase the feature via a polynomial expansion but with some other function or the product of several features.

For ridge regression, there are still hyper-parameters that were not optimized. For instance, we could increase the splitting parameter K-fold for the cross-validation and plot the bias-variance trade-off to help us choose a better model.

For the logistic regression, the main issue was that the training was too long. An improvement path would be to find ways to reduce the converging time of the stochastic gradient descent. Before optimising gamma and the polynomial degree, we could for example make a grid search on the number of iterations, the initial weights and functions for the step size with respect to time computation for a certain threshold of the loss function.

V. Conclusion

As a result, we can see that rigid regression is an efficient, fast and easy to implement method. Compared to our first submission based on a simple least squares (AIcrowd score of 0.745), this method allowed us to increase our score of 0.064, achieving a AIcrowd score of 0.809.

Acknowledgements

The authors thanks Martin Jaggi, Rudiger Urbanke and all the Teaching Assistants for their careful reading and helpful suggestions.