Higgs Boson Challenge

Ignacio Aleman, Thierry Bossy, Raphael Strebel
EPFL, Lausanne

***Abstract*—The Higgs Boson challenge is one of the most popular Machine Learning competitions. Based on simulations done in the ATLAS Experiments, the goal is to find out whether or not key signatures of the Higgs Boson are present. In this paper, we tackle this binary classification task by preprocessing the data and applying basic ML methods such as regression. The best solution we find consists of partitioning the input data according to one specific experiment measure and training independent models for each partition. From the results we conclude that this method performs fairly well. Indeed, we classify correctly around 83% of the test data.**

I. Introduction

In 1964, Peter Higgs – amongst others – theorized the existence of a mechanism that gave mass to matter, the Higgs mechanism. In turn, this mechanism implied the presence of the Higgs Boson. To confirm this – as well as other fundamental theories – CERN was established, and in 2012 the ATLAS and CMS both independently managed to show the presence of the Higgs Boson.

The idea was simple, to collide particles at high speeds and to capture key signatures during the collision. Then the question to be answered was, given the observed signatures, can we conclude the presence of the Higgs Boson?

The Higgs Boson Machine Learning Challenge was designed to classify the characterizing events in order to deretmine the presence of the tau decay of a Higgs Boson.

To get a better picture of the data, we went over the official review of the challenge [1] which gives useful information on the acquisition of the data. For example it explains why certain values are missing, which gives an idea as to how it might be worth to split the data. In the following we describe our approach to this challenge, starting with the preprocessing of the data. We then explain which models we choose and conclude by analysing the results we obtain. Finally we give an brief overview of other methods that could be tried to improve the performance of the model.

II. preprocessing

Preprocessing is an essential step that allows us to considerably improve the result given by our model. This is even more the case as we only used basic techniques in this project, as opposed to neural networks. By inspecting the shape of the data and the role of certain features, some steps can be applied before feeding our model with the data.

A. Jet number

From [1] we found out that the *jet_number*, a feature of the data, directly implies values of certain features of our data.

- **if jet num = 0**: PRI jet leading pt, PRI jet leading eta, PRI jet leading phi are undefined
- **if jet num $\leq$ 1**: DER deltaeta jet jet, DER mass jet jet, DER prodeta jet jet, DER lep eta centrality, PRI jet subleading pt, PRI jet subleading eta, PRI jet subleading phi are undefined.

This means that by splitting the input X on values of the jet number, we can remove certain columns of subsets of our data. As the jet number can only take values in the set $\{0, 1, 2, 3\}$, and as a value of 2 or 3 makes no difference on the implications of other features, we divide X into $\{X_0, X_1, X_{2,3}\}$. We then remove the features of those subsets accordingly, as well as the column that contains the jet number in all subsets of X since we no longer need it. See Figure 1 for a more visual explanation.

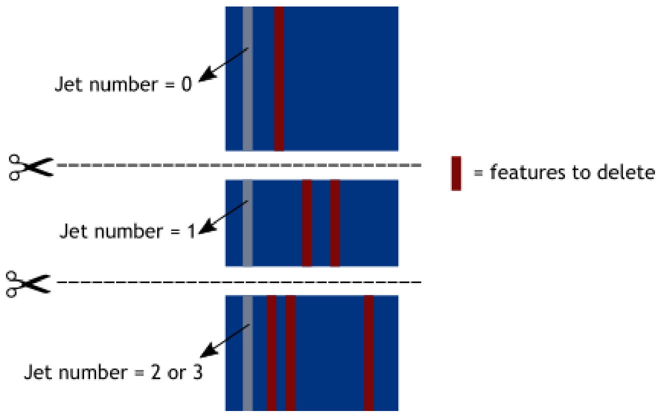

Figure 1. Splitting the data samples according to jet number

B. Undefined values

Probably because of erroneous measurements, the input still contains undefined values even after the previous simplification. To deal with this issue, we replace undefined values of a feature by the mean of the feature.

C. Standardisation

To standardise the features of the data, we determine the mean and the variance of every column, and normalize each column of the input by subtracting its mean and dividing by its standard deviation.

D. Feature augmentation

A challenge we face in this project is that we have a lot more data points than dimensions, which can easily lead to over-fitting. A natural response to this kind of problem is to increase the number of features by applying functions on the data. First of all, we add a bias 1 column to our input based on our intuition for simple linear models. In our case, we decided to compute polynomials of our data and consider them as new features. But considering a few polynomials does not nearly come close to the magnitude of the number of data points. For this reason it is better to also consider other approaches to augment the dimension of X. We continue the process by computing all cross-products of the columns of X and, as before, consider them as new features. This roughly squares the dimension of each subset X_i, which is already much better. Next, we apply a hyperbolic tangent function to our initial features so that the result is normalized in the domain $]-1,1[$ and does not penalize outliers too greatly as explained in [2]. The final data augmentation we do is to give the option of combining the previously described functions. Indeed, instead of computing only functions of the initial features, we can get a much larger dimensionality if we choose to compute functions of the previously determined features. For example, we could compute the polynomials of degrees 2 and 3 of an input X, then we can compute the $tanh$ of each column and append the result to the data. If we wanted to go even further, we could also compute the cross-product of all previously determined columns and consider them as part of the augmented data.

III. Models

We first tried both standard regression as well as logistic regression, and we found that using standard regression far outperformed logistic regression. We thus decided to go for standard regression, and to try and control over-fitting we choose to go for ridge regression (though we found that λ, the L_2 penalty, had little influence).

Thus, in essence our model selection process consisted in finding the right combination of feature augmentation. Recall that we split out data – based on the jet number – into 3 sets (X_0, X_1 and $X_{2,3}$).

For each subset of the data, we tried to find the right combination of feature augmentation (see I) to maximise the predictive accuracy.

Data	Degree	Bias	Cross	Tanh	Cumulative
X_0	[2, 3]	yes	yes	yes	no
X_1	[2]	yes	yes	yes	yes
$X_{2,3}$	[2]	yes	yes	yes	yes

Table I
TABLE OF CHOSEN DATA AUGMENTATION PER DATA SPLIT

IV. Results

With the aforementioned model we found an accuracy of 0.829 on the test set via the AI crowd portal. Before evaluating on the challenge test set, we split our own train data into a resp. test/split set to check the accuracy. We performed 10 random splits to get an idea of the variance of our accuracy. We describe the result – also per jet num – in Table II. We see that our model performs quite well for jet num 0 and 2, 3, but not so well for jet num 1.

Dataset	Accuracy
X_0	0.840 +- 0.006
X_1	0.819 +- 0.002
$X_{2,3}$	0.851 +- 0.002
X	0.837 +- 0.003

Table II
MODEL ACCURACY PER DATA SPLIT

V. Discussion

Surprisingly a linear model worked better than a fancier logistic regression. However, we did not try to optimise the data augmentation for the logistic model, so the aforementioned claim should be taken with a grain of salt.

Apart from feature augmentation, we also tried to perform PCA on the data, however we did not find any sort of gain and therefore did not follow this direction further.

Clearly finding the right and large enough feature augmentation technique gave the largest improvement in accuracy.

VI. Future work

In line with the previous discussion, the authors believe that a more through search of feature augmentation should further allow to increase the accuracy. (e.g. Kernel methods)

References

[1] C. G. I. G. B. K. D. R. Claire Adam-Bourdariosa, Glen Cowanb, "Learning to discover: the higgs boson machine learning challenge," 2004, https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf.

[2] A. F. Kalay, "Preprocessing for neural networks - normalization techniques," 2018, https://alfurka.github.io/2018-11-10-preprocessing-for-nn/.