Learning to discover: the Higgs boson machine learning challenge

Oriol Barbany*, Natalia Gullon† and Sophia Kypraiou‡
Machine Learning (CS-433), School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne
Email: *oriol.barbanymayor@epfl.ch, †natalia.gullonaltes@epfl.ch, ‡sofia.kypraiou@epfl.ch

***Abstract*—Machine learning techniques have been successfully applied to a bunch of problems from different disciplines. In most cases where data is too complex or too large to be treated, statistical models are needed and their results yield to the state of the art. This paper introduces a machine learning method to predict if a certain event corresponds to the decayment of the Higgs boson given features extracted from the ATLAS experiment performed at the Large Hadron Collider at CERN. The proposed algorithm is based on regularized logistic regression with adaptive learning rate, and obtained a categorization accuracy of 0.82 in the Kaggle platform of the challenge[1].**

I. Introduction

The Higgs boson is an elementary particle in the Standard Model of physics that explains why other particles have mass. In order to discover this particle, protons are smashed one into another at high speeds to generate smaller particles as by-products of collisions. Decay signature and other products that result from its decay process are measured and the likelihood that a given event's signature is the result of a Higgs boson is then estimated.

Therefore, the aim of this project is to build a binary classifier to predict whether an event corresponds to the decayment of a Higgs boson or not from a given vector of features representing the decay signature of a collision event.

II. Methodology

A. Data cleaning

Most of the time spent on this project is dedicated to the preprocessing of the raw data. The features of the different events corresponding to experiments performed at ATLAS have a lot of meaning-less values, meaning that the obtained results are outside the normal range of a given variable. The first approach is, therefore, to remove all features with such values. Nevertheless, as stated in [1], depending on the number of jets of the event, some features can be undefined. Thus, we filter the raw data by the jet number and then eliminate all meaning-less values, which, in most cases, are all the events of a given feature. Depending on the topology of the event, if the event is not too far from the expected topology, a feature representing the estimated mass of the Higgs boson candidate could be also defined. Given that this last feature is thought to be meaningful if defined, we also filter the dataset between valid estimated mass or not.

[1] https://www.kaggle.com/c/epfml18-higgs

Doing a exploratory analysis of every chunk of data, we find that a lot of values could be defined as outliers following the outer fence approach presented in [2]. This approach is based in determining the inter-quartile range IQR of the data separated by features, i.e. the difference between the third and first quartiles, and then setting expected minimum and maximum values from that. All the data points that lay outside this range are considered outliers. Then, the outer fence defines the following limits:

$$\min = Q_1 - 3IQR \quad ; \quad \min = Q_3 + 3IQR \tag{1}$$

It turns out that if we explore the outliers by features, they represent a very low percentage by mere definition. However, if we eliminate an event that has outliers, we would end up with only around 20% of the original dataset. In order to avoid introducing fake data, we do not change the value of an outlier candidate such as the median or the nearest quartile.

For linear models (e.g. linear regression or logistic regression), multicolinearity can yield to solutions that are wildly varying and possibly numerically unstable. Therefore, as part of the data analysis, we compute the similarity of the features using the Pearson correlation coefficient by measuring the linear correlation between the variables. Unfortunately, although 3 of the features (DER_sum_pt, PRI_met_sumet, PRI_jet_all_pt [1]) are highly correlated between them (with value ≥ 0.96), removing one or more of these features does not show any significant improvement on our results. This shows us the difference between correlation and causation.

Another approach regarding the data preprocessing is to build polynomials with cross terms. That is because it is very likely that the theoretical value of the existence of the Higgs boson decaying into tau particles is based on combining some parameters, as in almost all physics models. Therefore, we create powers of a certain degree for every feature, as usual in feature expansion, but we also incorporate sums, products and squares of products for cross terms of the generated polynomial. This ends up with a lot of new features which results in better models.

Applying dimensionality reduction with Principal Component Analysis (PCA) to our data might also improve the results of our model. Nevertheless, given that the polynomial with cross terms obtains better results and does not seem to overfit, even if it has some hundreds of features, it is no longer needed to apply this transformation.

B. Model

Different models are tested to tackle our classification problem using different algorithms to optimize a loss function: Gradient Descent (GD), Stochastic Gradient Descent (SGD), least squares, ridge regression and logistic regression.

For our final model, we use logistic regression because it is optimized for binary classification and we add a penalty factor λ in order to avoid over-fitting.

We should remark that we also add a little modification to the original loss function for logistic regression in order to avoid computational overflow. The sigmoid function (Equation 2) only takes values of 0 or 1 in the limits to $\pm\infty$. However, due to limitations of float type, when applying the sigmoid to very large or small numbers we get an approximation of 1 or 0. That produces a problem when computing the logarithm in the loss function. Therefore, we add a threshold in the input of the function in order to ensure a floating-point results and the loss is computed following Equation 4.

$$\sigma(t) = \frac{1}{1 + \mathrm{e}^{-t}} \tag{2}$$

$$\mathcal{L}(w) = -\sum_{n=1}^{N} [y_n \log(\sigma(\max\{x_n^T w, -10\})) + \tag{3}$$

$$+ (1 - y_n) \log(1 - \sigma(\min\{x_n^T, 10\}))] \tag{4}$$

C. Cross-validation

In order to tune the hyper-parameters of our model, k-fold cross-validation is used with $k = 10$ folds. The train dataset is randomly chopped into 10 equal sized sets. We tuned the values of the polynomial degree, in both cases where it was simple or with cross terms, and the penalty factor λ. The best parameters are then selected by performing grid search for all parameters and choosing those that yield to the highest average accuracy across folds. This algorithm is used separately for each of the 4 partitions of the train dataset according to the number of JET feature and each of them is also divided in 2 depending on the presence of the mass value, as explained in Section II-A. That makes a total of 8 partitions with different hyper-parameters to tune.

The accuracies for the best parameters obtained with cross-validation are illustrated in Table I.

JET	With mass	Best degree	Best λ	Accuracy
0	True	7	10^{-5}	0.797 ± 0.004
0	False	12	10^{-5}	0.986 ± 0.003
1	True	9	10^{-4}	0.739 ± 0.005
1	False	10	10^{-6}	0.977 ± 0.007
2	True	8	10^{-5}	0.557 ± 0.006
2	False	9	10^{-10}	0.892 ± 0.016
3	True	9	10^{-4}	0.832 ± 0.008
3	False	9	10^{-2}	0.975 ± 0.011

TABLE I
BEST PARAMETERS OBTAINED WITH CROSS-VALIDATION

III. Experiments and results

In Table II we can see the different performances of some of the models tested. We added a random guess model as a reference to see the improvement of each model. All those models have a previous data cleaning that only consisted in removing the features with meaning-less values.

- **Model A:** Random guess (reference)
- **Model B:** Gradient Descent (MSE loss function)
- **Model C:** Stochastic Gradient Descent (MSE loss function)
- **Model D:** Least squares
- **Model E:** Ridge regression
- **Model F:** Regularized logistic regression
- **Model G:** Regularized logistic regression with JET and mass filtering and adaptive learning rate

Model	Accuracy
A	0.500 ± 0.001
B	0.706 ± 0.002
C	0.709 ± 0.003
D	0.774 ± 0.001
E	0.767 ± 0.001
F	0.743 ± 0.001
G	0.823 ± 0.004

TABLE II
PERFORMANCE WITH DIFFERENT MODELS

As we can see, the best model in terms of the obtained accuracy, is the regularized logistic regression with Gradient Descent optimization and adaptive learning rate. Regarding the approach to update the learning rate, we decay by a γ factor after a given number of iterations. This is inspired by the Step Learning rate Scheduler of Pytorch[3].

The decaying factor of the learning rate is set to $\gamma = 0.1$, which is the default value provided by the Pytorch library, and it is updated every 2000 iterations. This last value is set based on the evolution of the loss function across iterations to avoid having a very large step size at the first iterations or very few improvements in the last ones.

IV. Conclusions

With this project, we show the importance of understanding the data and doing a proper preprocessing before applying machine learning algorithms to build a statistical model. Our problem has a lot of non valid values which are finally treated by filtering the data of certain events that condition the meaningfulness of others. In this dataset, the number of outliers is bigger than in other problems and, given that these are often non-overlapping, we are not able to eliminate them.

Regarding the best obtained model, it is trivial to see that the logistic loss makes much more sense in a binary classification framework than other usual metrics like the mean-square error, which do not penalize the miss-classification but only the distance to the ground-truth. The source code of this project is available on GitHub[2].

[2] https://github.com/Barbany/ML-Project1

REFERENCES

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau, "The higgs boson machine learning challenge," in *Proceedings of the 2014 International Conference on High-Energy Physics and Machine Learning - Volume 42*, ser. HEPML'14. JMLR.org, 2014, pp. 19–55. [Online]. Available: http://dl.acm.org/citation.cfm?id=2996850.2996852

[2] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.

[3] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.