Using Machine Learning to Detect the Higgs Boson Particle

Farah Charab, Lorenzo Lazzara, Andrea Manzini
EPFL, School of Computer and Communication Sciences

***Abstract*—The ongoing advances in the area of Machine Learning have played a central role in many of the technological advances that we heavily rely on nowadays. To name a few, Machine learning lies at the core of the (internet) searches that we perform these days, speech recognition and machine translation, classifying DNA sequences, self-driving cars, and so on. The aim of this project is to use Machine Learning to help us better understand one of the greatest discovery of particle physics, the Higgs Boson particle. Namely, we apply machine learning techniques to actual data from CERN particle accelerator to recreate the process of discovering the Higgs particle.**

I. Introduction

The discovery of the Higgs Boson particle at the Large Hadron Collider at CERN in 2013 resolved a long standing open question that was considered “the central problem in particle physics”. The collision experiments at CERN that yielded this breakthrough result generated a huge amount of data, that requires computer-aided computation in order to understand it. In this project, we study the resulting data sets that were generated in order to learn how to distinguish the Higgs Boson particle from the background noise. In mathematical terms, the problem boils down to the following: given some (primitive and derived) measurements of a certain particle, decide whether it is a Higgs Boson particle or just background noise. In this project, we implement various algorithms in order to perform this task. We carefully study our data sets and the simulations of these algorithms in order to tweak their corresponding parameters, and thus maximize the accuracy of our classifier.

The remainder of this report is organized as follows: We first start by stating some preliminary results we got by running the different algorithms on the data sets provided. Based on the results we obtained and the data statistics, we propose different methods to clean up and augment the data sets in order to further improve the accuracy of our classification. We conclude the study with the model we have chosen and propose some further directions to take.

II. Machine Learning Algorithms

A. Preliminary Results

We provide basic implementations for Least Squares (LS), Ridge Regression (RR), Gradient Descent (GD) , Stochastic Gradient Descent (SGD), Logistic Regression (LR) and Penalized Logistic Regression (PLR) algorithms; and we present the results of the simulations of these algorithms on the training data set provided without any preprocessing. The results of our findings are tabulated in Table I.

For the LS and RR methods, we use an 8-fold cross validation to calculate the parameters and the prediction scores. For the remaining algorithms, we did not do cross validation since we were not able to get anything better than a 64% prediction score regardless of the choice of the hyperparameters.

In Table I, λ is used to denote the penalty parameter, γ is used to denote the step size used by the gradient descent algorithms, and the last two columns present the mean and the standard deviation of the prediction score across the different folds, respectively.

Algorithm	λ	γ	n_iter	$\bar{s}$	σ_s
Least Squares	-	-	-	0.7442	0.0024
Ridge Regression	$1.2 \cdot 10^{-6}$	-	-	0.7443	0.0023
Gradient Descent	-	10^{-9}	200	0.6398	-
Stochastic GD	-	10^{-9}	200	0.6403	-
Logistic Regression	-	-	-	≈ 0.6 [1]	-
Penalized LR	-	-	-	≈ 0.6	-

Table I: Accuracy of the Various Algorithms on the Raw Data Sets.

B. Analyzing the Data

As mentioned in the project description, a special value that appears in both the training and testing data is the value -999. This value is outside the normal range of all variables, and can either mean that the variable is meaningless or cannot be computed. We present in Table II statistics regarding the occurrences of this special value in the training data set, where we only depict the features in which it appears in.

#	Name	overall %	% in b	% in s
0	DER_mass_MMC	15.24	15.92	15.34
4	DER_deltaeta_jet_jet	70.98	70.90	71.13
5	DER_mass_jet_jet	70.98	70.90	71.13
6	DER_prodeta_jet_jet	70.98	70.90	71.13
12	DER_lep_eta_centrality	70.98	39.93	71.13
23	PRI_jet_leading_pt	39.96	39.93	40.02
24	PRI_jet_leading_eta	39.96	39.93	40.02
25	PRI_jet_leading_phi	39.96	39.93	40.02
26	PRI_jet_subleading_pt	70.98	70.90	71.13
27	PRI_jet_subleading_eta	70.98	70.90	71.13
28	PRI_jet_subleading_phi	70.98	70.90	71.13

Table II: Data Analysis

As expected, the presence of -999 value in a primitive feature highly correlates with the presence of the same

[1] We no longer have the exact numbers for LR and PLR; and due to their low accuracy, we proceeded with other algorithms.

value for a feature derived from it. For instance, features {`26,27,28`} are missing in 70.9% of the rows, and whenever these features are missing, the derived features {`4,5,6,12`} are also missing. The remaining 19 features not present in the table do not have any -999 value.

There are several approaches that one can potentially use in order to deal with these values. We tried removing the columns that have a high percentage of -999's (70%) in the training data from both training & testing data sets, however this decreased the accuracy of our learning algorithms. Furthermore, increasing the threshold beyond 70.9% leaves the data intact as can be seen in Table II. Thus, we decided to abandon the filtering idea.

A parallel approach is the following: consider a feature $i \in [30]$, and let μ_i, μ_i^b, μ_i^s denote the mean of all the values different from -999 for feature i, the mean of all such values when restricted to rows classified as `b`, and the mean of all such values when restricted to rows classified as `s`, respectively. Then one can preprocess the data in one of the following ways.

- For every data point in the training data set and for every feature i, we replace all of -999 values with μ_i^b (μ_i^s) if it is classified as `b` (`s`), respectively. Furthermore, every -999 value in the test data can then be replaced by $\frac{\mu_i^b+\mu_i^s}{2}$, or by μ_i.
- For every data point in both the training and testing data, and for every feature i, we can replace all of -999 values with μ_i.

The same approaches can be carried out by taking the medians $median_i, median_i^b$ and $median_i^s$ instead of the means.

We used an 8-fold cross validation to test these approaches. We tabulate the results obtained for both LS and RR algorithms in Table III, since these were performing better than the remaining algorithms implemented [2]. Given

Heuristic	Least Square		Ridge Regression	
	%	std. dev	%	std. dev
No preprocessing	74.42	0.002	74.43	0.002
μ_i^b, μ_i^s	87.29	0.001	87.29	0.001
μ_i	71.87	0.002	82.17	0.002
$median_i^b, median_i^s$	90.36	0.0008	90.36	0.0008
$median_i$	74.41	0.001	74.43	0.001

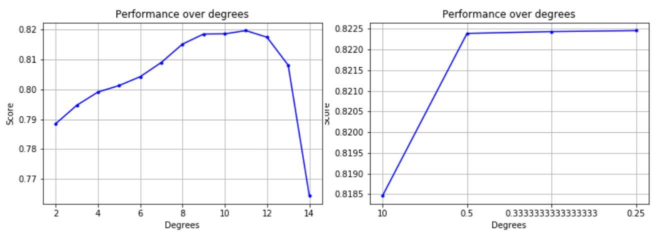
Table III: Performance of LS and RR for various data processing heuristics

that RR is a generalization of LS and given that these algorithms were performing better than the other methods, we decided to further investigate RR.

C. Adding Features

The other approach of preprocessing that we adapted is feature augmentation. In particular, we tried to extract some *meaningful* correlation from the data by adding cross terms, ratios, and polynomial expansions. We elaborate more on that in what follows:

- Cross terms (680 features): After trying different combinations we decided to add cross products, their square root, and their square. Given that taking the cross terms of n features grows quadratically with n, we decided to only take cross products among the primitive features and not the derived ones. The assumption was that derived quantities, which were specifically introduced by physicists, should have a meaning as they are, whereas primitive features could naturally be correlated. The scores gained by some quick tests seem to validate our assumption.
- Polynomial terms (360 features): We used polynomial augmentation in order to adapt our model to non-linear dependencies. In particular, for every feature x, we added x^i for $i \in \{1/2, 0, 2, 3, \ldots, 10\}$. The choice of the degrees is justified by Figure (1)



(a) Adding Integer Degrees (b) Adding fractional degrees

Figure 1: Chosen Model: Degree Vs Score

D. Model and Discussion

After considering different options, we decided to choose Ridge Regression, since RR had a superior performance compared to the other algorithms. For $\lambda = 3.1e^{-7}$, we obtain a prediction score of 82% when using an 8-fold cross validation and when running our classifier on the test data provided. These results are attained when using the data augmentation proposed in section C along with the third clean up method proposed in Table III.

III. Conclusion

Based on the results presented in Table III, we propose further investigating the second and third data processing heuristics. Note, however, that we expect to have a worse performance on the test data since the scores presented in the table aren't very realistic. This is because these scores were obtained by running cross validation on the processed train data. That is, we didn't separately process the fold we are testing our algorithms on as we would process our test data normally[3].

[2]Note that we omit specifying the λ corresponding to the rigid regression as it is optimized according to the preprocessing heuristic specified.

[3]Due to time constraints