Higgs Boson Machine Learning Challenge

Zhang Yuyao[1], Xu Yitao[2], Han Chenyu[2]
1 School of Mathematics, EPFL, Switzerland
2 School of Computer and Communication Science, EPFL, Switzerland

***Abstract*—In this project, we use six machine learning models to predict whether a recorded event indicates the Higgs Boson's decay or not, which is a binary classification problem. We divide the data into three groups according to the value of a specific feature and train different models on each group. We implement multiple feature engineering methods to improve our results, including data cleaning and feature augmentation. We run thorough experiments to find the optimal parameter combinations and achieve a classification accuracy of 0.831 and an F1 score of 0.742 on the leaderboard.**

I. Introduction

Higgs Boson is an elementary particle theorized roughly 50 years ago and was said to give mass to other elementary particles[1]. Researchers generate it by proton-proton collisions but decay rapidly and thus are not observable. One way to detect it is to use the decay signature of the collision event.
The dataset contains 250000 events with 30 features. The label is binary, and either s for Higgs Boson signal or b for background. We first divide the data into Three groups according to the value of feature $PRI\ jet\ num$ since multiple features are undefined when $PRI\ jet\ num$ takes specific values. Then we clean the data after observing the distribution of data. We implement six machine learning models to perform the classification task and further augment feature space to increase the representational power of our models. We run thorough experiments on each combination of different tricks and parameters and find the best setting with a 5-fold cross-validation. We achieved a classification accuracy of 0.831 and an F1 score of 0.742 on the Kaggle leaderboard, which is tested by a more extensive test set containing more than 500000 samples

II. Methods

A. Data grouping

There are many weird values in the given dataset, i.e., values of -999.0 (~1.5M in 7.5M). A large number of that specific value indicates that it does not simply represent missing values. After examining the B part of the appendix of [1], we found that many features are undefined under certain values of a key feature: PRI_jet_num. This feature can take four values (0,1,2,3). When it takes 0,1, or $>= 2$ values, the types of undefined features vary. Therefore, instead of simply filling the seemingly missing value -999.0 with the column mean or median, we first group data points according to the value of PRI_jet_num being 0,1, or $>= 2$, resulting in 3 groups. Then, we drop the undefined features in each group, which causes the different number of features in different groups. Moreover, we exclude the PRI_jet_num feature in each group since it has been used for grouping. Only when $PRI\_jet\_num >= 2$ are all feature well-defined. Therefore, we need to train our model separately on those 3 groups.

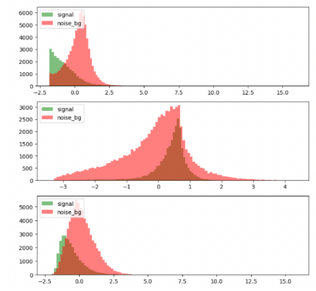

Figure 1. The distribution of each feature. We can see that features are tend to be at center.

B. Data cleaning

We find that in [1], it is said that DER_mass_MMC may be undefined under certain circumstances. This kind of problem can only be treated as missing values, and we choose to fill in the missing values of DER_mass_MMC with the median value of this feature.
The graphs shows that each feature is more concentrated in the middle. We assume that some values will influence the model with a high difference from the mean of the data. Therefore, we do a data cleaning using the n-sigma principle, assuming that they are in normal distributions.
By the probabilistic definition, the 3-sigma interval contains 99.7% of the data, while 1-sigma contains only 66.7%. We skipped the 2-sigma because the difference between it and 3-sigma is relatively small(95%) and found that the performance improved by 1% by changing from 3-sigma to 1-sigma way on the training set, which proves our assumption.

C. Models

We implement 6 models in total, as described below:

1) **MSE+GD**. Gradient descent of mean squared loss.

2) **MSE+SGD**. Stochastic gradient descent of mean squared loss.
3) **Least squares**. Using normal equation to obtain a closed-form solution of linear regression.
4) **Ridge regression**. Normal equation with weight L2-norm penalty.
5) **Logistic regression**. Gradient descent with binary cross-entropy loss.
6) **Regularized logistic regression**. Gradient descent with binary cross-entropy loss plus a weight L2-norm penalty term.

For the first four methods, we use a binary label of $[-1, 1]$ and predict label 1 for output > 0.5 whereas 0 for output $<= 0.5$. For those two logistic regression methods, we modify the label to $[0, 1]$ for modeling the probability of a given input belonging to a certain class, and also use 0.5 as threshold for predicting 0 ($<= 0.5$) or 1(> 0.5).

D. Feature normalization and augmentation

All our models use simple linear combinations of the input features, which might result in insufficient representational power. Therefore, we extend the feature using a polynomial basis function. The degree is a critical parameter to tune and we run thorough experiments on deciding the optimal degree to use.
Another observation is that different features in the dataset can have vastly different range. It can cause that the same step size in different directions in the feature space can have drastically different effects. Therefore, we consider feature normalization after polynomial feature augmentation. The normalization is done on each feature by equation 1:

$$X = \frac{X - min(X)}{max(X) - min(X)} \tag{1}$$

, where X is one feature column in the dataset.

E. Model selection

Each model has its unique parameter setting. After incorporating all the aforementioned tricks, we can obtain a large number of different parameter combinations for each model. Since it is unclear whether a particular trick can be helpful for a model, we need to traverse all possibilities to decide which is the best parameter setting for every model we consider. To compare the performance and decide the best parameter combination scheme, we run a 5-fold cross-validation with each parameter setting and record the average classification accuracy across all 5 test sets. Finally, we pick the parameter combination with the highest average classification accuracy as the best one.

III. Results

The results are in table I. We can see that ridge regression and least squares achieve the best result among all models. We prefer ridge regression since it is numerically more stable than least squares. We use the best parameter combinations for ridge regression to train on the whole dataset and get 0.831 accuracy and 0.742 F1 scores on the leaderboard. The best settings of other models are in our experiment notebooks.

	group0	group1	group2
MSE+GD	83.79	78.03	80.16
MSE+SGD	82.88	74.22	73.91
Least squares	84.53	80.82	83.55
Ridge regression	84.53	80.82	83.55
Logistic regression	83.65	76.31	77.62
Reg logistic regression	83.58	78.32	79.87

Table I
Best performance of each model in different data groups.

IV. Analysis of results

A. Ridge regression v.s. logistic(reg) regression

We can see from the results that ridge regression's performance greatly exceeds both logistic and regularized logistic regression. In theory, ridge regression calculates the weights by the normal equation and guarantees an optimal solution, while logistic regression needs to perform gradient descent to achieve it. Therefore, we think this may be due to the choice of the maximum iteration and the learning rate. We generated a series of learning rates from 0.01 to 0.1 to train logistic regression with a maximum iteration of 5000 and discovered that the higher the learning rate, the higher the performance is (83.12 to 83.58 in group0), so it seems that it hasn't converged to the optimum yet. We can fix this by having dynamic control over the learning rate, but comparing to get the optimal solution immediately, it is low in efficiency.

B. Non-linearity of the dataset

We achieve our best result with degree 9 of polynomial basis, indicating that the underlying function is highly non-linear. Given more time, we could try more complex model such as neural network to better model the non-linearity in the dataset.

V. Summary

In this project, we build six models to learn the relationship between an event and the occurrence Higgs Boson. We group the data according to PRI_jet_num and run thorough experiments to find the best combinations of tricks and parameters with 5-fold cross-validation, including degree of polynomial basis, data cleaning, feature normalization, and different models. We find ridge regression to be the best among all models and analyze the reason why the ridge regression can become the best. We achieve a classification accuracy of 0.831 and an F1 score of 0.742 on the Kaggle leaderboard.

References

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kegl, and D. Rousseau, “Learning to discover: the higgs boson machine learning challenge,” *URL http://higgsml. lal. in2p3. fr/documentation*, vol. 9, 2014.