The Higgs boson Machine Learning Challenge

Emanuele Nevali, Matteo Suez, Leonardo Bruno Trentini
Machine Learning and Optimization Laboratory, EPFL, Switzerland

***Abstract*—In this project, carried out as part of the 2022 Machine Learnng course at EPFL, we applied machine learning techniques to actual CERN particle accelerator data to recreate the process of "discovering" the Higgs particle.**

I. Introduction

Rarely, collisions between protons at high speed can produce a Higgs boson. Since the Higgs boson decays rapidly into other particles, it is not possible to observe it directly, but rather measure its "decay signature", or the products that result from its decay process. The aim of this project is to estimate the likelihood that a given event's signature was the result of a Higgs boson (signal) or some other process/particle (background).

After analysis and preprocessing of the dataset, we introduce the model built with the 6 demanded algorithms and we explain how we managed to choose the best hyperparameters. In conclusion, we present our results.

II. Data analysis and preprocessing

A. Exploratory data analysis

Our dataset contains 250000 points for training and 568238 for testing with their corresponding binary label ("s" for "signal" and "b" for "background", which have to be predicted for the test set values). All of these points are characterized by 30 different features.

B. Categorization

First of all, we notice that there is only one categorical feature, i.e. *Pri jet num*: it shows the number of jets (see [1] for a physical explanation), and it is an integer that ranges from 0 to 3.

We noticed that some features are meaningless for some values of jets, therefore we have split the dataset into 4 subclasses, each one characterized by a different *Pri jet num*.

C. Missing Values

Thanks to the documentation, we noticed that each "-999" value in the dataset consists in a missing value: we decided to replace all of them with the median of the feature. In particular, we chose to use the median instead of the mean because of its nature of robust estimator to outliers.

D. Heavy-tailed features

Noticing that many of the features were heavy-tailed, we computed for these a logaritmic transformation: $x_k^* = log(1 + x_k)$ (where k is the feature column). These transformation helps in reducing or removing the skewness of our original data.

E. Absolute transform of Symmetrical Features

For features whose distribution is symmetric around 0 and for which the proportion of categories is not too even (in particular, features in columns [14, 17, 23, 26]) , we calculated and replaced with the absolute value.

This increases the gap between the two categories, and makes them better distinguishable by the model.

F. Outliers erasure

After a search for outliers in the data, we set $\alpha = 0.1$ and decided to edge the extreme values of each feature to the α-quantile (for the lower tail) and to the $(1 - \alpha)$-quantile (for the upper tail).

G. Standardization

In order to ensure a better functioning of our algorithms, we decided to standardize the data by subtracting from each feature its mean and dividing by its standard deviation: $Z = \frac{X-\mu}{\sigma}$. The same process was applied both to the train set and to the test set.

H. Polynomial Expansion

We tried also to improve our predictions by expanding the features In particular, we augmented features by implementing a polynomial expansion for each column of the form $\phi : [X] \to [X, X^2, ..., X^D]$.

The polynomial degree D is one of the hyper-parameters of our model: we found its optimal value by performing a 4-fold cross validation on each subset.

III. Methods

After the pre-processing on the dataset, we managed to find a model for our problem by implementing six different methods:

- **Gradient Descent**
- **Stochastic Gradient Descent**
- **Least squares** using normal equations
- **Ridge regression** using normal equations

- **Logistic regression** using gradient descent or stochastic gradient descent ($y \in 0, 1$)
- **Regularized logistic regression** using gradient descent or stochastic gradient descent ($y \in 0, 1$, with regularization term $\lambda \|\omega\|^2$)

All functions return: (w, loss), which is the last weight vector of the method, and the corresponding loss value (or cost function).

Optimal parameters λ (regularization to reduce overfitting), γ (step size for gradient and stochastic gradient descent) and D (degrees for polynomial expansion) were chosen through a 4-fold cross validation for each subset.

IV. Results

bla bla bla
bla bla bla

V. Conclusions

bla bla bla
bla bla bla

References

[1] C. Adam-Bourdariosa, G. Cowanb, C. Germain, I. Guyond, B. Kegl, and D. Rousseau, "Learning to discover: the higgs boson machine learning challenge," *Higgs Challenge*, 2014.