Tracing the signatures of Higgs Boson

Ioannis Bantzis
EPFL
ioannis.bantzis@epfl.ch

Manos Chatzakis
EPFL
emmanouil.chatzakis@epfl.ch

Maxence Hofer
EPFL
maxence.hofer.epfl.ch

Abstract—**In this work, we present a machine learning binary classifier for the CERN signature dataset, which works by separately training sets of input data and then combining the results to predict if a given signature belongs to a Higgs Boson or not. Our experimental evaluation over this method shows that we are able to achieve up to XXXX accuracy for input data through cross validation, and up to XXXX accuracy for Kaggle competition.**

I. Introduction

Motivation. Tracing the presence of Higgs boson after proton collisions is one of the most prominent research directions of CERN, and has provided a rich database of particle footprints, by gathering data of their decaying signatures. By examining those signatures, we can determine if the collision generated a Higgs boson. In this work, we provide a binary classification protype, able to predict if a particle signature corresponds to a Higgs boson.

Challenge. In this context, there are multiple challenges we needed to overcome; The data are dense, and thus we needed to preprocess them carefully and using different methods, by observing their structure. Furthermore, as there are multiple regression models available, we had to implement different methods and evaluate them. Some models may perform better in some context, others may not. add add add

Contributions. Our contribution can be summarized as follows:

- We implement different regression models for the dataset
- We develop a method that trains the data in different parts in order to achieve higher accuracy
- We present an experimental evaluation about the performance of the regression models over the CERN signature dataset, which shows that our method that partially trains the data can achieve up to XXXX accuracy through cross validation and up to XXXX on Kaggle competition.

The rest of this work is organized as follows: Section I is the introduction, Section 2 presents the regression models, Section 3 shows how we preprocessed the data, Section 4 describes our training method, Section 5 is the experimental evaluation and Section 6 is the conclusion.

II. Models

In this section we briefly describe the regression models we used, providing information about how the optimal weghts are obtained.

Mean Squared Error with Gradient Descent (MSE_GD). (MSE_GD) finds the optimal weights: $w^{(t+1)} = w^{(t)} - \gamma \nabla \mathcal{L}(w^{(t)})$, where w is the weight vector, $\mathcal{L}$ is the loss function, γ is the step size and t is the current iteration.

Mean Squared Error with Stochastic Gradient Descent (MSE_SGD). Just like MSE_GD, MSE_SGD find the optimal weights by using one point of the dataset each time (batch size is 1), using $w^{(t+1)} = w^{(t)} - \gamma \nabla \mathcal{L}_n(w^{(t)})$, where w is the weight vector, $\mathcal{L}_n$ is the loss function for a sampled input point n, γ is the step size and t is the current iteration.

Least squares (LS). LS calculates the optimal weights using $w^* = \left(X^\top X\right)^{-1} X^\top y$ where w^* is the optimal weights vector, X is the feature matrix and y the label vector.

Ridge regression (RR). RR calculates the optimal weights using $w^* = \left(X^\top X + 2N\lambda I\right)^{-1} X^\top y$ where w^* is the optimal weights vector, X is the feature matrix, y is the label vector, I is the identity matrix and λ is the trade-off parameter.

Logistic regression (LR).

Regularized logistic regression (RLR).

III. Preprocessing

Before applying the different models, as defined in section II, some preprocessing operations will need to be applied on the data. First, we observed that the feature PRI_jet_num was an integer between 0 and 3. We then separated the data in four different sub-data set according to their PRI_jet_num value. Then, we needed to get rid of the wrong values, given as -999.0 in the data. Theses values were present only in certain columns, 11 for the matrix with PRI_jet_num=0, (10 full of false values and the first column with 26123/99913), 8 for PRI_jet_num=1 (7 full and the first with 7562/77544), and just the first column for PRI_jet_num=2, 3, respectively with 2952/50379 and 1477/22164 false values. As there was completely false column for PRI_jet_num=0, 1, the idea was to remove these columns. For the column with only some false values, we implemented two methods: one that removes the column and one that replaces the -999.0 by the average of the other values in the corresponding column. Furthermore, a constant feature was added to the data (a column containing only 1). Finally, we expanded the features up to a certain degree d. This means we created a matrix x, which has a size d times the size of x, and corresponding to {x, x^2, ..., x^d}. This allows to get non-linear relations between y and x.

IV. Results

In this section, we present our experimental evaluation. We extensively evaluate the implemented regression models and our final approach. We present how we tuned the parameters of the models, we compare models and approaches and lastly, we present our Kaggle competition results. We conduct our experimental evaluation on a machine of M1-pro chip, 10 cores, 16 GB of memory, running MacOS 12.6.

Parameter Tuning. We used XXXX-fold cross validation to find the best-fitting parameters for our models.

Model Comparison. After parameter tuning, we compare the available models by calculating their accuracy.

Aicrowd Competition. Aicrowd hosts a competition website were different binary classification approaches are compared in a test dataset, calculating their prediction power. By uploading our final approach, we got an accuracy score of XXXX which is ranked among the top-XXXX scores among over XXXX submissions.

V. Conclusion

We presented a machine learning prototype able to perform binary classification over the CERN signature dataset in order to predict if a particle is a Higgs Boson using their collision decay signatures. Our experimental evaluation shows that our protype is able to achieve XXXX accuracy using cross validation and XXXX accurace in the Kaggle competition, which ranks it among the best implementations.