Finding the Higgs Boson

APOSTOLOV Alexander (alexander.apostolov@epfl.ch)
BAUM Auguste (auguste.baum@epfl.ch)
CHRAIBI Ghali (ghali.chraibi@epfl.ch)

CS-433 Machine Learning - October 2020, EPFL, Switzerland

***Abstract*—In this paper, we use linear classification and regression models to analyse Higgs boson data published by CERN. After exploratory analysis and testing different models, we succeeded at building a ridge regression model achieving 80% accuracy. This was done by separating the data into different parts according to categorical variables.**

I. Introduction

The Higgs boson in an elementary particle discovered in 2013 explaining why other particles have mass. This particle is not directly observable through experimentation, but one can still measure its "decay signature" (i.e. the products that result from its decay process). However, it is hard to distinguish between the decay signature of a Higgs boson and those of other particles.

Therefore, our goal is to use binary classification techniques to predict whether a given event's signature originates from a Higgs boson or not.

II. Models and Methods

A. Algorithms

We first implemented many of the classic algorithms to solve a binary classification problem in order to compare how well they perform on our dataset. These are:

- Least Square (LS), using Gradient Descent (GD), Stochastic Gradient Descent (SGD) and its closed-form equation.
- Ridge Regression (closed-form equation of LS with a regularization term).
- Logistic Regression using SGD.
- Regularized Logistic Regression using SGD.

B. Data Processing

1) Data label: The given labels take values in {-1, 1}, but it is more convenient to have labels in {0, 1} for (regularized) logistic regression. We therefore map all the -1 to 0.

2) Dealing with invalid values: The raw dataset contains many invalid values encoded as -999.0. We observed that all those were concentrated in specific features (11 features out of the 30). We considered two ways of dealing with these values :

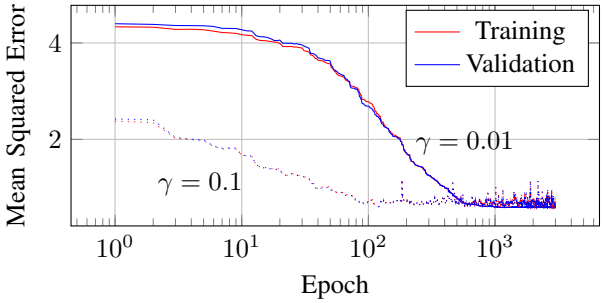

Figure 1: Convergence of regularized logistic regression for $\lambda = 0.1$ and two different values of γ.

- Drop the features containing invalid values (as they can represent up to 70% of the data for those features).
- Replace the invalid values by the mean of the corresponding feature.

3) Feature expansion: To produce a more flexible model, we use polynomial expansion (without interaction between features).

4) Normalization: Finally we normalized the data so that each feature has the same scale in order to achieve better convergence.

C. Cross-validation

We used 4-Fold cross validation with grid search to select the hyperparameters (linear space for degree of polynomial expansion ad logarithmic space for the learning rate γ and the regularisation term λ) for each algorithm we considered. Cross-validation was run only on 80% of the available training data in order to use the remaining 20% exclusively for comparing the algorithms once optimally tuned.

To choose the number of iterations, we plotted the training and validation errors at each epoch for all the algorithms and some hyperparameters. One of these plots can be seen in Figure 1. The errors seemed to have converged after 1000 iterations, so we decided to run all our grid-searches with a conservative value of 10 000 iterations.

This proved to be a difficult decision since setting a lower learning rate can lead to better results, if the number of iterations is high enough. However, increasing the number of iterations can be time-consuming due to the number of combinations of hyperparameters.

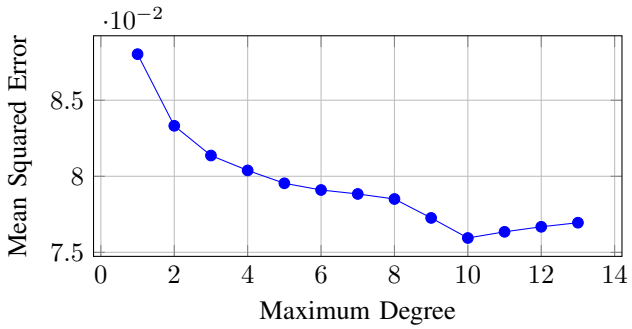

Figure 2: Validation error depending on degree of polynomial using closed-form least squares weights; adding more monomial terms leads to overfitting.

During cross-validation we could observe that increasing the degree of the polynomial expansion leads to overfitting, as can be seen in Figure 2.

We chose the better hyperparameters according to the combination which yields the smallest average validation loss across the 4 folds.

D. First Model

For our first model, we took the dataset without the columns containing invalid values and trained it using all the different algorithms mentioned above.

E. Second Model

For the second model, we took the dataset where we replaced the invalid values by the mean of their corresponding feature and trained it using all the different algorithms described in subsection II-A.

F. Third Model

For the last model, we divided our dataset into four parts according to the four categories of the unique categorical variable, PRI_jet_num. We then removed the columns containing invalid values in each dataset. We think that training 4 different models and applying the corresponding one according to the value of PRI_jet_num can increase predictibility.

	LS	Ridge	LS SGD	Logistic	Regularised Logistic
NaN removed	0.790	0.776	0.706	0.690	0.695
NaN replaced	0.815	0.806	0.734	0.714	0.714
Categorised & NaN removed	—	0.792	—	0.711	0.709

Table I: Best accuracy on test set for each data model.

III. Results

After grid-searching for the best hyperparameters, it appears that ridge regression achieves the best results among all algorithms when used on the third model based on our accuracy and the results from AICrowd.

Category	0	1	2	3
Max degree	14	15	12	15

Table II: Parameters used to train the ridge regression that produced 79.2% accuracy (cf. Table I, third line). The value of λ used was 10^{-8}.

The parameters used for each category are shown in Table II. Using this model, we achieved an accuracy of 0.794 on AICrowd (ID: 92621).

IV. Discussion

There are a number of free variables that can influence the performance of the training. The learning algorithm, the learning rate and regularisation (when required), the maximum degree of the polynomial expansion...

This complexity lead us to make choices based on intuition; namely, which bounds and precision to use for the grid searches. Indeed, it is very possible that, with a lower learning rate and different regularisations (provided that convergence is reached fast enough), we could have achieved better results. Unfortunately, compromises had to be made with respect to the time-performance trade-off, which may have lead to us to miss very high-performing combinations of parameters.

Similarly, feature expansion was performed to a lesser degree than what might have been required: we decided not to include interactions terms in order to limit the number of features and the effect of multicollinearity, and did not use any other nonlinear transformation such as $\log$, $\exp$ or $\sin$. In fact, doing this would probably allow us to lower the degree which is quite high (cf. Table II).

In the third model, further work can be done to train the 4 different submodels in order to use different algorithms and expansion per submodel. Additionally, we could also have replaced invalid values by the column means, instead of removing them. This further processing might yield better results, according to the second line of Table I.

V. Conclusion

This project was a very good practical introduction to Machine Learning. We managed to apply the techniques seen in class to obtain a model that was satisfactory, even though the breadth of our explorations remained limited. The biggest take-away from this project would be not to neglect exploration of the data to see if nonlinear relationships can be extracted. That way, nonlinear-looking data can be brought back to a linear setting, where our methods can be applied to their full potential.