CS433 - Machine Learning - Project 1

Bastien Bernath & Etienne Boisson & Xavier Theimer-Lienhard
IC Faculty, EPFL, Switzerland

***Abstract*—This paper reports our work on the Higgs Boson Challenge that consist on predicting whether an event was either signal (a Higgs Boson) or background (something else). This work was done in the context of a project for the Machine Learning course at EPFL in 2022.**

I. Introduction

The Higgs Boson is an elementary particle which explains why particles have a mass. We can never observe it, because its lifespan is too short. However when it decays it produces other particles which we can observe. Other processes can produce the same particles as a Boson with a certain probability so we can never be sure from one measurement if the particles we measure come from a Boson or not. Therefore, our goal in this exercise is to predict, from the large data-set that the CERN team created, the likelihood that the particles we measure come from a decaying Boson or from another process.

II. Exploration and pre-processing of data

The first thing we did was to explore the dataset which was composed of 30 features. Those features were either physics measurements or features obtained from combining those measurements. By exploring the data set we discovered that:

- A lot of the data was set to -999 which represents data measurements that were unavailable.
- Feature 21 isn't continuous, all it's values are integers.
- Some features from the original data set are highly co-linearly dependent.
- The data set is unbalanced; the ratio of positive versus negative labels is one half.

Using what we discovered from our data, we created a cleaner and more useful dataset than the original one.

A. Dealing with missing values (-999 values)

The first step was to remove the -999 values from our data set. In the previous step, we saw that a lot of the -999 values were located on the same features. We therefore decided that if a feature had more than 20% of -999 values, we would remove the feature from the dataset. For the other -999 values, we replaced them with the median value of the whole feature. This process removed 10 features from the original dataset.

B. Standardising the data

The next step was to standardize the data. We took that decision because our features were not using the same units and were not on the same order of magnitude. We subtracted the mean of the feature and then divided it by the standard deviation of the feature. In doing so, it makes our features comparable to each other.

C. Dealing with correlated features

Correlated features can be redundant and lead to numerical instability. It's better to use features that are not correlated with one-another when learning regression models. As we saw in the exploration of our dataset, multiple features are correlated. To palliate this issue, we removed features which had more than a specific threshold correlation with another feature. To find the correlation between features, we used the Pearson correlation coefficient. We used cross validation to find the threshold which minimized the most the loss. The best threshold we found was to remove one of the two features when they have at least 50% of correlation. We implemented that method in the `clean_standardize()` method in the `helper.py` file.

D. Dealing with biased dataset

While working on the dataset we realised that a model that always predicted "it's not a Boson" for every sample had a 67% accuracy. This comes from the fact that there is approximately two third of the data that corresponds to a non-Boson signature. In order to deal with this, we implemented this in the `clean_standardize()` method in the `helper.py` file, that balance the dataset. We balanced the data set by removing randomly some non-Boson signature inputs to match with the number of Boson signature. Unfortunately, we'll see that this new idea to pre-process our data wasn't helpful.

E. Polynomial expansion of the features

A standard method for learning better models is to expand the features using polynomial expansion. We implemented it in `helper.py`. The polynomial expansion method also adds the bias column on the data set. This pre-processing step works really well on Least Squares and Ridge Regression methods as we use high degree polynomial expansion in order to obtain the best results. On the other hand, the methods using the gradient descent strategy offer better results when not using polynomial expansion (but still adding the bias column).

F. *Transformation of the labels for logistic regression*

To match our labels to the Sigmoid function used in the two logistic regression methods we have to switch label encoding from -1,1 to 0,1.

III. Dataset creation

In order to see the effect of the steps of our pre-processing, we decided to create three different data sets with three different pipelines of pre-processing. In all three data sets we standardized the data, removed the features with too many missing values (-999 values) and added a bias term.

A. *Dataset I*

- Removing the columns with more than 20% of -999 values
- Standardization of the data
- Addition of a bias column

B. *Dataset II*

- Same pre-processing steps as the first dataset
- Removing features that have more than 50% of correlation with another feature
- Doing a polynomial expansion for Ridge Regression and Least Squares

C. *Dataset III*

- Same pre-processing as the second dataset
- Balancing the data§set to have an equal proportion of Boson signatures and non-Boson signatures

IV. Finding the perfect hyper-parameters

A. *Naive implementations of machine learning methods*

At first, we tested a lot of different configurations, exploring our dataset and parameter choices with our implementations.

B. *Using cross validation*

From this, we executed all the cross validation pipeline using 5-fold on the *Dataset II* using these different machine learning algorithms: **GD, SGD, LS, RR, LR, RLR**[1]. This allowed us to isolate the best values for the parameters such as the learning coefficient γ, the regularisation factor λ, the polynomial expansion *degree* and the correlation suppression *threshold*.

GD & SGD: We did cross validation for degrees [1,2,3] and many different γ: ranging from 0.0001 to 1. We found, for Gradient Descent: best degree: 1, best γ: 0.04,
for Stochastic Gradient Descent: best degree: 1, best γ: 0.009

LS: We did cross validation for degrees ranging from 1 to 10. Best degree: 6

RR: We noticed that ridge regression worked better than the other methods, so we did a deeper cross validation on it. We tested λ ranging between [10^{-7}, 1], and degrees ranging between [1, 20]. Best λ : $5e^{-5}$ and best degree: 15. We also cross validated our pre-processing of correlated features threshold, we removed at first the correlated features that had more than 20% correlation, and then we tried all the values ranging up to no features removed and found that a 50% threshold was optimal.

LR & RLR: We changed the label encoding and did cross validation for degrees [1,2,3] and many different γ: ranging from 0.0001 to 1. For Regularized Logistic Regression the λ was ranging between [10^{-3}, $5e^{-1}$]. For Logistic Regression, best degree was 1 and best γ was 0.008. For Regularized logistic regression, best degree was 1, best γ was 0.085 and best λ was 10^{-3}

Dataset	GD	SGD	LS	RR	LR	RLR
I	72.15%	68.09%	73.39%	72.57%	71.68%	66.66%
II	71.91%	72.24%	77.58%	79.80%	72.30%	72.21%
III	70.97%	69.44%	76.08%	77.21%	70.99%	71.20%

Table I
ACCURACY OF IMPLEMENTATIONS FOR OUR 3 DIFFERENT DATA PRE-PROCESSING, OBTAINED ON OUR CROSS-VALIDATION, THE RESULTS ON AICROWD ARE SIMILAR

V. Discussion, Conclusion and Improvements

A. *Discussion:*

We can see that there is a significant improvement between data pre-processing I and II, but dataset III performed worse. This might be because we removed a third of the data set and removing the bias didn't compensate this loss. We were also surprised that ridge regression achieved the lowest loss for polynomial expansion of degree 15.

B. *Conclusion:*

To conclude, we succeeded in building a model that predict the Boson signature with 79.8% of accuracy with a ridge regression with polynomial expansion of degree 15. All our methods even achieved to go over the 70% accuracy on the *Dataset II*. We built different datasets in order to cover a wide range of possibility for our methods.

C. *Improvements:*

We noted that our **RLR** performs a little bit worse than the **LR**. For an improvement we could implement a larger cross validation for this method. In order the improve the results of such a challenge, it might be a good choice to test other methods than regression, like Neural Networks, or trying better regression models such as Support Vector Machines. Additionally, some more pre-processing steps on the data, such as removing outliers or trigonometric expansion on the angle features, could help obtaining slightly better results.

D. *Acknowledgement:*

We want to thank the CS-433 teaching team for their help.

[1]GD: Gradient Descent, SGD: Stochastic Gradient Descent, LS: Least Squares, RR: Ridge Regression, LR: Logistic Regression, RLR: Regularized Logistic Regression