CS-433 Machine Learning: Project 1 report

Davide Nanni, Edoardo Debenedetti, Mari Sofie Lerfaldet
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—The Higgs boson challenge is based on the ATLAS and CMS experiment that claimed the discovery of the Higgs boson [1]. This report describes our attempt to tackle this challenge as the first project in the Machine Learning course CS-433 taught at EPFL. Our best result was achieved by using the Ridge Regression method, with a categorical accuracy of 0.807 on AIcrowd.**

I. Introduction

Our objective in this project is to predict simulated particle collision events as either a Higgs boson signal or background decay (i.e. whether the decay was into a fermion pair or not). This is done by performing a first data analysis step and then by training different machine learning models of both regression and classification. By cleaning the dataset, tuning the training parameters and comparing the different methods, the aim is to reach the highest possible accuracy.

II. Models and Methods

A. Implementation of Machine Learning Models

We implemented six machine learning models, as required by the project description: Least Squares Gradient Descent, Least Squares Stochastic Gradient Descent, Least Squares (with normal equations), Ridge Regression, Logistic Regression with Stochastic Gradient Descent and Regularized Logistic Regression with Gradient Descent.

Additionally, we have also implemented Regularized Logistic Regression with Newton Method.

B. Exploratory data analysis and feature processing

The training and test sets are characterized by 30 features and contain 250,000 and 568,238 events respectively. All features are of float type, except for PRI_jet_num, which is of integer type. Missing values are represented by -999.0. The label we need to predict is either 'b' (i.e. background event) or 's' (i.e. signal event). The categories are well balanced, as their proportion is of 34.3% and 65.7% each.

1) Dealing with missing values: There are a total of 38,114 missing values in the feature DER_mass_MMC in the training set. According to the challenge documentation [1], the mass is labeled -999.0 if the topology of the event is too far from the expected topology. To replace the missing values, we used the median of the feature, since it is more robust than the mean, which is vulnerable to outliers.

Moreover, if we consider only the subset of data in which the mass value is missing, we noticed that the number of signal events is only 8% of the total, significantly less than the original 34.3%. We thought that the fact that the mass is missing can be meaningful and useful for the classification, which would be coherent with the aforementioned reason why the mass has been marked as missing. Therefore, we decided to add a column of 0 or 1 integer values: 1 if the mass was missing and 0 if it was not. There were also missing values for features related to the jet number of the event, and we are discussing this topic in the next section.

2) Categorization: Regarding the other missing values, the challenge description [1] is quite clear: the missing values depend on the number of jets of the event (i.e. the PRI_jet_num column). The number of jets is an integer between 0 and 3:

- If it is 0, a specific set $\mathcal{S}$ of the features presents missing values.
- If it is 1, only a specific subset $\mathcal{S}' \subset \mathcal{S}$ of the features presents missing values.
- If it is either 2 or 3, there are no missing values.

We then tried to approach this subdivision in two ways. One by mapping all -999.0 values into 0s in the columns belonging to $\mathcal{S}$, then creating a new column containing the row-wise sum of the columns belonging to $\mathcal{S}$ and finally removing these columns. The other by splitting the dataset into three subsets (one with the events with jet number 0, one with jet number 1 and one with jet numbers 2 and 3) and then training three models, one for each subset, since the three subsets have a different set of features. The second method performed better (67% accuracy vs 80%).

3) Feature selection: At this point, we used common feature selection methods to try and find a possibly better subset of features, as per Scikit-learn documentation [2]:

- Correlation: we found a high correlation between the features DER_sum_pt and PRI_met_sumet, however the removal of one of them did not improve the score.
- Low variance: no low variance features were found, thus no feature was removed at this point.
- Univariate Feature Selection: the purpose was to select only the N most important features and compare the result of the Logistic Regression on the sub-dataset, for N ranging from 4 to 10. The results however were always worse than the one given by the full dataset, thus no selection was carried out at this point.
- Recursive Feature Elimination: the results of this automated selection method were always worse compared to the one given by the full dataset, thus no selection was carried out at this point.

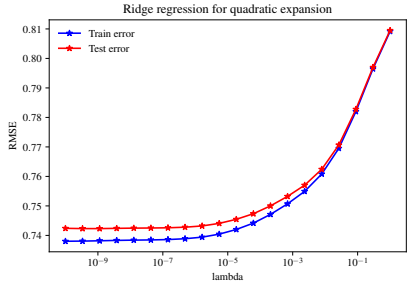

Figure 1. Lambda grid search

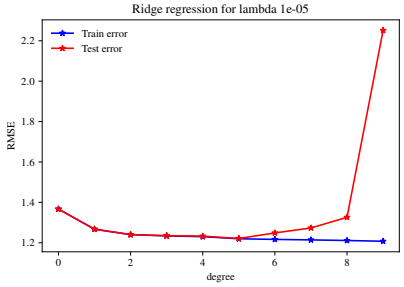

Figure 2. Degree grid search

4) Data transformation: We tried to improve our predictions by transforming and expanding the features in several ways. We first tried to augment features adding a polynomial basis of different degrees via *Vandermonde* matrices. However, we finally chose to use a quadratic expansion of the features consisting in adding a set of second degree features calculated by multiplying each original feature both with themselves and the others, as it turned out to have the best results. Visualizing the distributions of the features we noticed that many of them are heavy-tailed. We then shrank the support of their distributions by applying, feature-wise, the formula $n'_k = log(1 + |\min_{m \in k} m| + n)$, where k is the feature column. As a final step, we normalized all the features in the dataset.

C. Testing and Tuning

Given that the nature of the problem is inherently that of a classification problem, we first focused on tuning the Regularized Logistic Regression with SGD. However, we also performed multiple tests with Ridge Regression. Surprisingly, the latter outperformed the former. We then decided to tune Ridge Regression hyperparameters, such as the regularizer λ and the degree of the polynomial augmentation. We did it by using grid-search methods, looking for the best tradeoff between underfitting and overfitting.

As can be seen from Figure 2, the lowest RMSE turns out to be given by degree 5. However, since there is negligible difference among the degrees between 2 and 5, we decided to stick with degree 2 in order to keep the overall complexity of the model low. Next, as stated in II-B4, we tried using the combination of degree 2 of all the features, which gave us a better accuracy with cross-validation (81% vs 79%).

As for the λ, Figure 1 shows that smaller values cause smaller RMSE for both cross-validation training and test sets. We then chose to take $\lambda = 10^{-5}$.

III. Results

The final model is trained as follows:

- The mass missing values are treated as described in II-B1.
- Three different subsets of events are used (as in II-B2), and then three different models are trained.
- The logarithm of the features is taken into account (cfr. II-B4)
- The degree 2 polynomial expansion of the features is used.
- The training model is Ridge Regression with $\lambda = 10^{-5}$.

As a final result, we obtained a model that achieved an accuracy of 0.807 on AIcrowd and of 0.809 on local cross-validation test set.

IV. Discussion

Surprisingly, our tests with the Logistic Regression did not produce better results than Ridge Regression both locally and on AIcrowd validation, even training with cross-validation and a simpler train-test set subdivision. This could be due to the fact that Logistic Regression is the result of an iterative method and in order to reach a good accuracy we would have had to run SGD for more iterations (we tried with 10,000 and 50,000 iterations). On the other hand, Ridge Regression simply solves a linear system, and does not have to solve an optimization problem.

We have also encountered some unexpected behaviours in tuning hyperparameters for Ridge Regression. In fact, in some cases, with high polynomial degrees (e.g. 12) and cross-validation, both test and training set accuracies were growing, while we were expecting the test set accuracy to start decreasing after a certain polynomial degree. However, the test set RMSE was behaving as expected (increasing after degree 5), and then we chose to keep a low degree. We also tuned the lambda with both low (2) and high (12) polynomial degrees, and we noticed that with high degrees RMSE increased and accuracy decreased faster than with low degrees. We interpreted it as due to overfitting.

V. Summary

For this project we used machine learning to make predictions of simulated particle collision events, and whether they were a Higgs boson signal or a background decays. Important steps in the process were dealing with missing values, sub-categorization, data transformation and testing/tuning the models. By comparing the Regularized Logistic Regression with SDG with Ridge Regression, it became evident that the latter was better suited for the task.

REFERENCES

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, K. Balázs, and D. Rousseau, "Learning to discover: the higgs boson machine learning challenge," 2014. [Online]. Available: https://higgsml.lal.in2p3.fr/files/2014/04/documentation_v1.8.pdf

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.