EPFL Machine Learning CS-433 - Project 1

WANG Qimin
School of Architecture, Civil and Environmental Engineering
Email: qimin.wang@epfl.ch

CUI Mingbo
School of Microengineering Robotics
Email: mingbo.cui@epfl.ch

HUANG Ruibin
School of Computer and Communication Sciences
Email: ruibin.huang@epfl.ch

***Abstract*—This report describes the complete model building process and the findings of Project One of Machine Learning course (CS-433) in EPFL. The goal of the project is to apply machine learning concepts on a real-world dataset which is from kaggle competition by CERN. In this report, six different machine learning methods were implemented to build different models. Comparing the models by cross-validation, the best model was chosen to be the baseline of the project. To improve the prediction results, the model was evaluated again after applying exploratory data analysis, feature processing and data cleaning. Repeating the above steps, we obtained a satisfactory model and also meaningful findings.**

INTRODUCTION

In this project, we intended to use machine learning algorithms to detect Higgs boson particle. 250,000 labeled records with 30 initial features (leaving alone the No. and the label) are provided, while 568,238 unlabeled records for prediction. Starting from apply simple machine learning algorithm on the original features, we could only get an unsatisfied accuracy, which resulted most likely from that the dataset was not well exploited. Then we paid more attention to preprocessing data and enriching features, and finally resulted much better.

I. DATA ANALYSIS AND FEATURE ENGINEERING

Taking the data from training dataset as the example, we found that 11 features have the outlier of -999, which has the mean of data (represented with the diamond point) far away from the main continuous range,as shown in Fig.1. After replacing the outliers with the mean of each feature, Pearson correlation coefficient of the 30 features with the label ($s :\rightarrow 1$ and $b :\rightarrow -1$) is presented in Fig.2, which is our primary criteria for feature selection.

To put the feature value into a controllable and reasonable range, we took the log value instead, and took the cosine value for several angle-like features, by judging from the features names with endings like '-phi' and '-eta'.

We also introduced features with specific physics meanings [1] into our model. The new features were created using the particle mass as the weight, multiplied by the quotients of the two other features.

For the final submission, the new features were standardized and normalized using sigmoid function. All these processes can be referred to in *feature_selection.py*

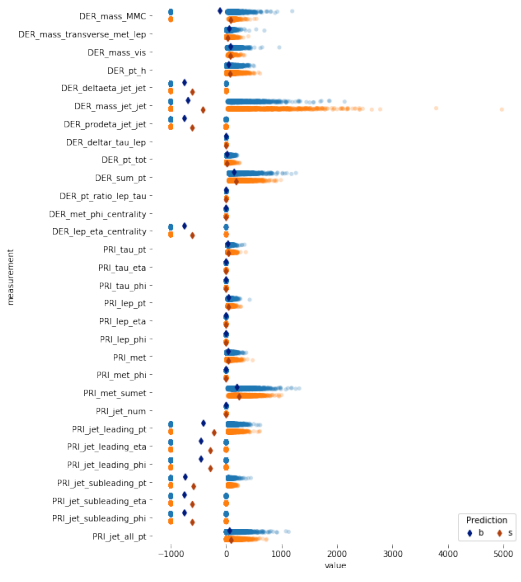

Figure 1. Feature distribution of the original training dataset

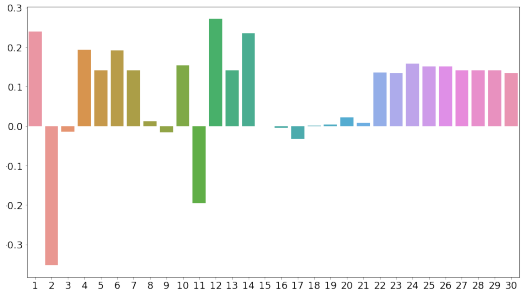

Figure 2. Correlation coefficient of the outlier-free features with the label

II. MODEL SELECTION

As the old saying goes: all models are wrong, but some are useful. The first thing we did is to create a way to evaluate the usefulness of our model. In this project, the accuracy of prediction is the only evaluation criteria. Benefiting from the thinking of cross-validation, our training dataset is split into 15 folds, 14 random folds of which are serving as the training dataset and the rest is serving as the testing dataset in every evaluation loop.

Given an adequate amount of data as the dataset in this project, we could get some reasonable results by applying

simple machine learning method on it. To get an intuition about this project, we choose to use gradient descent as our first model, which is the most common and also the most useful technique. Not surprisingly, this simple model did not give us a surprising result with an accuracy of 0.6958. Other models such as stochastic gradient descent and logistic regression do not give a good feedback. Models of least squares and ridge regression, which could output theoretical best weights guaranteed by the normal equation, return us with a 4.85% improvement in predictions.

Normally more features mean more information, enabling us to approach more closely to the truth. On the other hand, more features may be cursed by over-fitting, but it could be avoided to some extent by using ridge regression. To improve the accuracy of our predictions to a higher level, we dedicated to enriching the features of our dataset. Adding degrees to the features seems to be a reasonable way. To have a test of the impact of adding powers of features, we chose to add a degree of 2 to all features. As expected, adding power to the features truly adds some power to our model, which encouraged us with a better score of 0.7741 by using the model of ridge regression. However, it is easy to figure out that adding the same degree to all features is not reasonable. There should be an optimal degree for every feature. Out of this consideration, we have written functions to select the best degree for every feature based on the greedy algorithm. With this enriched dataset, we got score 0.7991 when applying ridge regression, which has improved 10.33% accuracy from the very beginning.

However, this result did not seems to be the best. Creating features seems to be the right direction to improve the performance of the model. To get a more surprising score, we focus back on our attention to the dataset, aiming to dig out more information. After the explored data analysis, we create some features and improved the accuracy to 0.8138 on ridge regression model, which also has a final score of 0.82346 in Kaggle.

The corresponding accuracy and parameters in the process of selecting models are shown in Table I and Table II, respectively. Note that the classification threshold is taken as 0.5 for (regularized-) logistic regression, and as 0 for others.

Accuracy / Method	Stage 1	Stage 2	Stage 3
Gradient Descent	69.58%	64.70%	-
SGD	66.06%	64.39%	-
Logistic Regression	69.27%	66.30%	72.05%
Regularized LR	69.27%	66.30%	70.73%
Least Squares	74.43%	79.91%	80.52%
Ridge Regressions	74.43%	79.91%	82.35%

Table I
PERFORMANCE OF MODEL

Parameter / Method	Stage 1		Stage 2		Stage 3	
	λ	γ	λ	γ	λ	γ
Gradient Descent	-	10^{-7}	-	10^{-6}	-	10^{-2}
SGD	-	10^{-8}	-	10^{-6}	-	10^{-4}
Logistic Regression	-	10^{-9}	-	10^{-8}	-	10^{-6}
Regularized LR	10^{-1}	10^{-9}	10^{-4}	10^{-8}	10^{-5}	10^{-6}
Least Squares	-	-	-	-	-	-
Ridge Regressions	10^{-4}	-	10^{-6}	-	10^{-7}	-

Table II
PARAMETERS IN THE PROCESS OF SELECTING MODELS

III. PERFORMANCE ANALYSIS AND CONCLUSION

In this project, we realized and compared six different regression models, applied feature engineering and finally got the best results from ridge regression model. As for why other models performed less well, we think it is mainly due to our feature engineering methods, which might be more effective for ridge regression.

We have created features to provide more information. This approach improved our models but also caused over-fitting. Although the dataset was a binary classification dataset, the logistic regression behaved not very well in our model, which should show strong robustness and usefulness in binary classification problems. One possible explanation is that we might select the threshold improperly. For example, a threshold of 0.51 might perform better than 0.5 regarding the logistic regression. Finding better threshold remains also as one of the further work that we need to finish in the future. In order to get small stepwise to enable gradient descent, the values of parameters were quite small in our models. The learning time was quite long due to the small stepwise. Whats more, the stepwise was so small that the loss will be very large when we only changed the values of parameters slightly. Because the margin was so small, we did not use grid search to find the optimal parameters.

Feature engineering has played a critical role in our project. We analyzed the dataset and processed the features. After adding in new features, the performance of these models was improved significantly. To summarize, one key to improve the models is paying more attention to the dataset itself. How to improve feature engineering will be an important work for us in the future.

REFERENCES

[1] Phunterlau, "Winning solution of kaggle higgs competition: what a single model can do?" 2017, [Online; accessed 28-Oct-2016]. [Online]. Available: https://no2147483647.wordpress.com/2014/09/17/winning-solution-of-kaggle-higgs-competition-what-a-single-model-can-do/