PCML CS-433: Higgs Challenge Project

Joachim Muth, SCIPER 214757, joachim.muth@epfl.ch
Junze Bao, SCIPER 266983, junze.bao@epfl.ch
Chanhee Hwang, SCIPER 260872, chanhee.hwang@epfl.ch

School of Computer and Communication Sciences, EPF Lausanne, Switzerland

***Abstract*—This report presents our work for the first project of PCML course. The project mainly contains two parts: one is to implement six basic model functions and the other is to make predictions for physics particles in a Kaggle competition. With the help of dataset documentation and illustration tables and charts, we found that the train dataset can be separated into eight groups with different number of features. We then applied previously implemented ridge regression and logistic regression model to each group to finally achieve an accuracy of 77% with ridge regression (TBD).**

I. Data description

The train dataset consists of $N_{tr} = 250000$ events. Each event is associated with an event ID ($int \in [100000, 349999]$), prediction value (our target y_n variable) and 30 features (x_n feature vector). All 30 features are real valued, except for one feature $PRI\_jet\_num \in \{0, 1, 2, 3\}$ which is categorical. However, some features depend on each other, especially many features are not defined (shown as -999 in the dataset) when the value of the feature jet number is less than or equal to 1. Others, e.g. the last feature $PRI\_jet\_all\_pt$, can be calculated by other features.

The test dataset consists of $N_{te} = 568238$ events ($ID \in [350000, 918237]$). Each event has the same features as the train dataset, with the prediction value being unknown. Therefore our goal is to determine whether an event is a Higgs boson particle event.

II. Data preprocessing

We preprocessed our data and did some feature exploration before selecting a model.

A. Split data

In order to determine the importance of each feature, and to deal with the great number of *NaN* in the dataset, we first analyzed their distrubutions.

[1] shows that it is possible to categorize the events into four different groups based on the value of jet number (*int* $\in [0, 3]$), for the fact that some measures are not defined when jet number is a certain value. During the split, we removed the feature jet number for each group because it is always the same value within each group. As mentioned in the data description, the last feature is a sum of all other features. We also removed this feature due to this reason.

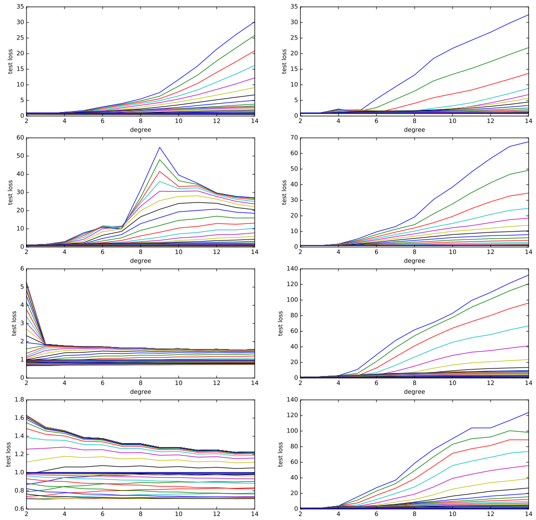

Figure 1. Grid search for parameter selection d and λ on the eight data subsets.

We then obtained four groups with sizes of 99913, 77544, 50379 and 22164 respectively. By skimming through the dataset, we found there are still many undefined values in the first feature DER_mass_MMC, so we further split each group into two subgroups, one with meaningful mass value m_H and the mass value of the other is *NaN*. Finally we have eight different subgroups.

B. Standardization/Normalization

In order to work on comparable data, and not to overweight some of them, we apply standardization to raw dataset before any further processes. Normalization, rescaling the raw data down to range $[0, 1]$, is also applied to reduce te complexity of computation so as to avoid potential expoenential overflow when using logsitic regression model.

C. *Feature selection*

As our eight groups are well split, we discarded the features only containing *NaN* values. All of the remaining features have meaningful real values, but this does not mean that they are all useful. The histograms of their frequencies show candidates of possibly useless features which have almost no variance.

D. *Polynomial building*

Polynomial basis functions can be used to construct more complex and more powerful features, so we constructed a polynomial matrix of degree d with our feature matrix. The degree was chosen by a 4-fold cross validation and is different for each of our 8 groups.

III. Model selection

A. *Models descriptions*

1) Least Squares: LS resolves the matrix equation between $x^T x \cdot w = x^T y$ to find w. Mathematicaly perfectly correct, its big drawback comes from it high sensitivity to outliers and multicollinear dataset, which is often the case when analysing real-world data.

2) Least Squares (Stochastic) Gradient Descent: LSGD starts from a initial w matrix and iteratively improves it depending of the gradient $g = \frac{x^T \cdot y - xw}{\#variables}$. It presents the same weaknesses as LS but provides faster calculation, especially its stochastic variant which select an arbitrary number of small random subsets and calculate the mean between all the their gradient.

3) Ridge Regression: RR add a regularization term λ to the matrix $x^T x$ in order to counter the multicollinearity problem. This λ is empirically defined.

4) (Regularized) Logistique Regression: LR is based on $\sigma(x) = \frac{e^x}{1+e^x}$ function which better scales classification problem where y vector is composed of binary solutions $\{0, 1\}$. It works in a iteratively way by improving w through gradient calculation.

B. *Models comparison*

We only considered ridge regression for linear regression models because optimal w can be derived directly without gradient descent approximation. Ridge regression can also penalize overfitting caused by high polynomial degrees. We also tried logistic regression, which seems to be more suitable for such classification.

C. *Ridge Regression*

We applied a 2-steps grid-search method on each of the eight groups for two parameters: **polynomial degree** $d \in [0, 15]$ and **regularization parameter** $\lambda \in [1^{-10}, 1]$. Each pair of parameters is tested through a 4-fold cross-validation and scored by RMSE error. Figure 1 shows the result. Once the best parameters are chosen for each subsets, we zoomed on them and applied again a grid-search on the zoomed interval, to gain precision. Table II sums up the chosen parameters.

Method	te_loss	accuray %
Least Squares	0.81	75.16
Least Squares GD	0.84	72.70
Least Squares SGD	0.83	73.54
Ridge Regression	0.82	74.88
Logistic Regression	666.42	72.14
Regularized Logistic Regr.	666.42	72.14

Table I
BENCHMARK OF SIX DIFFERENT MODELS

Subset	lambda	degree	Subset	lambda	degree
jet0 no mass	1.599e-05	2	jet0 mass	3.237e-05	2
jet1 no mass	0.1526	13	jet1 mass	3.237e-05	2
jet2 no mass	0.0184	2	jet2 mass	7.906e-06	3
jet3 no mass	0.6250	3	jet3 mass	0.0091	3

Table II
SELECTED PARAMETERS OF RIDGE REGRESSION

D. *Logistic Regression*

We chose regularized logistic regression for fear of overfitting due to using polynomials. We preset a range of lambdas and degrees, followed by using cross validation to test which pair of parameters produces the best accuracy [1] for all groups.

We can still get a real value probablity with logistic regession, although that value will be compared with partition value [2] when predicting specific class labels. Thus we tried 20 partition values in range $[0.3, 0.8]$, and conclude 0.43 is the best choice for this case because local cross validation accuracy reached the top at that value.

Although we achieved 83% in local cross validation, eventually we only achieved 76% in Kaggle with the following parameters: $n_iters = 50000, gamma = 0.000001, lambda_{=}0.01, initial_w = np.zeros(90), cut = 0.43, poly = 2$.

E. *Other experiments*

1) We also tried not to split the dataset into 8 groups, but 4 groups based on jet number or even no groups at all, but expectedly the accuracy did not improve we can add some statistics here.
2) When the train dataset is treated as a whole, we thought it would be better to deal with *NaN* values, e.g. replacing -999 with the mean or median value of all other events. However unexpectedly the accuracy went down dramatically.
3) any other experiments you did ??????

[1] accuracy is calculated as the division of the number of correctly predicted values over the total number of events

[2] typically the 0.5

IV. Summary

Both models gave us an accuracy of more than 80% when tested by 4-fold cross validation locally. However the accuracy against test dataset only scored 77.748% on Kaggle with ridge regression.

The fact that the prediction is better on the local test than in Kaggle.com indicates there might be some overfitting on our training method. However, train error and test error are close as shown in cross validation charts. Right? more summary

References

[1] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kgl, and D. Rousseau, “Learning to discover: the higgs boson machine learning challenge,” 2014.

[2] M. E. Khan, “Logistic regression,” 2015, machine Learning Course - CS-433.

[3] A. Donda, “Avoiding numerical overflow when calculating the value and gradient of the logistic loss function,” 2013, stackoverflow.com.