EPFL Machine Learning - Project 1

Pierre-Alexandre Lee, Thomas Garcia, Yves Lamonato
EPFL Lausanne, Switzerland

1 Introduction

This is the report for the Machine learning EPFL course first project. Using a CERN dataset on the Higgs Boson, we need to find the best model to distinguish background noise from an actual event using machine learning.

2 The dataset

2.1 Raw data

We were given the data in two csv files: a training set and a testing set. In both sets we are given a list of measures *xn* with 30 features each. For the training set, the true binary classification is already given to us, {-1} for background, {1} for true event.

2.2 Data pre-processing

1. *Exploring data*: From the dataset documentation on Kaggle, we know that any missing data was assigned the value {-999}. After some exploratory data analysis, we found that those missing data were determined by the *jet* feature. Therefore for each *jet* value we found which feature could be dropped, and made a different model for each *jet* (that is jets {0}, {1}, and together {2,3} since they missed the same features).

2. *Missing values*: After this jet separation, any value left with {-999} is replaced with {0}.

3. *Data Normalization*: For each *jet* value, we standardize the data by subtracting the mean and dividing by the variance. This helps reduce the range of the variable and thus help to prevent overflow.

4. *Features augmentation*: Finally, to increase the family of functions we can find, we added for each columns the *sin* and *cos* of the columns, and for each positive columns (that is all elements ¿ 0), we added the square root and natural logarithm of the column.

3 Models

3.1 Models used

Those models were tested (the results are displayed in a table below):

- Least squares:
  This was the first model that we tried to get some sort of baseline.

- Ridge regression:
  This is the best model for our processed data, with different values of the hyper-parameter lambda. We also applied another transformation for this model : we raised all the columns from the power 1 to a certain degree d, which we also treated like an hyper-parameter.
  As you will see with the plots, le smaller the lambda, the higher the accuracy. But this caused over fitting which reduced the score in Kaggle so we set the lambda to be at least 1e-08.

- Regularized logistic regression:
  For this model we had to transform the given classification from -1, 1 to 0, 1. The prediction method was then also changed to first apply the sigmoid to the prediction (obtained from np.dot(xn, w)) and then change the prediction threshold to 0.5.

For all models, the best values for the hyper-parameters were determined using grid search.

3.2 Cross-validation

In order to correctly estimate the training and test error, each model was tested using a 10-fold cross-validation, a method in which the data is split in 10 subset. Each subset is then used as a test subset while the rest is the train subset. Both the train and test error are then averaged from all 10 folds, which gives more accuracy to those errors, and help to not over fit the training set.

3.3 Model accuracy

In order to evaluate our models, we used its accuracy, which we computed this way : Using the data and the obtained weights, we make the prediction 'yn', and then compare it to the known classification. We then took the percentage of correct values we predicted

4 Results

The table below summarizes the results. Note that we only display the plots and table for the *jet* value 0, since the methods and results for the other jet subsets were similar.

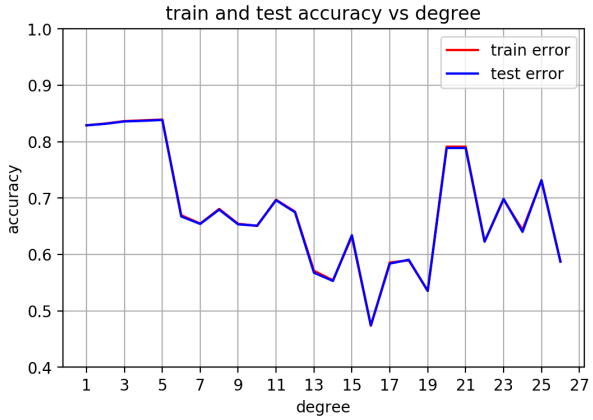

Figure 1: (Left) Training and testing accuracy for Ridge regression with jet=0, lambda=1e-8, for different degrees of polynomial expansion.

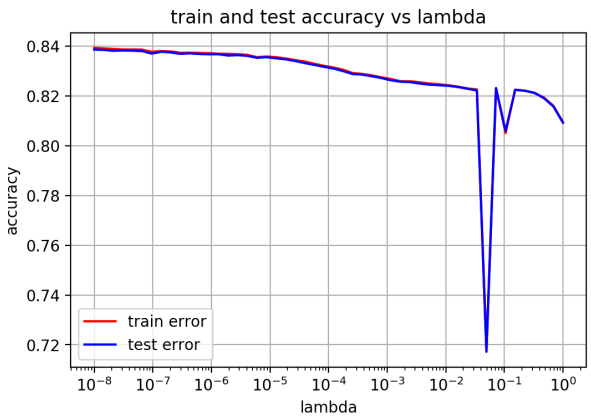

Figure 2: (Right) Training and testing accuracy for Ridge regression with jet=0, degree=5, for different values of lambda.

Model	**Testing accuracy**	**Variance**
Least squares	0.844675385334	2.76e-07
Ridge regression	0.844635350637	5.58e-07
Regularized logistic regression	0.826116982945	1.3884e-05

Table 1: Accuracy and standard deviation for our models (for jet=0)

5 Conclusion

As we can see in the table, least squares has the highest accuracy, but we selected the model with the best score in Kaggle, ridge regression. This certainly comes from the fact that Least square has no lambda hyper-parameter to penalize over fitting. Because of this, the local score wasn't always equivalent to the kaggle score.

It is worth saying that we tried a method which would determine which two columns we should add the product the the features in order to increase the score, but even after determining which columns product to add for each jet, the total accuracy was smaller.