Project-I by Group Rome

Roman Shirochenko
Corentin Tallec
EPFL
roman.shirochenko@epfl.ch
corentin.tallec@epfl.ch

Abstract

In this report we summarize the results we obtained for the first machine learning of the EPFL machine learning course. Facing two datasets, we fit linear models and perform linear regression on the first one, and perform data classification using logistic regression on the second one. For the first dataset, we find that simple linear regression and ridge regression gives decent results, but that we can get better results by isolating one simple feature and performing transformations on that feature.

1 Linear Regression

1.1 Data Description

Our first data set consisted of a train set, with an input and an output variable respectively $\mathbf{X}$ and $\mathbf{y}$, composed of $N = 2800$ data samples, and a test set with only $\mathbf{X}$ variable observed and $N = 1200$ data samples. $\mathbf{X}$ spans over $D = 77$ features, among which 9 of them are categorical, with number of categories ranging from 2 to 4, 2 of them are binary and the other are real-valued.

Our goal is to produce linear predictions for test examples and approximate the test error.

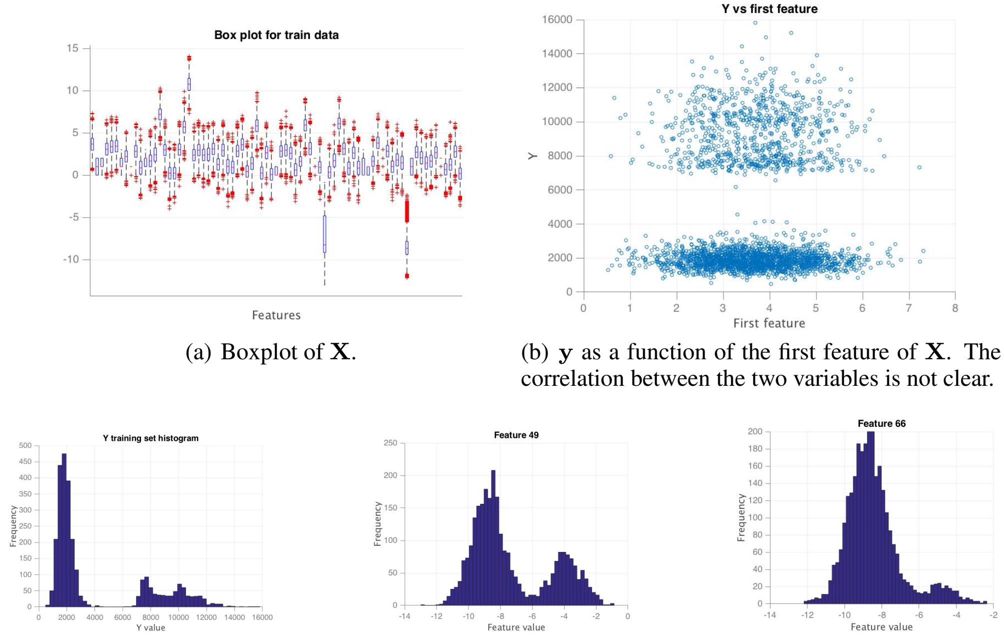
1.2 Data visualization and cleaning

The first tasks we performed on the data was to visualize it. The first thing we did was to obtain a visualization of the data, showed in 1(a) and 1(c). Those two figures led us to normalize both the input and the output variable (normalizing the output variable allow us to have results that are easier to interpret, as they are to be compared to values close to 1). No obvious outliers are noticeable on the $\mathbf{y}$ histogram.

We performed featurewise analysis to notice that most of our real valued variables have nearly gaussian distributions (or at least unimodal distributions), which is not the case of our output variable. Figure 1(b) shows how the output variable evolves as a function of the first feature. This behaviour is what we notice on most variables, and thus shows no clear correlations.

Although for variables 49 and 66 we notice a multimodal distribution that is closer to the behaviour of our output variable, as seen in Figure 1(d) and Figure 1(e). We thus chose to try and predict $\mathbf{y}$ using only those two features, transformed through polynomial transformation.

The categorical variables were encoded using dummy encoding, what led to a total feature number of 100. As the rank of our dataset is below 100 (it is 70), simple linear regression is likely to fail, and we use ridge regression to cope for the rank deficiency.



(a) Boxplot of $\mathbf{X}$.

(b) $\mathbf{y}$ as a function of the first feature of $\mathbf{X}$. The correlation between the two variables is not clear.

(c) Histogram of $\mathbf{y}$. The output distribution is bimodal.

(d) Histogram of feature 49 of $\mathbf{X}$. We see that the distribution is bimodal, as the output.

(e) Histogram of feature 66 of $\mathbf{X}$. Again, the distribution is bimodal.

Figure 1:

1.3 Ridge regression

As the dimension of the data is inferior to the number of feature, we expect least-squares to work quite poorly, and thus decided to apply mostly ridge regression.

For our first attempt at predicting the data, we simply used the initial features and used kFold with $k = 4$ to display the evolution of the RMSE as a function of the regularizing lambda both for the training and test data. The lambda we used ranged from 10^{-4} to 10^{3}, with 500 points. The results of the ridge regression are already satisfactory, with an error of approximately 0.32 ± 0.005 on the training set and 0.34 ± 0.005 on the test set, that is approximately one third of the standard deviation of the output variable.

We see that for sufficiently low lambda, both the training and test errors do not vary much (there is indeed only a very slight variation of test RMSE, with a nearly unnoticeable minimum between $\lambda = 10$ and $\lambda = 100$).

The results described here are summerized in the Figure 2(a).

1.4 Feature transformations

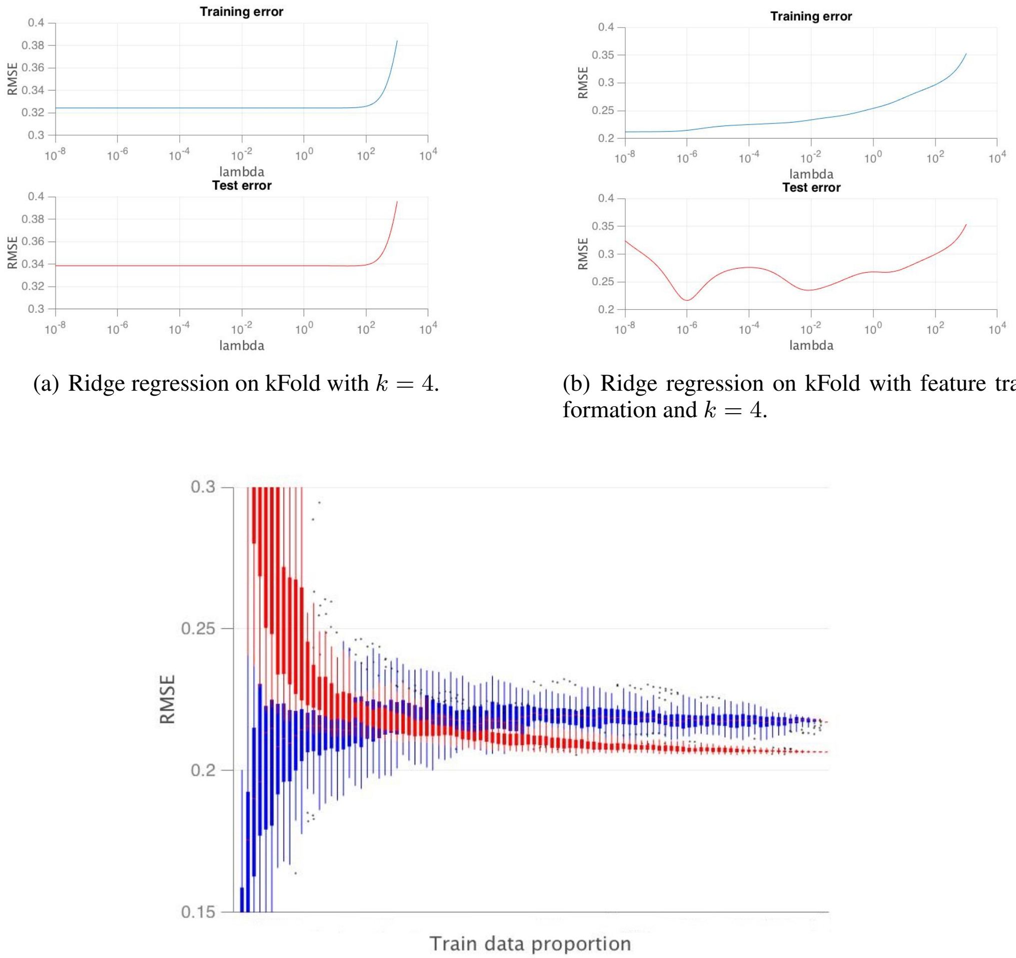
As notified in a previous section, among all the feature at our disposal, features 49 and 66 show distributions that ressemble the ditribution of the output variable.

This led us to try and regress only on thos two features and polynomial transformations of them. Using only those two features, we get an RMSE error that is quite close to the one we notified when all the features are used. When using polynomial features (up to degree 10), we basically get half the error that we got using all the features. Those results were obtained using ridge regression. As in the previous section, we obtained the RMSE as a function of lambda, calculating both training and test errors using a kFold procedure with $k = 4$. The results are summarized in Figure 2(b), where, as previously mentionned, we used as features only the two features and their polynomial transformation with degree ranging from 1 to 10.

This time, the test error varies much more as a function of lambda, and, given our curve, the best lambda possible seems to be around 10^{-6}.

Finally to evaluate our performances more accurately, we plotted learning curves for our model, with 20 features obtained by polynomial transformations and a lambda of 10^{-6}. The learning curve is obtained by using 20% of the training data to perform tests, while learning the model on a progressively increasing portion of the 80% remaining. We monitor the test and training error as functions of the proportion of data used to train. The results are summarized in Figure 2(c).

As expected, both training error and test error decrease as the proportion of training data increases. The test error obtained seems to exhibit a quite satisfactory behaviour.



(a) Ridge regression on kFold with $k = 4$.

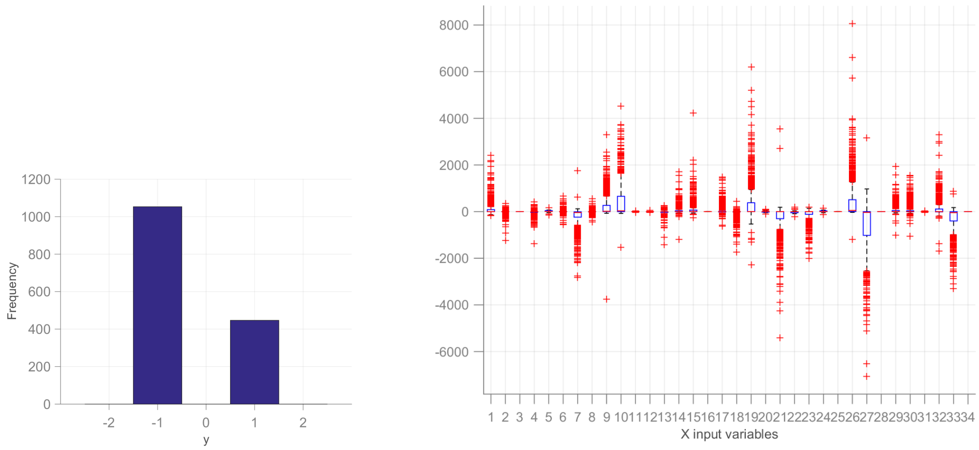
(b) Ridge regression on kFold with feature transformation and $k = 4$.

(c) Learning curve. Blue is training data and red is test data.

Figure 2:

1.5 Regression Summary

While simple ridge regression with an appropriately optimized lambda already gave decent results on our dataset, we obtained better results using polynomial transformations of well chosen features. Doing so, we obtain an error that is approximately one fifth of the standard deviation.



(a) Histogram of $\mathbf{Y}$ train.

(b) Boxplot of $\mathbf{X}$ train.

Figure 3:

2 Logistic Regression

2.1 Data Description

Our second data set consisted of a train set, with an input and an output binary variable respectively $\mathbf{X}$ and $\mathbf{y}$, composed of $N = 1500$ data samples, and a test set with only $\mathbf{X}$ variable observed and $N = 1500$ data samples. $\mathbf{X}$ spans over $D = 34$ features, among which 5 of them are categorical, with number of categories ranging from 1 to 5.

The categorical variables were encoded using dummy encoding, what led to a total feature number of 47. As the rank of our dataset is below 47 (it is 42), we use penalized logistic regression to cope for the rank deficiency.

Our goal is to produce binary classification predictions for test examples and approximate the test error.

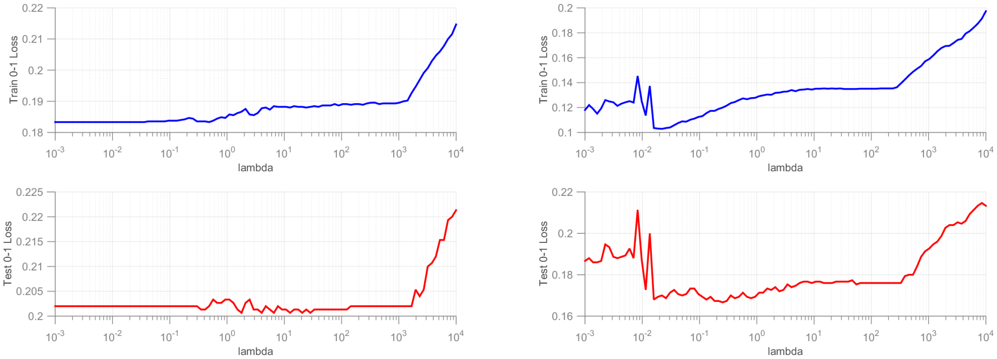
2.2 Data visualization and cleaning

Firstly, the data input $\mathbf{X}$ train and $\mathbf{Y}$ train variables were visualized. The figures are showed in 3(a) and 3(b). After plotting these figures, all -1 values were changed to 0 in $\mathbf{Y}$ train and $\mathbf{X}$ train variables were normalized (normalizing the output variable gave us a faster convergence for both gradient decent and Newton's method). We tried gradient decent and Newton‘s method for Logistic regression and Penalized Logistic Regression and chose Newton‘s method as the execution time was nearly the same, but Newton‘s method displayed better convergence than gradient descent method (as we needed to changing alpha parameter too) and with less iterations.

Plotting histogram graphs for each variable in $\mathbf{X}$ train, showed that features have different distributions and mostly do not follow normal distribution.

2.3 Penalized Logistic regression

We applied logistic regression and penalized logistic regression method on ourdataset. As we have many features and as the dataset is rank deficient, we expected that Penalized Logistic regression would improve test error result for kFold cross-validation with $k = 4$ by penalizing some features. But the test 1-0 loss error for both methods was nearly the same and equals to 0.183 ± 0.001. The summarized result with lambda ranging from 10^{-4} to 10^{3} with 100 points is presented in the Figure 4(a).



(a) Penalized logistic regression on kFold with $k = 4$.

(b) Penalized logistic regression on kFold with $k = 4$ and feature feature transformation.

Figure 4:

2.4 Feature transformation

Several features transformations were performed. Features deleting or using only classified features did not improve prediction result. Then we applied polynomial function transformation for each feature. We tried the same approach as with linear regression, we found features with two modes histogram distribution (the hypothesis is that features follow 0 and 1 output result) and use only them with polynomial feature transformation, but it did not improve the error resuls.

We compared penalized logistic regression without features engineering results with polynomial transformation. Method with transformation showed better result and gave the 1-0 loss test error result 0.167 ± 0.001. The lambda parameter without feature transformation was chosen to be 0.001 and with feature transformation was chosen to be 0.3 according to the results of a k-fold cross validation procedure with $k = 4$ and lambda ranging from 10^{-3} to 10^{4} with 100 points. The result described here are summarized in the Figure 4(a) and 4(b).

3 Summary

We analyzed a regression and classification datasets and that ridge regression and penalized logistic regression are reasonable fit. We tried some feature transformation and found that polynomial feature transformation gives us a little improvement in final test error result. We estimate that the average test error for regression problem is 0.225 ± 0.005 on the test set, that is approximately one fifth of the standard deviation of the output variable. The test error for classification problem is 0.167 ± 0.001.

All the scripts that were used can be find on the webpage: `https:\github.com/corentintallec/mlproject`

4 Acknowledgments

We would like to thank Emtiyaz Khan and all teacher‘s advisers for making the datasets for this project and answering our question during exercise sessions.

The regression part was done by Corentin Tallec, as well as section Linear regression of report. The classification part was done by Roman Shirochenko, as well as the Classification part of the report.