Project-I by Group LasVegas

Igor Kulev
EPFL

Andrii Maksai
EPFL

Marjan Shahpaski
EPFL

{igor.kulev, andrii.maksai, marjan.shahpaski}@epfl.ch

Abstract

This report provides a summary of our work done for the first project of the PCML course. The project consists of solving two problems: a regression and a classification problem. For the regression dataset, we have noticed that the data was generated by two distinct sources. Therefore, we separated the data in two parts, and have applied different linear models to each part. To the classification dataset, we applied ridge regression and penalized ridge regression. In addition, we have investigated a few feature transformations which provided some additional improvements.

1 Introduction

The first project of the Pattern Classification and Machine Learning course focuses on applying linear models to regression and classification problems. Two different datasets were therefore provided, one for each of the problems. The ultimate goal is to train a model which will be able to produce accurate response predictions to unseen input data. The unseen data is assumed to be produced by the same source(s) as the training data. The tasks involved in solving these problems are preprocessing the data, training linear models for the corresponding regression/classification tasks, and applying appropriate feature transformations to the input variables.

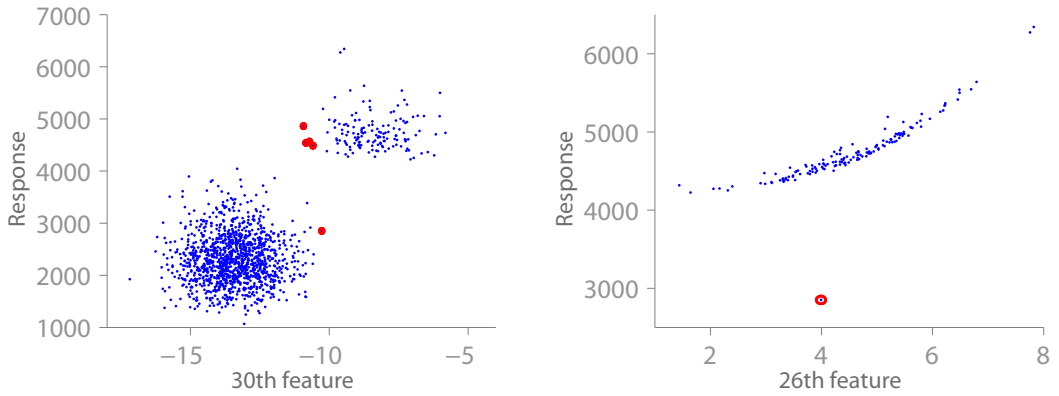
The algorithms which we use for regression are least squares and ridge regression. For classification, we have implemented logistic regression and penalized logistic regression.

2 Regression

Regression models try to establish the relationships between the input and the output variable(s) of a process. It is therefore used for predicting future outcomes for new (unobserved) data, or for interpreting the underlying connection(s) between the input and the output variables. In this project we will use linear regression models, which assume a linear relation between the inputs and the outputs. We will also use feature transforms which create non-linear input basis for the linear regression, however the regression parameters will remain linear.

2.1 Data Description

The training data for regression consists of $N_{tr} = 1400$ input and output data samples. Each input sample is a vector $\mathbf{x}_n$ with dimensionality $D = 38$. All input samples are stored together in the matrix $\mathbf{X}_{tr}$. The output variables are scalar, and they are stored in the vector $\mathbf{y}_{tr}$. Each input vector $\mathbf{x}_n$ contains 30 real valued variables and 8 categorical variables. From the categorical variables, 1



(a) Correlation of the 30th feature with the response. It can be clearly seen that the data was generated by two sources. The data points encircled in red are the outliers produced by our initial classification.

(b) Strong correlation between the 26th feature and the response for the points from the "right" cluster. The single outlier is encircled in red.

Figure 1: Input-output data correlation and outlier visualization.

is binary (variable 37), 3 have three categories (variables 31, 33 and 34), and 4 have four categories (variables 32, 35, 36 and 38).

The test dataset consists of $N_{te} = 600$ samples. Here we observe only the matrix of input variables $\mathbf{X}_{te}$, and we do not have access to the vector of output variables $\mathbf{y}_{te}$. The goal is to train a regression model by using the training data $\mathbf{X}_{tr}$, and then use it to produce predictions $\hat{\mathbf{y}}_{te}$ for the unknown output variables $\mathbf{y}_{te}$, which correspond to the input variables stored in $\mathbf{X}_{te}$. An approximation of the test error for unseen data for our model is also expected in the form of RMSE.

2.2 Data visualization and cleaning

By visualising the correlation of each input feature with the output variable, we have noticed that there are two distinct clusters of values which the output variable takes. Figure 1(a) shows the response as a function of the 30th feature. From the figure it becomes clear that the data was generated by two independent sources, and that the 30th feature provides a very clear separation between them. In addition, the histogram of the 30th feature of the test data samples $\mathbf{X}_{te}$ also shows a clear distinction between two Gaussian lobes. In fact, when plotting their histograms out of the 30 continuous features, 29 appear to have a Gaussian distribution of their values, and one (the 30th features) has a two Gaussian lobes. By visual inspection, we have concluded that the value of -10.5 of the 30th feature provides the cleanest separation between the "left" from the "right" clusters.

In order to have a soft margin, and therefore probabilities of the cluster belonging for each data sample, we could have used logistic regression for the separation of the two clusters. This approach, however, will increase the complexity without providing an added value, since we do not see a clear way of using the computed probabilities in this context.

After we separate the "left" and the "right" clusters, we eliminate the outliers. The best separation boundary at -10.5 for the 30th feature misclassifies five data samples. They are shown encircled in red on Figure 1(a). The encircled samples which have a response between 4500 and 5000 are classified in the "left" dataset and represent the only outliers, since all other data points are inside four standard deviations from the mean of the cluster. On Figure 1(b) we show the response vs. the 26th feature for the data samples which fall into the "right" cluster. We see a very clear pattern followed by all data samples except the one encircled with red, which is a result from the misclassification. We manually remove the outlier data samples from both clusters.

We normalize the input variables which have continuous values to zero mean and unit variance for all training data, or for each data cluster separately, depending on the algorithm used (explained in the following subsection). For the categorical variables which have more that 2 categories we have

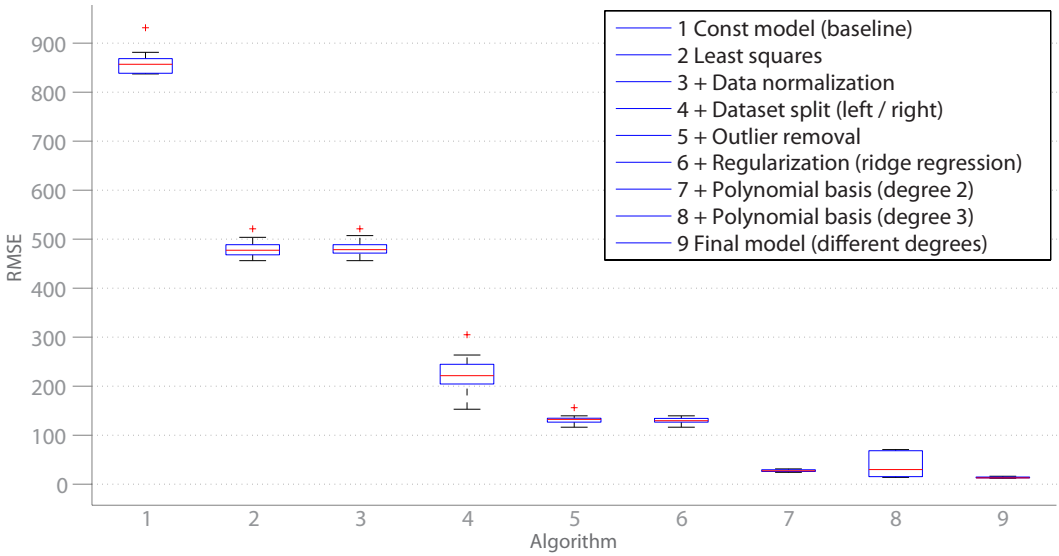


Figure 2: The box plots show an overview of the mean and standard deviation of the different regression methods and feature transformations which we have tried. Our model of choice is #9.

used dummy encoding. Due to this encoding, the final number of input variables has increased from 32 to 50.

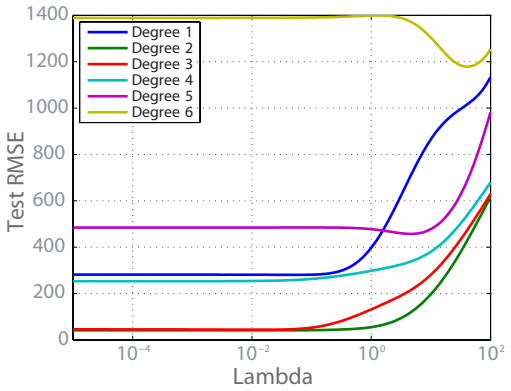
2.3 Methodology

For finding the best solution for the regression task we have tried several linear regression methods, as well as different feature transformations. Figure 2 gives an overview of the test error RMSE performance of the different methods. For producing the test RMSE, we have divided our input training dataset $\mathbf{X}_{tr}$ in two parts, training and testing with 80%-20% ratio, respectively. The model parameters were computed by using only the training set, and the test RMSE was computed on the test set of data, which was completely isolated from the model training. Where required, the additional parameters of the models (e.g. polynomial basis degree, lambda) were computed by applying a 10 fold cross validation only on the training set. The computation of the test RMSE was repeated 10 times, and with each trial we selected the train and test sets by random permutations, and by keeping the data ratio constant. We will justify the choice of 80%-20% train-test split ratio in the following text, by plotting the learning curve (Figure 3(b)).

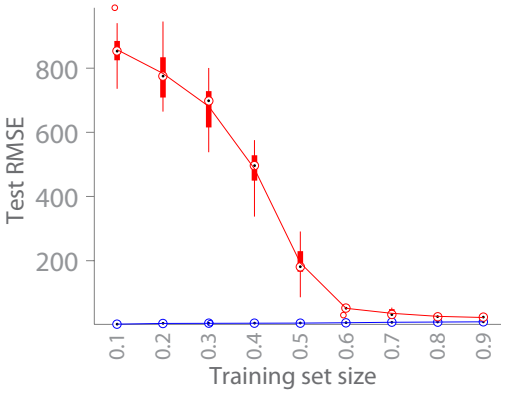
The first algorithm in Figure 2 is the baseline; it is just the mean of the output variable, and it does not depend on the input variables. For the second model, we apply least squares, and gradually increase its complexity by adding different features. From the sixth model, we use ridge regression (regularized least squares), and, together with some data manipulation and feature transformations, we select it as the best algorithm for making predictions for our dataset.

Least squares easily outperforms the baseline algorithm. This is expected, since we previously observed that the input features are correlated with the output. We use all 50 input features in the model. As it can be seen from algorithm 3, normalizing the input variables does not provide an improvement for this dataset. However, when we split our data according to the 30th feature, as explained in the previous subsection, we experience a great improvement. In this case, we compute different model coefficients for the ”left” and the ”right” cluster. As a next step, we remove the outliers in each (”left” / ”right”) split separately before training the model. This leads to an unexpectedly large increase in the performance of the model, although there are 4 outliers out of 1255 data points in the ”left” and 1 outlier out of 145 data points in the ”right” split.

When applying least squares we have noticed that our Gram matrix $\tilde{\mathbf{X}}^{\mathbf{T}}\tilde{\mathbf{X}}$ is ill-conditioned. To overcome this problem we have used ridge regression for lifting the eigenvalues of the matrix.



(a) Test RMSE for 100 different lambda values in the range from 10^{-5} to 10^{2}, computed for the "right" data split. The various curves show the RMSE when the input features have been raised to the power of the corresponding degree.



(b) Learning curve for the regression task. After the 80%-20% split for train-test data, the test error stabilizes and nears the train error.

Figure 3: Comparison of different lambda and degree combinations. Learning curve for regression.

We selected the optimal lambda for ridge regression from the range $[10^{-5}, 10^{2}]$ by testing 100 values in between, and choosing the lambda which minimizes validation RMSE, by employing 10 fold cross validation. We did not penalize the intercept β_0, which had values greater than 1000 for our dataset. The optimal lambda for both data splits are λ_L=5.4e-01 and λ_R=4.7e-02. Resorting to ridge regression only slightly improved the test RMSE and its variance, however, it solved the ill-conditioning problem.

2.4 Feature transformations

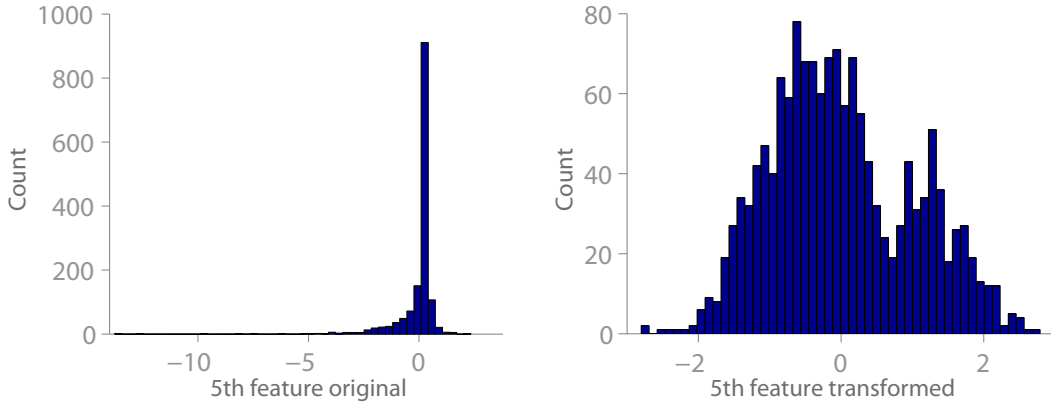
We also investigated the effect of different non-linear basis functions on our prediction accuracy. Although we will work with non-linear basis functions which will transform our features, the regression parameters will remain linear.

In algorithms 7-9 from Figure 2, we have used basis functions with different degrees. For algorithm 7 we squared all data entries which have continuous values for both "left" and "right" data splits, and in algorithm 8 we have cubed them. It can be seen that these feature transformations significantly improve the prediction accuracy compared to the algorithms which did not employ them.

Figure 3(a) shows a plot of the test RMSE for the "right" split, for different λ_R. The various curves show the RMSE when the input features have been transformed by raising them to the corresponding degree. From the figure it can be seen that degree 2 and 3 clearly outperform all other degrees. A similar conclusion can be drawn by observing algorithms 7 and 8 from Figure 2. In addition, from Figure 3(a) it can be seen that all degrees have their optimal lambda value close to 0.

In algorithm 9 we have allowed the features of the "left" and the "right" splits to be transformed with independent basis function. The optimal degree for the "left" split is 2, and for the "right" split is 3. The assembly of steps contained in algorithm 9 constitutes our final model, which we will use for providing predictions for the values in $\mathbf{X}_{te}$.

To finalize the discussion for regression, we show a plot of the learning curve in Figure 3(b). The plot is produced by using our final algorithm 9, while changing the ratio of train and test data. We start with 10%-90% train-test split, and finish with 90%-10% train-test split. At the beginning the train RMSE is close to 0, since the training data is very limited, and the model is overfitting, hence the huge test error. As the amount of training data increases, we observe that the test error rapidly decreases, nears the train error and stabilizes after 80%-20%. This tells us that the amount of training data is sufficient for the calibration of our algorithm 9.



(a) Histogram of the 5th feature for classification in its original space.

(b) Histogram of the 5th feature for classification in the transformed space.

Figure 4: Feature transformations for classification.

3 Classification

Classification categorizes the input variables(s) in discrete output classes or categories. It is used for predicting the categories for new (unobserved) data samples. For this task, we will train linear binary classifiers which we will later use to classify unseen (test) input variables.

3.1 Data Description

The data format for classification is similar to the data which we have already seen for the regression task. We will use the same notation for this task too. The training data for classification consists of $N_{tr} = 1500$ input and output data samples. Each input sample is a vector $\mathbf{x}_n$ of dimensionality $D = 32$, and all input samples are stored in the matrix $\mathbf{X}_{tr}$. The output variables are binary, since there are two classes, and they are stored in the vector $\mathbf{y}_{tr}$. Each input vector $\mathbf{x}_n$ contains 30 real valued variables and 2 categorical variables. From the categorical variables, 1 has three categories (variable 22), and the other one has four categories (variable 1).

The test dataset consists of $N_{te} = 1500$ samples. Here we again observe only the matrix of input variables $\mathbf{X}_{te}$, and we do not have access to the vector of output variables $\mathbf{y}_{te}$. The objective of the classification task is to train a classification model by using the training data, and then to produce class predictions for the unknown output variables $\mathbf{y}_{te}$, corresponding to the input variables in $\mathbf{X}_{te}$. An approximation of the total misclassification error for our model is also expected in the form of RMSE, 0-1 loss and log loss.

3.2 Data visualization, feature transformations and cleaning

By plotting the histogram of each continuous input variable, we have noticed that most of them have a Poisson-like distribution of their values. In order to bring their distributions closer to Gaussian, we have applied a feature transformation to each continuous element of the input training matrix $\mathbf{X}_{tr}$, by raising them to the power of $\frac{1}{4}$. Since some of the features had negative entries, we handled them by computing $-\sqrt[4]{-x}$. Figure 4(a) shows the original distribution of the 5th feature. On Figure 4(b) we can see the Gaussian nature of the distribution of the 5th feature after applying the transformation.

Similarly to the regression task, we again normalize the continuous variables and remove the outliers. We have used the transformed feature space for the removal of the outliers. We eliminated the data vectors which had at least one feature outside of the 4σ interval from the mean of the Gaussian of the corresponding feature.

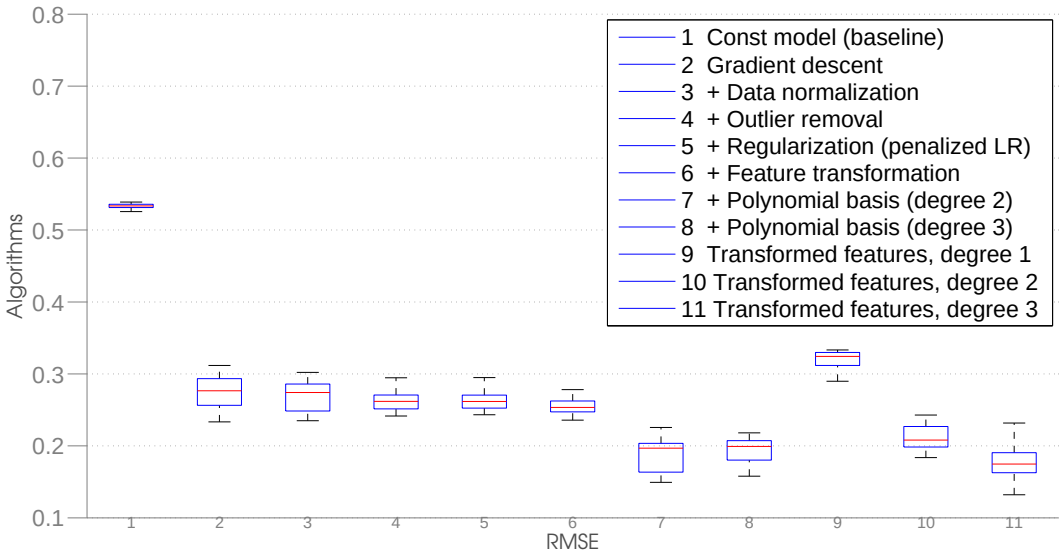


Figure 5: The box plots show an overview of the mean and standard deviation of the different classification methods and feature transformations which we have tried. Our model of choice is #1 THIS FIGURE NEED TO BE UPDATED: RMSE ¿ 1

3.3 Methodology

The evaluation of the classification algorithms was done similarly to the evaluation of the regression algorithms. The train-test split was again 80%-20%, we have used 10 fold cross validation for determining the necessary method specific parameters, and the mean and variance of the test error was established from 10 trials of randomly permuted train and test datasets.

A comparison of the performance of the various methods which we have used for classification can be seen on Figure 5. We have used three different linear classification methods, namely: constant model (baseline), logistic regression with gradient descent, and penalized logistic regression. The baseline, as for the regression task does not depend on the input variables; it is just the mean of the output variables, which is passed through the logistic function. The plain logistic regression from algorithm 2 clearly outperforms the baseline. The normalization of the features plays a very important role in lowering the test RMSE in the case of classification, as it can be seen from algorithm 3, which was not the case for regression.

The next interesting algorithm is the addition of the polynomial basis functions in algorithm 7. There, we add non-linear basis function to the original features, by raising them to the power of 2, and in algorithm 8 we also add the basis functions of degree 3. The error decreases slightly, thus the blessing of high dimensionality overweights of curse of dimensionality (we work in 189 dimensional space). It is also important to note that the optimal lambda for the penalized logistic regression (algorithm 5) was calculated from the range $\lambda = [10^{-2}, 10^{...}$ with 10 samples in between.

In algorithms 9-11, we use only the transformed features, which we have described in the previous subsection, and do not use the original features. The test error for degree 1 is higher then for the original features with degree 1, however, for degrees 2 and 3, the transformed features outperform the original features. The mean test error for algorithm 11 is the lowest across all algorithms, and it uses only a single set of 63 transformed features (and cubed for the non-linear basis). Therefore, it is our model of choice for making the class predictions of the test data stored in $\mathbf{X}_{te}$.

The learning curve for the classification task (not shown due to lack of space) is similar to the learning curve for the regression task, which is shown in Figure 3(b). However, there is also a significant difference between the two. Namely, the test error for the classification task continues to decrease, and the training error continues to increase as we reach the 90%-10% train-test data split. This suggests that a larger dataset for classification will be useful for our algorithm of choice #11.

4 Summary

In this work we have analysed a regression problem. First, we split the dataset in "left" and "right" data splits, since the data was produced by two sources. Then we tested a set of linear regression algorithms together with different feature transformations, and have settled for ridge regression working with different polynomial basis for the two data splits. The achieved test RMSE is ≈ 15, which outperforms the baseline by 50 fold, and simple least squares by 30 fold.

We also analysed a classification dataset. There we noticed strongly skewed Poisson-like distributions of the input features. Therefore, we transformed the features to assume more Gaussian-like distributions. The model of choice was penalized logistic regression which worked solely on the transformed features, and polynomial basis functions of degree 3. The achieved test RMSE is ≈ 0.18. This model significantly outperformed the baseline and plain logistic regression.

Acknowledgments

The code for the essential functions was developed by all team members independently for error checking and correction. The bulk of the testing was done by Igor and Andrii. Igor produced the final "test errors" and "predictions". Andrii produced the figures, and Marjan summarized our finding by writing most of the report. All team members were engaged in the frequent project discussions.