Project-I by group TORONTO

Michalina Pacholska **Jakub Sygnowski**

Abstract

This raport describes our work on first project done for Machine Learning class at EPFL in Fall 2014. We were given two synthetic datasets - a regression and a classification one and used methods learnt in the class to train few models and predicted regression and classification outcome for the test data we got. For classification, we tried fitting logistic regression, its penalized version and SVM. We achieved the best test error for penalized logistic regression with all input variables and their absolute values. For regression, we found out our data matrix is ill conditioned, the reason of which were discrete input variables. We removed them and then fit least squares with both gradient descent and normal equations and ridge regression. Ridge regression did not give us significant improvement over least squares and we used the latter model to predict the outcomes.

1 Classification

1.1 Data preparation

Our dataset consists of train data, for which we have both input variables X and output y and test data, for which we observe only X and have to produce our predictions and approximation of certainity.

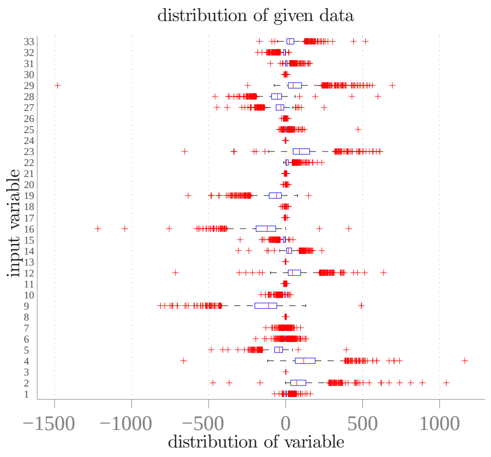
Both original train and test data sets included $N = 1500$ samples, each has 33 dimensions. All except 3-rd input variable are continuous, 3-rd is a binary one. Our X_train matrix has full-rank, so we don't expect one input variable to be a linear function of other ones. Data we got was not normalized, as shown in figure 1(a), so we decided to normalize it. We also normalized test data using same means and standard deviations. After that, we randomly a subset of $N = 70$ data samples to leave it aside and use it at the end to estimate RSME error without being biased because of using cross-validation.

Figure 1(a) shows also that our data is not free of outliers. We decided to remove from dataset all samples that have absolute value of any input variable ≥ 5 standard deviations of this input variable (after normalization).

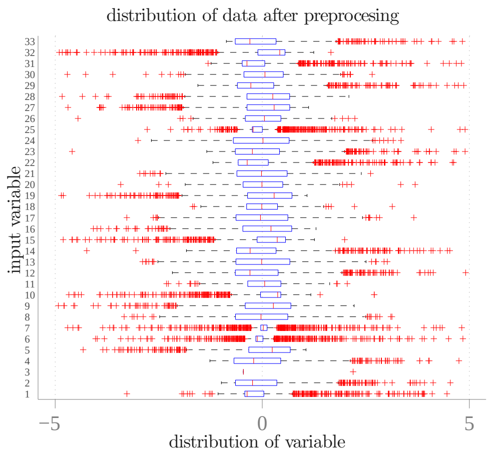
After these procedures, we end up with normalized data shown on figure 1(b). There was $N = 1284$ data samples left available to train our model.

1.2 Data analysis

We investigated correlation between different input variables and an output variable and amongst input variables themselves. Figure 2(a) shows that some variables are clearly more correlated to the output then the others. While building models we sometimes tried to create them only using few most correlated variables. On the other hand, we found no interesting correlation between input variables. We also tried to apply PCA to extract principal components of the data, but we found that particular principal components are not so well correlated with output variable (see fig. 2(b)), so we abandoned this path and did not use PCA later.

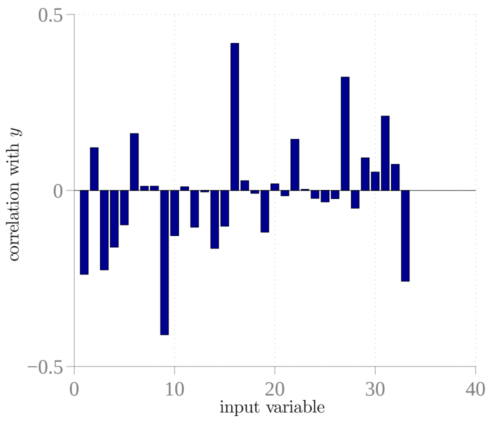


(a) Boxplot of original input data $\mathbf{X}$. Data is not centered and therefore we normalize it.

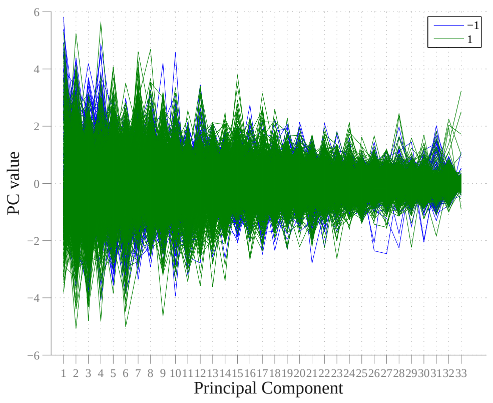


(b) Boxplot of input data after normalization and removing outliers.

Figure 1:



(a) Correlation between input variables and output variable. Some variables have much bigger correlation to the output variable than the others.



(b) Data transformation using PCA. Big correlation of some input variables with y is lost.

Figure 2:

1.3 Predicting models

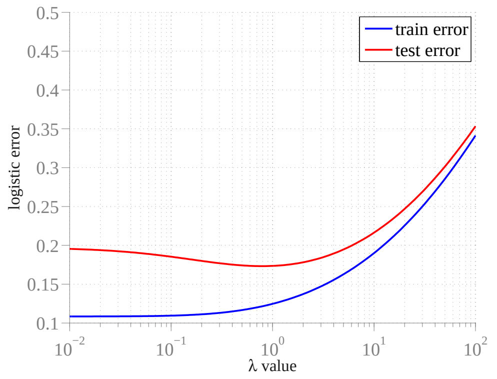
1.3.1 Logistic regression using gradient descent

First model we tried to fit was simple logistic regression (using Newton's method with Hessian). Using 10-fold cross-validation, we achieved mean logistic test error of 0.2301. The results had big variance. We later tried to fit the model using only 9-th input variable (the one with biggest correlation with output), but it got worse test error.

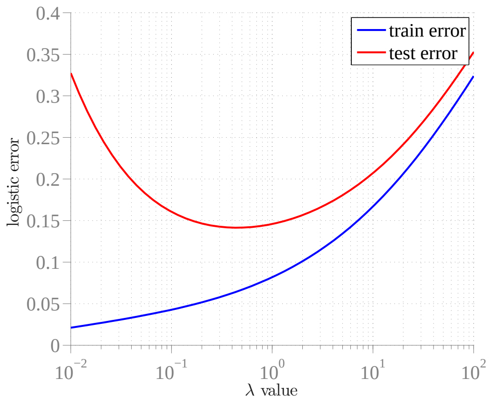
1.3.2 Penalized logistic regression

We did a lot of experiments using penalized logistic regression. As converging using gradient descent was slow, we wrote a Hessian-based version.

To choose optimal value of λ, we used 10-fold cross-validation. We tested 200 candidates in logspace between 10^{-2} to 10^{2}. Figure 3(a) shows the result: optimal lambda was $\lambda = 0.8$ with test error around 0.19.



(a) Train and test error of penalized logistic regression without transformations for different value of λ



(b) Comparison of train and test error for different value of λ in model with absolute value of input variables

Figure 3:

1.3.3 Feature transformations

Later, we tried various feature transformations. We mention the ones which did not get any improvement in no significant order:

- running penalized logistic regression for only some ($\{1, 9, 16, 27, 33\}$) of input variables with biggest correlation to the output variable
- penalized log. regression for subset of variables and their squares and square roots
- penalized log. regression with all input variables and their square roots
- penalized log. regression with all input variables and their cubes

On the other hand, we got similiar improvement (logistic error went down to around 0.15) for both adding squares and absolute value of every input variable. The result of cross-validation for different λ and all variables and their absolute value is shown on figure 3(b). As expected, we observe increase in test error for small value of λ (variance), but overall, for $\lambda = 0.5$.

1.3.4 SVM

We also tried to fit (builtin) SVM model for our problem. Unfortunately, when we chose only a subset of input variables we got bigger error then before, and when we tried to fit SVMs for all input data, training never finished. Changing the kernel didn't help.

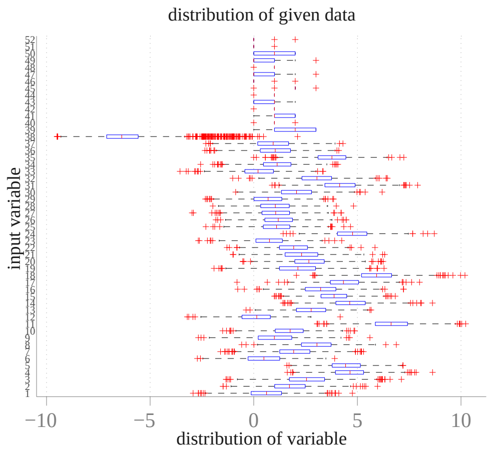
1.4 Summary

After fitting the mentioned models, we found out that most of input variables have an inpact on predicting the outcome y. Our logistic regression test error (0.18) was on average bigger than the penalized regression one (0.15), so we decided to use penalized logistic regression to predict the unknown outcomes. We did not managed to make SVM work for all the features due to its computational complexity.

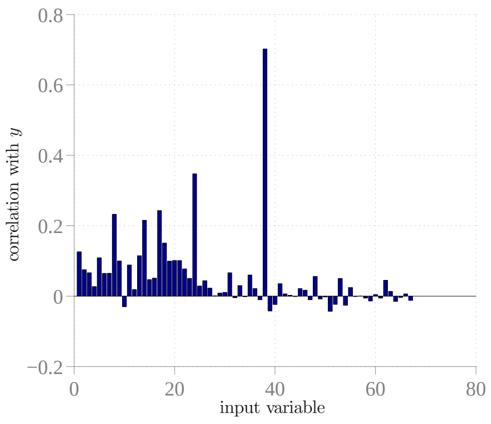
2 Regresssion

2.1 Data preparation

For the regression problem, we got input data matrix X with $N = 1400$ data samples and $D = 52$ input variables. First 38 of input variables are continuous, last 14 are discrete, with more than 2 categories. X matrix is again full-rank. Figure 4(a) shows that data is again not normalized, especially 38-th column.



(a) Boxplot of original input data $\mathbf{X}$. Data is not normalized and categorical variables are not separated



(b) Correlation of input variables with the output variable

Figure 4:

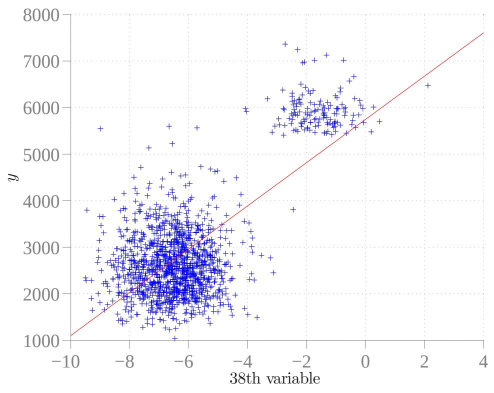
2.1.1 Dummy encoding

First procedure which we applied to our data is dummy encoding of categorical variables. After changing variables to binary, we had matrix X with 80 columns but only 67 rank, which means that there was some redundancy in categorical variables. To avoid problems with ill-conditioning, we chose 67 linearly independent columns of X and discarded others.

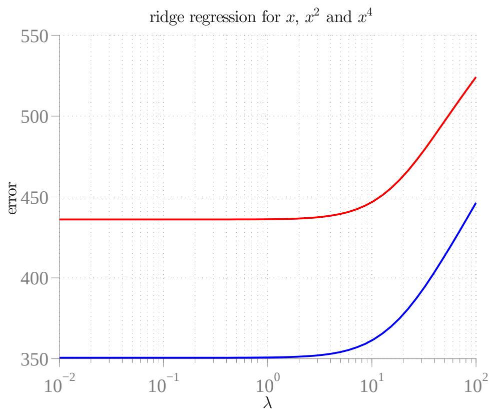
Further, we normalized the data and removed sample case #500 as an outlier.

2.2 Data analysis

We took a look at the correlation between input variables, and there was not much correlation. More interesting was correlation of each input variable with the output one, which is shown on the figure 4(b). We clearly see that there's a big correlation between 38-th input variable and our output, but if we took a look at the plot of this variable vs y (fig. 5(a)) we see that data is in fact clustered and big correlation may not guide us to very good predictions.



(a) Plot of 38-th input variable vs output variable. We see the clusterization of the data separated. Fit is good for one cluster, but not for the second one.



(b) Train and test error for different values of λ in ridge regression. We see that penalizing big β values is not justified.

Figure 5:

2.3 Least squares and feature transformations

We started by analysing the behaviour of least squares under 10-fold cross-validation. We run each testing case for 10 choices of seeds and averaged the results. Testing cases were:

- all input variables, like we got them: $rsme = 641$
- all input variables and squares of continuous ones ($\{1, \dots, 38\}$), $rsme = 59$
- only continuous variables and their squares: $rsme = 471$
- only continuous variables, their squares and fourth powers: $rsme = 469$

This was not very suprising, as:

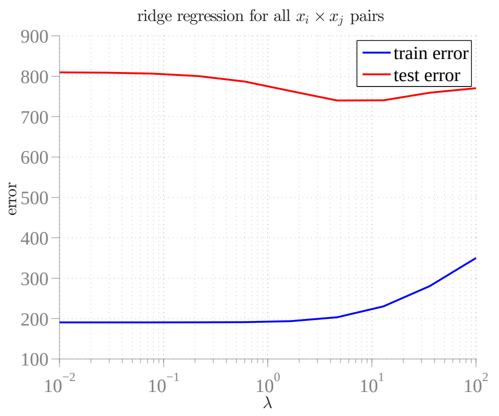
1. we had badly conditioned matrix X, only because of our discrete variables, which could lead to computational errors when solving normal equations with all variables,
2. discrete variables were very lightly correlated with output variable, so removing them did not made us lost much of information.

2.4 Gradient descent

Running gradient descent for this dataset was problematic. We for step size $\alpha > 0.2$ it did not converge, and at the same time it was taking a lot of time to converge using smaller step sizes. Removing discrete variables again helped, and we managed to make it converge using continuous variables and their squares by making even smaller step size, but results were very similar to those achieved by using normal equations, so we decided to not use gradient descent later.

2.5 Rigde regression

We tried to fit ridge regression for different values of λ and tried various feature transformations. It turned out (fig. 5(b)) that model works best when λ is set to zero and features are chosen the same way as with least squares method. Amongst others, we also tried fitting ridge regression to all continuous variables and their pairwise multiplication pairs, but we made our model overfitting, as shown in the figure 6(a).



(a) Errors for ridge regression model with all second order terms. Big difference of train and test error tells us that our model has big variance.

Figure 6:

2.6 Summary

We found out that using of discrete input variables in predicting the regression outcome increases the error and makes the matrix ill-conditioned. We dropped this variables completely and fit the least squares using normal equations and gradient descent and ridge regression for different values of λ. It turned out that all predictors (on model with input variables, their squares and 4-th powers) give the same rsme test error of 442, with the remark that convering gradient descent was hard due to setting the step size. Even in ridge regression, adding more variables guided us to overfitting. We finally used the normal equations predictor as the lightest computationally one.