Project-I by Group Chicago

Christophe Bertrand
EPFL
christophe.bertrand@epfl.ch

Adam Ztot
EPFL
adam.ztot@epfl.ch

Abstract

In this project report, we present the findings of our data analysis on the Chicago data. We will first state the data exploratory analysis part, which will be followed first by the regression analysis on the regression Chicago data, and then a classification on the classification Chicago data.

1 Regression

1.1 Data Description

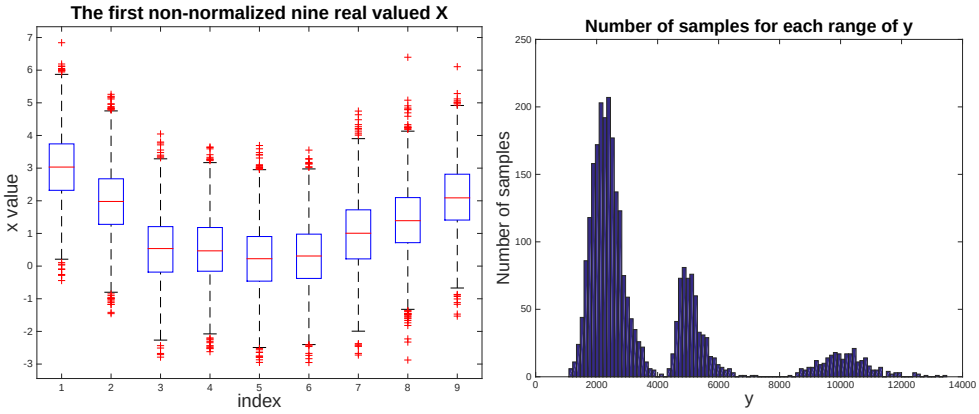
Our dataset contained some train data (X_{train}, y_{train}). We have 2800 samples available and 1200 test data. The test data are actually the different inputs for which we need to predict a value. Our data has D = 74 dimensions as inputs from which we have 61 that are real-valued variables, 10 are categorical and 3 are binary. We made some dummy encodings in our categorical data, but since it didnt improve the RMSE, we decided to discard them.

1.2 Data visualization and cleaning

As soon as we started the project, the first thing we did was to plot the data to see what we would be dealing with. What draw our attention immediately was the cloud decomposition of the data. So we were immediately tempted to create some groups out of it. But we preferred first to do some cleaning of our data since we could see that the mean and variance of the different inputs were usually different (see figure 1(a)). We normalized it to have a zero-mean and one-variance data . We also tried different transformations to the features and combined them for different features. We tried square, cube, log and exponential transformations, but none of them gave us satisfying results. So we decided that our main 2 steps would be to test our methods on the whole data, and also test them on clusters separately, and then see which ones would be the best. Because we had three clusters (see figure 1(b)), we needed two features separating them, hopefully, we had them. Those were the features 39 and 70. We separate the three clusters as we can see it in the figure 1(c). We tried to implement k-NN to split the data but our implementation didnt work so we just decided to make a simple split by looking at the two plots of figure 3. We tested some different values by checking how many points would be assigned to the wrong cluster and we finally decided to take the thresholds x_{39} = 1.3 and x_{70} = 0.53.

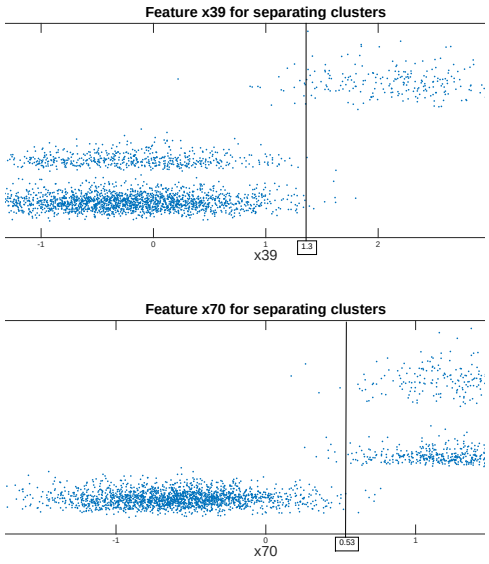
1.3 Least Squares (Gradient descent and normal equations)

After all this preprocessing was done we could finally start fitting the data. Several steps were necessary for each of the two features (whole data and clusters) : compute the model that will be fitted (mean, linear, polynomial of degree n, splines,), and then compute the associated beta. To compute the best model we were just doing a K-Fold cross-validation with K = 4. We tried some other values of K (8 and 10), with bigger ones we saw that we overfitted, but with smaller ones our model was impacted by less data, a good tradeoff seemed to be 4 for us since it was closer to the number of test data (2800/4 = 700, test data = 1200) Then we computed the beta with



(a) Boxplot of the first 20 real-valued columns of X_{train}

(b) Histogram of $\mathbf{y}$. We can clearly see that it is separated in three clusters.



(c) Features X39 and X70 which are used classify the inputs in the three clusters

Figure 1:

leastsquares (we implemented it using singular value decomposition to avoid the ill-conditioning), and with leastsquaresGD. Since it was not converging with leastsquares gradient descent, we decided to use a backtracking method to find alpha. We combined that backtracking method with the armijo condition [1], because the combination of backtracking Armijo with gradient descent guarantees us finite termination, objective unbounded, or gradient convergence, and then a convergence to a local minimum is guaranteed.

The first and simplest model we started with was the mean.

Mean :

Whole data : We just computed the mean of all the 2800 ys (mean = 3685), fit it to the model and compute the RMSE (4283, 4283).

Clusters : We computed the means of the points by cluster, and then fit it to the training data. Each input was associated to the corresponding mean according to the cluster it belongs to. Table 1

Linear Model :

Table 1: Means

Cluster 1	Cluster 2	Cluster 3
2360	5187	1015

Whole data : We fitted the linear model $\beta_0 + \sum_{i=1}^{74} \beta_i * x_i$ to the whole model, and we computed the RMSE.

Clusters : We fitted the different linear models to each of the cluster and created a full model out of it, that was applied to training data and gave us a much lower RMSE.

Polynomial :

Whole data :

We fitted a polynomial of the form $\beta_0 + \sum_{i=1}^{74} \beta_i * x_i + ... + \sum_{i=1}^{74} \beta_{d*i} * x_i^d$ and tried to choose the best max degree d thanks to the cross-validation. The RMSE we obtained were definitely better than before. The best degree we found was 2.

Clusters :

We fitted different polynomials of the previous form to the different clusters and we obtained those maximum degrees that were minimizing the RMSE (Table 2)

Table 2: Degrees associated

Cluster 1	Cluster 2	Cluster 3
K = 4 : d = 2,K = 10 : d = 2	K = 4 : d = 1,K = 10 : d = 2	K = 4 : d = 2,K = 10 : d = 2

We therefore computed different beta for each of them and computed the RMSE, which was our best RMSE without overfitting.

Table 3 shows the improvement of the error according to the model used (We decided to only put the RMSE with betas computed with leastsquares since leastsquaresGD was not improving it significantly). In the end, it seems to be great to take the model with obtained with K = 4 since it is simpler than the model obtained with K = 10.

1.4 Ridge regression

Since the rank of the training data is deficient for our data (rank = 65 instead of 74), ridge regression is also a great method to apply since it will lift the eigenvalues and then make our matrix full-rank, which will make a better condition number for calculations (lower). To perform ridge regression, we actually proceeded the same way as before, but to avoid redundancy, we will state only the kept model which was a non-linear polynomial basis. As we did before we worked with this basis trying to find the best max degree for the polynomial. Then for each max degree of a polynomial we needed to get the best lambda lifting the eigenvalues. So we had two loops which therefore included a 4-Fold cross validation loop to find the best lambda for the best polynomial. After that all we had to do was to compute the betas associated to this model thank to our ridge regression function. We performed these steps with the whole data and with the cluster separation. The results are shown in Table 4

Table 3: Evolution of the training error

	Mean	Linear	Polynomial
Whole data	4283	2061	K = 4 : 1659,K = 10 : 1659
Clusters	681	432	K = 4 : 415,K = 10 : 384

Table 4: RMSE

Whole data	1660
Clusters	420

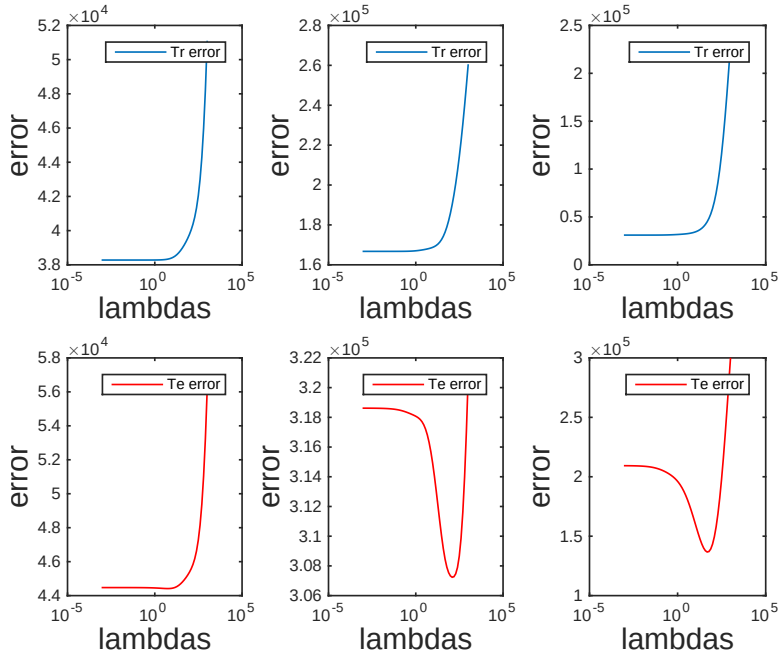


Figure 2: Ridge regression on the three different clusters

It was clear that we would decide to take the model with the cluster separation and we then plotted the different lambdas for each of the clusters for the best degree (see figure 2). We took for each cluster the lambda minimizing the mean test error.

1.5 Summary

To conclude we had to choose between two models, the best polynomial obtained with clustering with least squares (using normal equations) or with ridge regression. We decided to take the model that was using linear regression even if the RMSE was bigger since the optimization would be done better (the function is convex because full rank implies positive-definiteness). Then our model took this form :

$$M = \begin{cases} \beta_0 + \sum_{i=1}^{74} \beta_i * x_i + \sum_{i=1}^{74} \beta_{2*i} * x_i^2 + \lambda_1 * \sum_{i=1}^{2*74} \beta_i^2 & \text{if } x \in C_1 \\ \beta_0 + \sum_{i=1}^{74} \beta_i * x_i + \sum_{i=1}^{74} \beta_{2*i} * x_i^2 + \lambda_2 * \sum_{i=1}^{2*74} \beta_i^2 & \text{if } x \in C_2 \\ \beta_0 + \sum_{i=1}^{74} \beta_i * x_i + \lambda_1 * \sum_{i=1}^{74} \beta_i^2 & \text{if } x \in C_3 \end{cases}$$

2 Classification

2.1 Data Description

The data for the classification contains training data with output vector y_{train} and input matrix X_{train}. The number observation $N_{train} = 1500$ and the dimensionality of the input is $D = 30$. Out of these 30 features, 25 where real valued, two were binary and the remaining 3 were categorical with 5,4 and 3 categories.

The data also contains test data where we don't have the outputs. we have $N = 1500$ test examples. Our goal is to predict the test data and estimate the error of our predictions.

2.2 Data visualization and cleaning

We first looked at the values of y_{train}. These outputs took the values of -1 and 1, we changed the values of y_{train} to take the values 0 or 1 to be consistent with our formulas and the loss functions. The mean of y_{train} is $\bar{y} = 0.3613$.

The matrix X_{train} is full ranked, we therefore expect that the penalized logistic regression will not give us better results than the simple logistic regression. We also normalized the input to have a zero-mean and a standard deviation of one.

We use a dummy encoding for our categorical features, we therefore get a new dimensionality of 36.

2.3 Logistic regression

To get a base line for our model we first predict the outcome with $\bar{y}$.

We then compute the logistic regression and the penalized logistic regression. As expected the penalized logistic regression doesn't score better than the logistic regression. Using the dummy encoding for the categorical features showed only a very small improvement to our test error, we therefore decided to remove these features to keep our model as simple as possible. All these results are shown in Table 5

Table 5: Errors with linear model and dummy variables

	RMSE	0-1 -loss
$\bar{y}$	0.5994	0.3613
logistic regression	0.3251	0.1407
penalized logistic regression	0.3256	0.1413
logistic with dummy	0.3263	0.1367
penalized logistic with dummy	0.3268	0.1373

2.4 Feature transformation

To improve our model even further, we tried to apply a polynomial basis function to our data. Figure 3(a) shows the results we obtained after cross-validation with $K = 10$ and degrees from 1 to 4. The best polynomial is of degree 2. Figure 3(b) shows the test errors of our models. In this figure, we can see that the polynomial model is a bit better than the linear, however since this improvement is very small we decided to stay with the linear model.

2.5 Summary

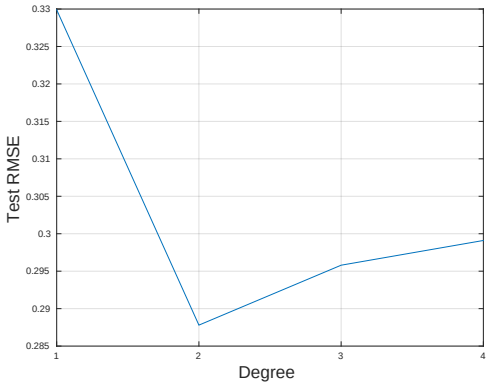
In the classification part, we removed the categorical variables. We then computed the logistic regression and penalized logistic regression. Since the matrix X_{train} was well formed, the penalized logistic regression didn't improve our test error. We then tried to improve our model with a polynomial transformation and found that the best degree is $d = 2$ but the improvement wasn't significant and we therefore decided to keep the simpler linear model.

Acknowledgments

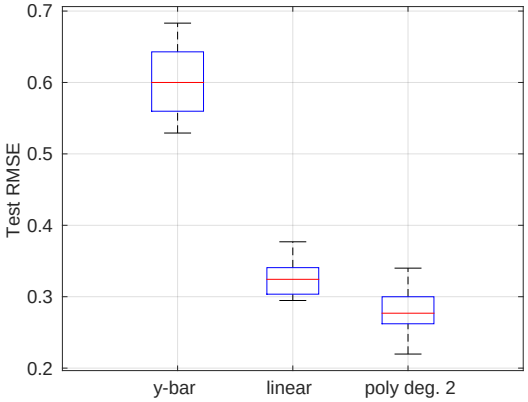
We would like to thank the PCML team for providing us the dataset and the help we needed to complete this project. All the code, as well as this report, was written by Adam and Christophe.

References

[1] Armijo rule and curvature `https://en.wikipedia.org/wiki/Wolfe_conditions#Armijo_rule_and_curvature` [online;accessed 2-November-2015]



(a) RMSE against different degrees.



(b) Comparison of test errors of $\bar{y}$ linear logistic regression and penalized logistic regression with polynomial of degree 2

Figure 3: