Project-I by Group Sao Paulo

Damien Engels
EPFL
damien.engels@epfl.ch

Noémie Jaquier
EPFL
noemie.jaquier@epfl.ch

Abstract

In this report, we study several linear methods to compute regression and classification on two different datasets. For the regression part, we used k-means clustering to separate the data in three clusters. As the input matrix is ill-conditioned, we applied a PCA algorithm to reduce its dimensionality. Ridge regression is preferred over least-squares and least-squares with gradient descent, as it has the lower bias and variance. For the classification part, we separated the data given their distribution along the dimensions of the input. Penalised logistic regression is preferred over logistic regression as it converges more reliably to a model and has lower bias and variance.

1 Regression

1.1 Data Description

Our regression-data contains $N = 2800$ train-examples and $N = 1200$ test-data. The train-examples are composed of input and output variables X and y. Each input x_n has a dimensionality $D = 70$ with 60 real-valued variables, 1 binary variable, 5 and 4 categorical variables with respectively 3 and 4 categories. The output of our test-data is not observed. Our goal is then to predict y for all test-examples and to give an approximation of the test-error.

1.2 Data Visualization and Cleaning

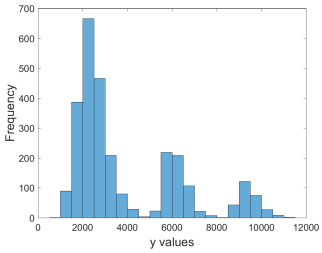
To visualise our data, we plotted histograms to see the distribution of y and of X dimensions. The input data is not centred, so that we need to normalise it.

Figure 1(a) shows the histogram of the output variable. We see that the data is split in three bursts of different sizes : a large amount of the data have small output values and only a small part have the highest values. All real-valued variables seem to have sort of Gaussian distributions, excepted 6^{th} and 55^{th} dimensions that are split in two bursts. By plotting y in function of each of these variables, as shown in Figures 1(b) and 1(c), we can see that y is highly correlated with dimensions 6 and 55 of X. The output-data can thus be separated in three clusters by looking at $x_n(6)$ ans $x_n(55)$ for each data.

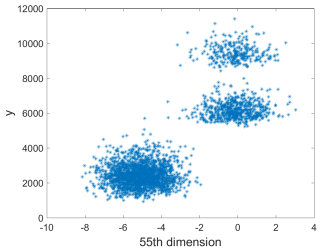
We also computed the eigenvalues of the matrix $X^T X$. As ten eigenvalues are smaller than 10^{-10}, we will need to transform X to remove dimensions corresponding to those small eigenvalues or to use methods such as ridge regression to lift them.

1.3 Clustering

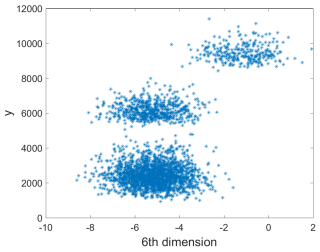
We used k-means clustering to separate our data during the training part. The clustering is always done before normalising the data. As the data are not equally shared between the three clusters, we first compute a clustering with $k = 2$ clusters using y and the most correlated dimension of X with y to compute the distance between each data-point and centre. In a second step, we keep only the data classified in the sparse cluster to make a second clustering with $k = 2$ using y and the second most correlated dimension of X. Figure 2 shows an example of clusters obtained after the training.



(a) Histogram of y. We can see three bursts of data.



(b) y in function of the most correlated dimension (55^{th}) of X with y.



(c) y in function of the second most correlated dimension (6^{th}) of X with y.

Figure 1: Regression data visualisation.

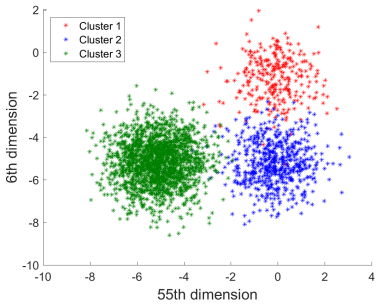
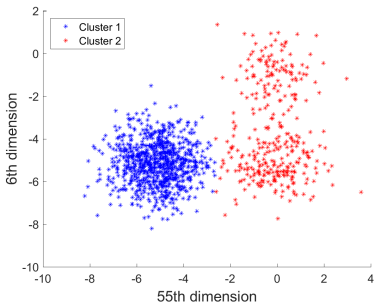


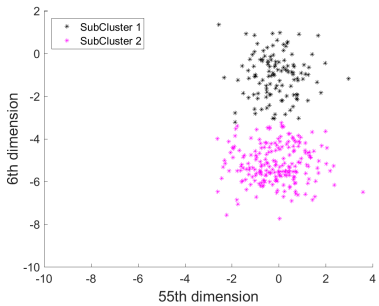
Figure 2: Clusters obtained along the two most correlated dimensions. We can see three clearly delimited groups of data.

In order to ensure a reliable clustering, we do not initialise the centres randomly. During the initialisation, we sort the data-points by their norm and separate the sorted list in k chunks, where k is the given number of clusters. The mean along each dimension is then calculated for each chunk and gives the position of each initial cluster centre.

During the test part, we follow the same principle as for the training. First, the most correlated dimension of X found during the training part is used to assign each point to one of the two big clusters, as shown in Figure 3(a). The second most correlated dimension is then used to divide the data classified in the sparse cluster in two sub-clusters, as shown in Figure 3(b).



(a) First k-means applied along the most correlated dimension of X with y.



(b) Second k-means applied along the second most correlated dimension of X with y.

Figure 3: Steps of k-means clustering for test-data.

1.4 Feature transformations

After the separation in cluster, each group of train-data is treated separately. For each cluster, the dimensions of X and y are normalised to have zero-mean and unitary standard deviation. We also implemented a method to fix the outliers and a PCA to improve our predictions.

We defined as outliers the data-points for which some elements of one or more real-valued dimensions are further than 3 standard deviations from the mean. As our data were normalised, all non-categorical elements higher than 3 were considered as outlier elements. To fix the outliers, value 3 (or -3) was assigned to those elements. Fixing the outliers with this method allows us to limit errors due to false-classified data in our model.

The PCA is applied only on the real-valued dimensions. It projects our input data in a new coordinate system based on the covariance between dimensions. It allows us to reduce the dimensionality of our data in this new space by removing the dimensions with eigenvalues lower than a defined threshold. We choose this threshold equal to 1 as eleven dimensions have eigenvalues in the order of 10^{-10} and all others are higher than 10. The data are then projected back in the initial coordinate system. In our case, the PCA reduces our input data by eleven dimensions.

After the clustering of test-data, each group is also treated separately. The data is first normalised given means and standard deviations computed during the training. The outliers are then fixed and PCA is applied by projecting the data with the eigenvectors computed during the training part.

1.5 Least-Squares and Ridge Regression

We applied least-squares, least-squares with gradient descent and ridge regression to our dataset. Using PCA allows us to solve the problem of ill-conditioned matrix, so that we can use the three methods mentioned above. Ridge regression is the only method that can be used without reducing the dimensionality of our input data.

We computed RMSE for each model with and without the categorical dimensions of X. As removing those dimensions does not change significantly the error, we decided to compute the regression only with the 60 real-valued dimensions of X.

Figure 4 shows train and test RMSE for a step size α varied from 10^{-3} to $10^{-0.5}$ with 20 points in between. The convergence condition was reached when the variation of β between two iterations was lower than 1.5α. Cross-validation was made using 66% of the data as train data and the rest as test data. Train and test RMSE are optimal for $\alpha \approx 0.2$. With higher α values, gradient descent is not able to converge and the error goes to ∞.

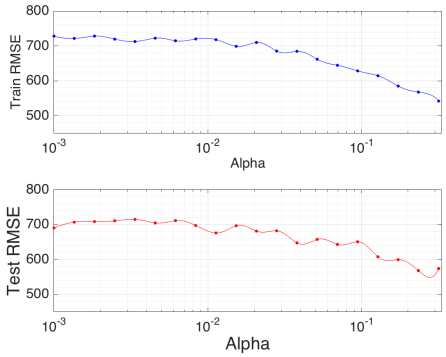


Figure 4: Least-squares with gradient descent for a 66% - 33% split with 3 iterations for each α.

Figure 5 shows train and test RMSE for λ varied from 10^{-3} to 10^{7} with 60 points in between. Cross-validation was made using 66% of the data as train data and the rest as test data. We can see that λ should be chosen between 0 and 100 to have the best prediction and that λ higher than 1000 increases RMSE and the model under-fits. For high λ values, the model and estimation bias are increased as their variance is reduced.

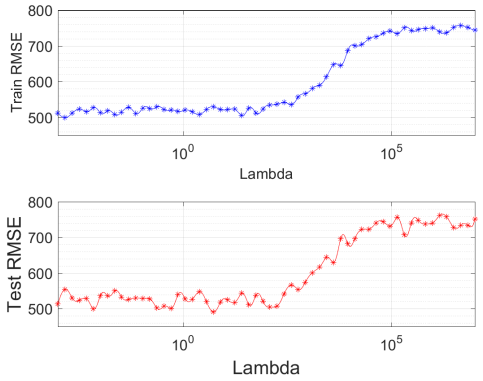


Figure 5: Ridge regression for a 66% - 33% split with 3 iterations for each λ.

Figure 6 compares three regression methods with $\alpha = 0.3$ for the gradient descent and $\lambda = 100$ for ridge regression. It is important to note that some outliers (11 for the train, 17 for the test) of least-squares method are not visible in this graph, as their RMSE is higher than 10^4.

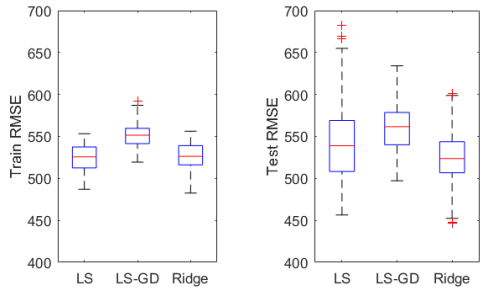


Figure 6: Boxplots of train and test RMSE for least-squares (LS), least-squares with gradient descent (LS-GD) and ridge regression (Ridge) with 100 iterations for each method.

The medians of train/test RMSE are respectively equal to $526/539$, $551/562$ and $526/524$ for least-squares, least-squares GD and ridge regression. We observe that least-squares GD has a high bias and a low variance compared to least-squares. Even if least-squares and ridge regression have similar train and test RMSE, we prefer to use ridge regression as its variance is lower and as outliers are not very far from the median compared to outliers of least-squares.

2 Classification

2.1 Data Description

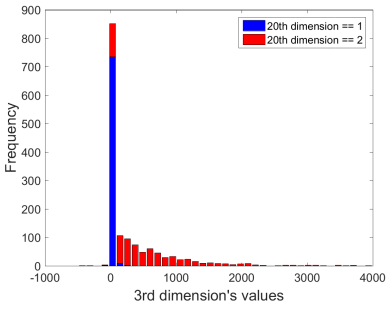
Our classification-data contains $N = 1500$ train-examples and $N = 1500$ test-data. The train-data is composed of input and output variables X and y. Each input x_n has a dimensionality $D = 37$ with 32 real-valued variables, 2 binary variables, 3 categorical variables with 5 categories. The output is a binary variable equal to 1 or -1. The output of our test-data is not observed. Our goal is then to predict y for all test-examples and to give an approximation of the test-error.

2.2 Data Visualization and Cleaning

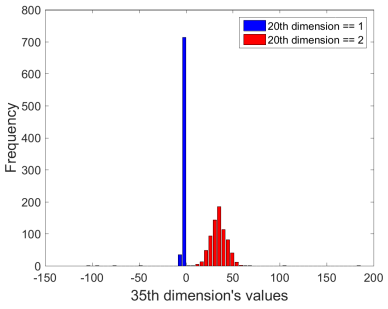
To visualise our data, we plotted histogram of X dimensions. The input data is again not centred, so that we need to normalise it.

Figures 7(a) and 7(b) show respectively the histogram of the 23^{th} and the 35^{th} variable of X. We see that the distribution is composed of a high peak and a sort of Gaussian distribution. We observed that most of the real-valued dimensions have similar distributions and that data-points belonging to the peak or to the Gaussian in one dimension are contained in the same part of the distribution in other dimensions. Moreover, the binary value of the 20^{th} dimension is different for data-points in the peak or in the Gaussian, so that we can use this dimension to separate the input data in two clusters.

We also computed the eigenvalues of the matrix $X^T X$. As the smaller eigenvalue is equal to 58.4, the matrix is full-ranked and we do not need to use methods to lift the eigenvalues.



(a) Histogram of the 23^{th} dimension of X.



(b) Histogram of the 35^{th} dimension of X.

Figure 7: Example of histogram to show the distribution of X dimensions. We can see that each distribution is composed of a peak in blue and of a Gaussian in red, separable given the 20^{th} variable.

2.3 Feature Transformations

The train-data is first separated in two clusters given the value of the 20th dimension of X. Each group of data is then treated separately. The real-valued variables of X are first normalised to have zero-mean and unitary standard deviation. We also implemented a method to remove the outliers.

We defined as outliers the data-points for which at least one element of a real-valued dimension is further than 3 standard deviations from the mean. In this case, the data point is removed from the training set.

After the clustering of the test-data, each group is also treated separately. The real-valued input dimensions are normalised given the means and standard deviations computed during the training.

2.4 Logistic Regression

We applied logistic regression and penalised logistic regression on our dataset. Both regressions were implemented with Newton's method. To improve the convergence, the step size was adapted at each time step to be proportional to the log likelihood such that $\alpha = log(\ell) * 10^{-4}$. The convergence criterion was reached when the difference between β computed during two consecutive iterations was lower than 0.001.

Figure 8 shows train and test loss function for λ varied from 10^{-3} to 1. Cross-validation was made using 50% of the data as train data and the rest as test data. We observe that changing λ does not really improve the train and test loss function. The difference of error between train and test shows that the model is overfitting. This is surely due to the amount of train data compared to the amount of test data, which is not high enough. However, we chose this split to make a cross-validation with similar train/test quantities to those we are provided to make output prediction.

Figure 9 compares two classification methods with $\lambda = 0.01$ for the penalised logistic regression. The medians of train/test loss function are respectively $0.061/0.126$ and $0.057/0.115$ for logistic regression and for the penalised method. We observe that penalised logistic regression gives smaller error and has lower variance than simple logistic regression. Even if the improvement is not really significant, we prefer to use the penalised method, as the other one must sometimes be stopped before a complete convergence because the cost function suddenly explodes to infinity.

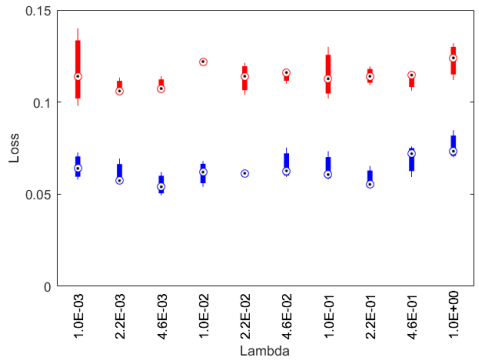


Figure 8: Penalised logistic regression for a $50\% - 50\%$ split with 3 iterations for each λ.

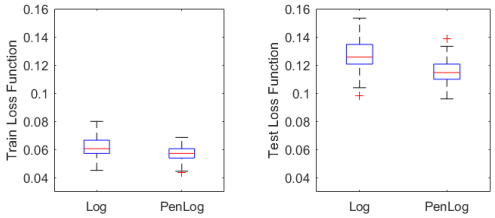


Figure 9: Boxplots of train and test loss function for logistic regression (Log) and penalised logistic regression (PenLog) with 10 iterations for each method.

3 Summary

In this report, we analysed a regression and a classification dataset. We used k-means clustering and PCA to preprocess the regression data. We found that ridge regression gives a reasonable fit with a test RMSE equal to 524 ± 20. Least-squares and least-squares with gradient descent predictions have respectively a higher variance and a higher bias compared to ridge regression.

We separated the classification data using a binary dimension that classify the data in two different distributions along the variables of the input data. We found that penalised logistic regression gives a reasonable classification with a test loss function equal to 0.115 ± 0.02 and is more reliable than logistic regression.

Acknowledgments

We would like to thank Zoé Baraschi for her support. She was there to help us in moments of darkness with food and hugs. We also would like to thank moodle to have given us this half an hour that was unhoped for. It gave us the courage and time to finish writing up this report.

We also collaborated efficiently to generate all the code and the report.

References

Applied Machine Learning course notes from Aude Billard

PCML course notes from Mohammad Emtiyaz Khan