Project 1

Abstract

In this report, we summarize our findings for Project 1, due on November 3, 2014. We were given two independent datasets: one for regression; the other for classification. In the regression dataset, two categories of observations could be clearly distinguished and were discriminated with a Gaussian Mixture Model. For each of the two groups, feature selection was performed based on various correlations. We then built regression models with both Least Squares and Ridge Regression. In the classification dataset, we decided to apply Logistic Regression after categorical variable exclusion. Outliers were removed in each fold of our cross-validation using a simple probabilistic approach. It appears that our models for both tasks predict the data reasonably well, as reflected in the computed errors.

1 Regression

1.1 Data Description

Our train-data consists of output variables $\mathbf{y}$ and input variables $\mathbf{X}$. Once the given MAT file is loaded, we have three variables : y_train as training output data, X_train as training input data and X_test as test input data. The dimensionality of data is 41, out of which first 27 are real valued while the last 14 are categorical.

We have $N = 1400$ training data samples. We also have $N = 600$ test-data samples where we do not observe $\mathbf{y}$. Our goal is to produce predictions for test samples, as well as an approximation of the test-error.

1.2 Exploratory Data Analysis (EDA)

For EDA, we had a look at scatter plots of all 41 covariates of training data with the output as well as among themselves. The purpose was to determine correlation, if any, and to find other helpful pointers/artifacts. A sample plot is shown in Figure 1. It was clearly evident that the output (last column/row in figure) was composed of two distinct clusters, which implied that there were two different output models.

Second observation was regarding the distribution of covariates. Scatter plots/histograms of the covariates showed that (a) there were no clusters within the real valued covariates, (b) their distributions were mostly gaussian-like, (c) most had no correlation with each other and even when there was a correlation, it was linear, (d) there was no visual evidence of quadratic or other non-linear correlations. The last observation was corroborated in model fitting as well.

1.3 Strategy

With two output clusters, there were evidently two models at play. This meant that we had to train two separate regression models, one for each output cluster. To that end, first we built a classification model that would map the input data to one of these clusters. For this purpose, since the clusters had near Gaussian distribution, we ran a Gaussian Mixture Model on training output (Figure 2). The training output was thus labelled as one of the two clusters depending on the GMM. Subsequently an SVM was trained to find a model that would map an input to either of these clusters. The cross-validated missclassification rate for the SVM was around 3%.

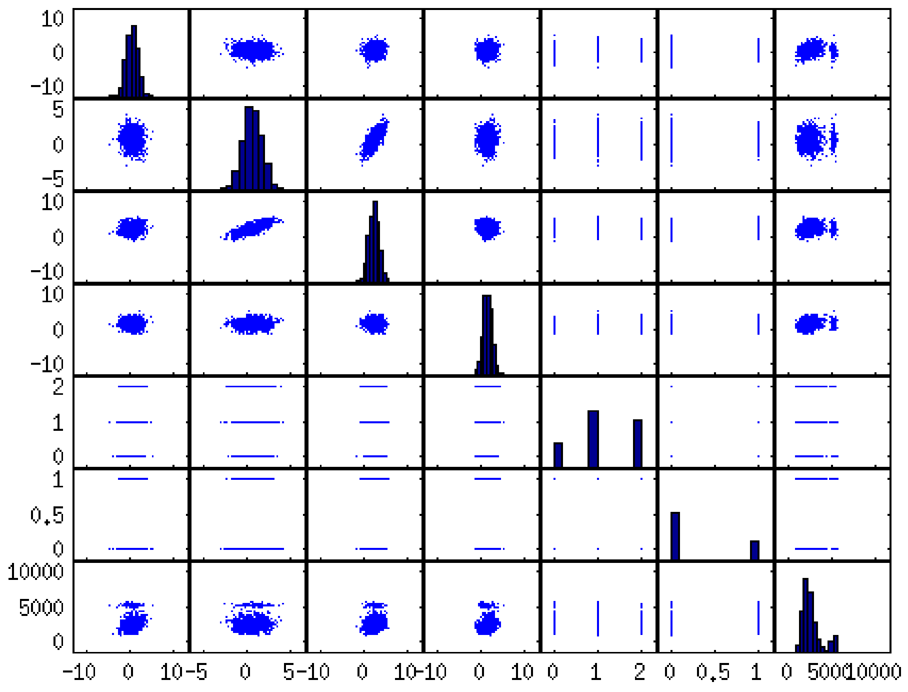

Figure 1: Scatter plot of few sample covariates and output

1.4 Regression Models

To find the two regression models, we used various permutations of the input data. These permutations involved taking each variable individually; taken two at a time; taken three and more at a time etc. Although we did not see any evidence of quadratic behavior, we included pure quadratic terms to corroborate our visual observation. For the 14 categorical/binary variables, we introduced k-1 dummy variables for each k-way category.

For analysis of each of the different permutations described above, we used R^2 statistic to find the explaining power of each model. This statistic is defined as $R^2 = \frac{RSS_{total} - RSSmodel}{RSS_{total}}$. As can be seen fromt the formula, the closer a model's RSS value to RSS of raw data, the better it describes the variance in the data. With no noise, the best model will have R^2 of 1.

The reason to chose R^2 was to have an additional measure of 'goodness of fit' of a model. There was no chance of overfitting because (a) the inherent dimensionality of data was too high as compared to that of selected models, and (b) we used cross-validation as additional safety measure.

Considering the bigger cluster as 2 and smaller cluster as 1, the final regression models are as follows:

Model 1: $y = 4317 + 95x_{22} + 66x_{24} + 71x_{25}$

Model 2: $y = 3609.7 + 22.56x_1 + 50.57x_2 + 168.62x_4 + 97x_5 + 23.35x_7 + 26.84x_8 + 339x_{12} + 36.8x_{13} + 221x_{16} + 103x_{17} + 90x_{18} + 39x_{19} + 55x_{20}$

1.5 Test Error Rate

To determine the error rate, I divided my training data into three subsets. Of the 1400 training samples, the first 1200 were used for cross validation of different models and the last 200 were used to determine an estimate of test error rate. The RMSE is 297 with std dev of 60.

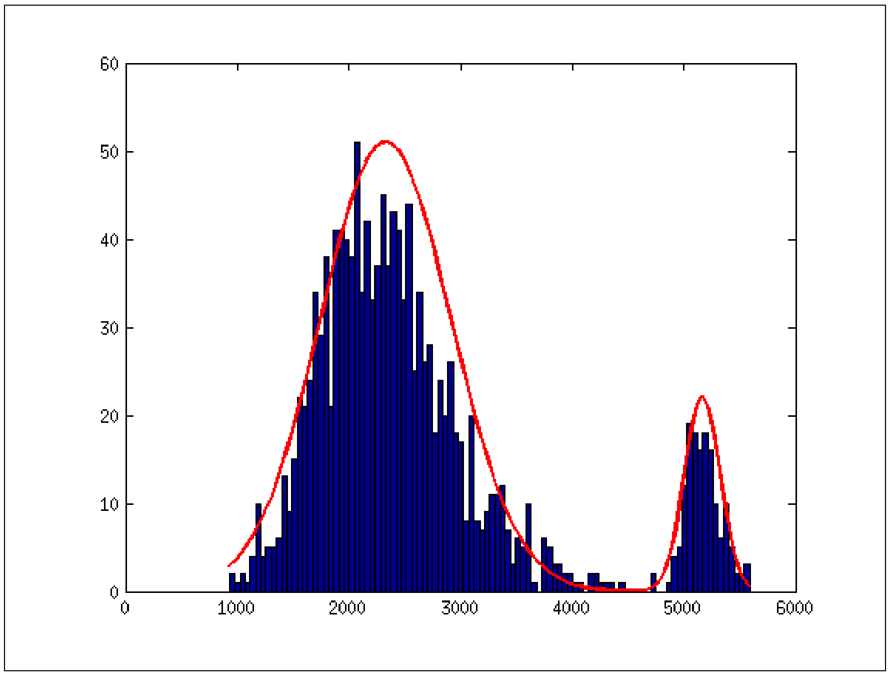

Figure 2: Histogram of y_train with a scaled superimposed Gaussian Mixture Model (k=2)

2 Classification

2.1 Data Description and Visualization

Our train-data consists of a vector of output variables $\mathbf{y}$, which take binary values (1 or -1), as well as a matrix of input variables $\mathbf{X}$. There are $N = 1500$ data examples. Each input vector $\mathbf{x}_n$ is of dimensionality $D = 34$. Out of these 34 variables, 25 are real-valued, while 9 are categorical. Out of these 9 discrete variables: 1 takes eight values, 1 takes seven values, 3 take six values, 1 takes five values, 1 takes four values, 1 takes three values, and 1 is binary.

Our test-data only consists of a matrix of input variables $\mathbf{X}$. There are $N = 1500$ test examples. Our goal is to classify these observations to output 1 or -1 with the lowest error rate. In this part, we compute the 0-1 loss error, the log-loss error as well as the root mean square error.

When plotting the output versus each input variable individually, we do not observe any visual trend. We confirmed this observation by computing the covariance matrix for each input variable. Variable $\#29$ showed by far the best correlation with $R^2 = 0.31$, but these correlation values did not help us much in our analysis. Also, we noticed that input variables do not correlate with each other.

2.2 Output Transformation and Feature Engineering

Before all, we nulled all outputs with value of -1, so that the output is either 0 or 1. This helped the general understanding of the task, since the logistic function returns a scalar in the range [0, 1].

For our feature engineering, we analyzed separately the real-valued and the categorical variables. We first normalized the real-valued variables, which significantly improved the errors we will describe later in this report. We tried many feature transformations such as log(x), sqrt(x), x^2, exp(+/-x), and products between input variables, but did not find any relevant improvements.

The right approach for dealing with categorical variables is to encode them with dummy variables, using the matlab function `dummyvar`. We did transform our 9 categorical variables into 38 dummy variables, but it appears that this transformation did not improve our errors. We even found better results when categorical variables are not included at all in the model. Therefore we decided to consider only the 25 normalized real-valued variables for our classification model.

Note that, for each fold of the cross-validation (CV) process, we will remove some outliers before computing the coefficients using a probabilistic approach (see Section 2.4).

2.3 Split Training Data into Model Data and Real-World Data

Traditional cross-validation use the complete data for both training and testing alternatively. Here, we split our data into two parts: one for creating our model; the other to be kept untouched on the side as a "real world" dataset. This approach prevents us from being overly confident about having good test error values, since it allows for computing more realistic error values on an unused dataset, which could mimic a test dataset whose output is unknown.

Following this theoretical speculation, we varied the proportion of data to be put aside as a "real world" dataset. A model-world ratio of 90:10 was proved to be efficient at keeping enough data both for an optimal training process and for having sufficient examples in our "real world".

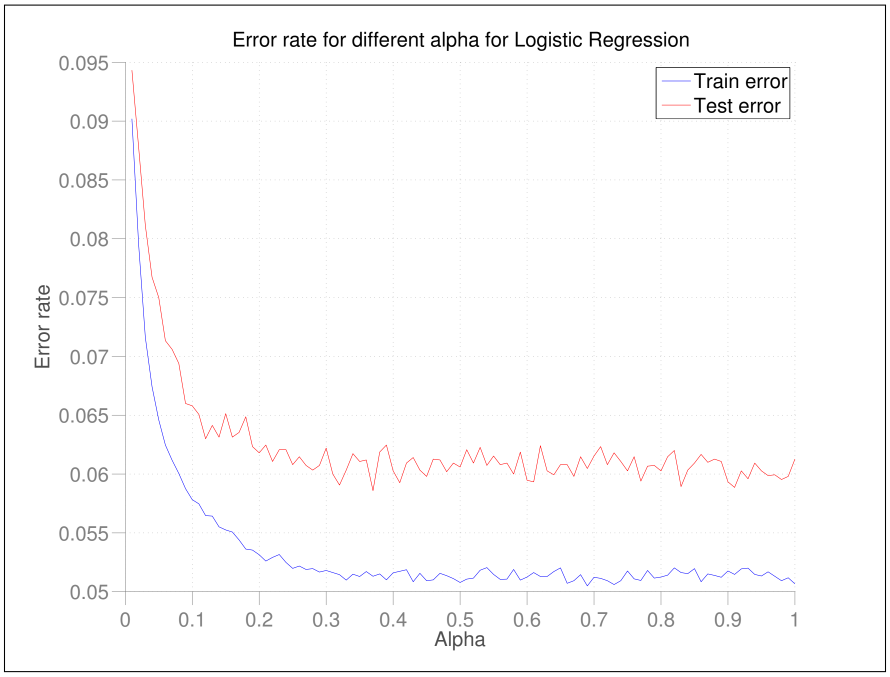

Figure 3: **Errors versus step-size**. The variations of the train and test errors are shown for different values of the step-size α. A steep decrease in error rates is observed for small values of α with a stabilization around $\alpha = 0.3$. The choice of the exact value will then depend on the other parameters.

2.4 Logistic Regression, Cross-Validation and Outlier Removal

In order to account for the variabilities arising from the random fold selection, our core algorithm computed 10 different trials of a 10-fold cross-validation using logistic regression. We decided to choose traditional gradient descent (GD) instead of Newton's method because results were similar for both methods, and we preferred to work with our implementation of GD. In Figure 3, we observe the curves for train error, test error and world error, when varying the step-size α. The curve for test error stabilizes from $\alpha = 0.3$, and we chose $\alpha = 6$ depending on the other parameters.

The three error curves should be interpreted as follows: the train error represents the averaged ratio of the number of wrong predictions over all examples (i.e. 0-1 loss) when predicting the training data during CV, which were used for constructing the model; the test error is the averaged 0-1 loss when the predicting testing data during CV; and finally the world error is the 0-1 loss when predicting the untouched test data using an averaged vector of coefficients β, and thereby represents the actual test error. Test error is the parameter to be optimized, while world error is a close approximation of the error value we should find when predicting completely new observations. Note that world error appears to be higher and to have more variance than test error.

An important step in the improvement of our errors was to remove a balanced amount of outliers before running logistic regression. For this, we designed a simple but efficient probabilistic approach. For each fold of the cross-validation and for each variable, we fitted two Gaussian distributions: one for all examples whose output value is 1, and the other for those whose output value is 0. We removed any data example whose probability is lower than an arbitrary threshold in the corresponding Gaussian distribution. In Figure 4, we observe the same curves as in Figure 3 when varying the threshold over a broad range. This threshold was set to 0.09 as it minimize the test error.

Finally, as observed in Figure 5, the use of penalized logistic regression did not prove useful, since no trend came to light when increasing the value of lambda. The variations observed when varying lambda are most probably due to random changes between every trial.

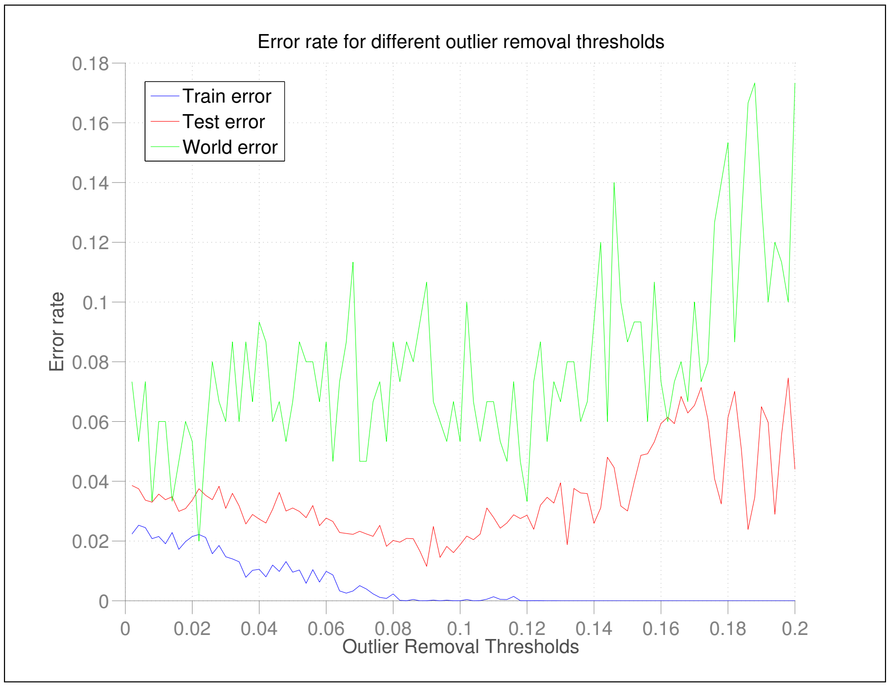

Figure 4: **Error versus threshold for outlier removal**. Both train and test errors vary with changes in the threshold value, reaching a minimum around 0.09, after which the test error rises again. World error shows its typical pattern with high variance and a clear increase for threshold higher than 0.12.

2.5 Classification Summary

The data presented to us did not show any apparent pattern or visible discrimination. Nevertheless, applying some transformations to it made it useful to compute our model. After separating our training data into a model part and an untouched "real world" part, we removed outliers from our model data set using a probabilistic approach in each fold of the cross-validation. Our classification model was optimized with the following parameters: α = 6 and threshold = 0.09 using the non-penalized logistic regression.

Our final model computes a training error close to 0 and a test error of around 2%. Our world error rate shows much more variations but is in the range of 3 to 7%. As suggested by those error rates, our model seems to predict the data reasonably well.

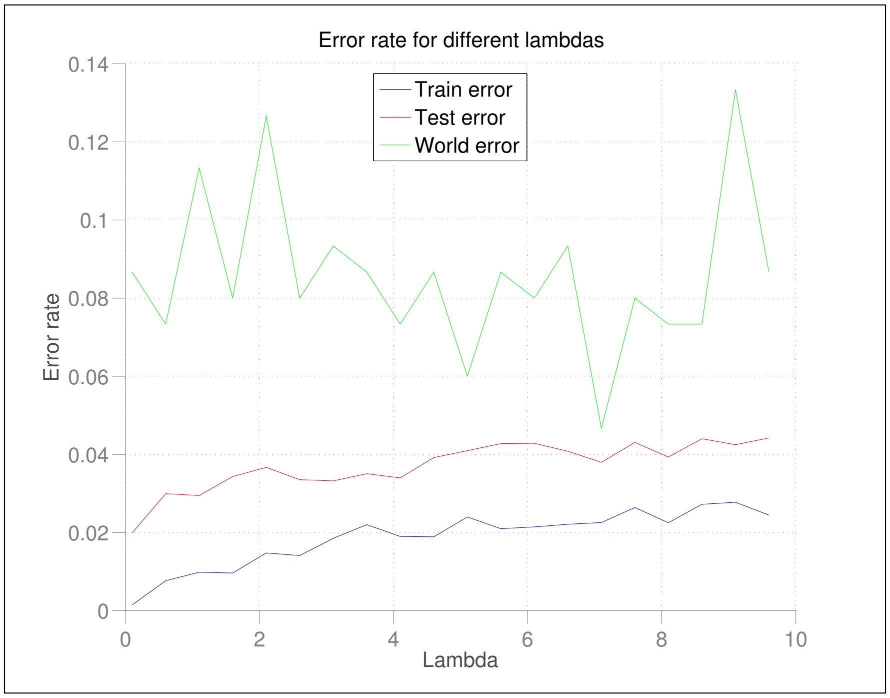

Figure 5: **Errors versus lambda for penalized logistic regression**. Train and test errors increase with λ coefficient, demonstrating the unusefulness of using penalized logistic regression.

3 Conclusion

Classification and Regression become challenging on real world data, especially in presence of categorical variables.

On one hand, in the case of regression, model selection is probably the most time consuming task. With N covariates, there are as many as 2^N combinations of all possible interactions. In such a huge search space, it becomes imperative to use heuristics like R^2 error and cross validation to prune this search space to a manageable size. Another method to prune this search space is to do exploratory data analysis. In our case, EDA resulted in the important discovery that there were in fact two output models. It also gave an indication that probably there are no quadratic/interaction terms. Both these intuitions helped greatly in reducing the search space.

On the other hand, feature selection and outlier removal were the keys to build an efficient model for classification. Our cross-validation process allows us to properly estimate new observations.

With the data given to us, we managed to implement all the basic models presented in the Pattern Classification and Machine Learning course. We made some promising projections on model extensions and tried many of them with sometimes surprising effects.

Ultimately, the complications encountered during this project allowed us to understand more deeply the issues and theory in machine learning and appreciate its complexity and often simple but clever solutions.