Project-I by Group Sydney

Diego Antognin and Jason Racine
EPFL
diego.antognini@epfl.ch, jason.racine@epfl.ch

Abstract

1 Regression

1.1 Data Description

The train-data for regression consists of $N = 2800$ input ($\mathbf{X}$) and output ($\mathbf{y}$) data samples. Each input sample is a vector $\mathbf{x}_n$ with dimension $D = 76$. Out of these 76 variables, 63 are real, 3 binary, 4 categorical with 3 categories, 6 are categorical with 4 categories.

We also have test-data of size $N = 1200$ without their corresponding output. Our goal is to produce predictions for those data, as well as an approximation of the test-error.

1.2 Data visualization and cleaning

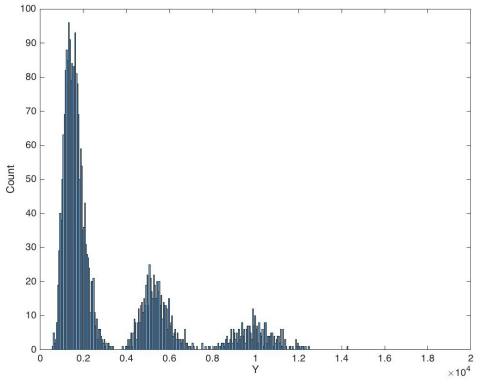
We first have plot the distribution of our features (plot not shown because too big for the 76 features). As expected, they are not center and we should normalized them (each cluster will be normalized independently). The Figure 1(a) shows an histogram of the output ($\mathbf{y}$) and we can conclude our data seem to be a combination of three Gaussian distributions. It will be used later in order to separate the data in three sets and apply different regression models on them. We can also observe that each cluster have different sizes : 1946, 576 and 278. Moreover, on the right, we can see some data points (there are 2) which have a higher values than the others. We consider them as outliers and we will remove.

To separate the data, we have observed that the feature 2 and 16 could help us. Figure 1(b). We can observe 11 misclassified data (green points), which we will remove them in order to not corrupt our model. So, this feature can allow us to find the first cluster. For the two others clusters, we need to oberve the feature 16. Figure 1(c). We can observe 14 misclassified data (green points and one blue). They also will be consider as outliers and remove. We set the threshold in order to minimize the number of misclassified data samples. Those thresholds are 0.42 for the feature 2 and 1.17 for the feature 16. So the final size of our clusters are $1937, 563$ and 273.

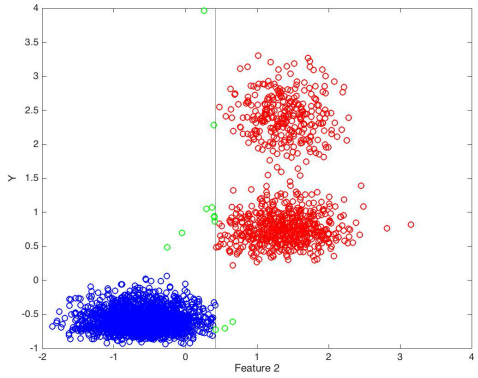
We are also interested about the correlation between the input and output variables. We have observed the correlation for each cluster and conclude that for the first cluster, they are mainly in $[-0.1, 0.1]$, except two features which are highly correlated. For the second and third cluster, also mainly in $[-0.1, 0.1]$ but this time, there are more correlated features (15). Moreover, the features seem not have correlation between them.

We use dummy encoding for the categorical variables (for a categorical variable of size k, we need $k - 1$ features), which gives us a total of 93 input variable.

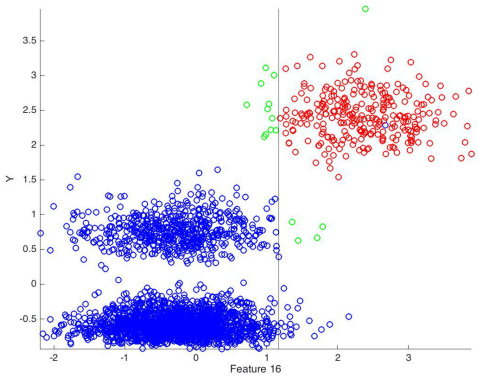
We can note that the rank of our input matrix $\mathbf{X}$ is rank-deficient with a rank of 66 instead of 76, and is rank-deficient of 65 for each cluster.



(a) Histogram of $\mathbf{y}$. We can see three Gaussian distributions and also some outliers on the right.



(b) Feature 2, because we have assumed that the data was generated from three Gaussian, this feature can help us to separate the data. Green data points are misclassified data. The separation is at $x = 0.42$.



(c) Feature 16, because we have assumed that the data was generated from three Gaussian, this feature can help us to separate the data. Green data points are misclassified data. We can see see a blue data point which is misclassified, it will be an outlier. The separation is at $x = 1.17$.

Figure 1:

1.3 Ridge Regression

Since our matrix is ill-conditioned, using the least squares with normal equation wasn't suitable. Moreover, the results obtained were less good (not so much) compared to the ridge regression method, and so we will not report the results obtained with this method. The method using the least squares with gradient descent was appropriated, but compared to the ridge regression, it was a little bit less efficient and was also much slower to tune correctly the learning step. This is the reason why we will not report the results using this method. Therefore, we have so used the ridge regression method, which is suitable because we should lift the eigenvalues, faster and a little bit more efficient than least squares using gradient descent. Only results obtained with this latter will be reported.

1.3.1 Evaluation methods

In order to evaluate our different models, we first have split our data ($\mathbf{X}$) in two sets of 80% and 20%, for the training and the test sets. We choose this percentage in order to have still enough data for the training set and also enough in order to estimate correctly our models on the testing

set with a concrete size. We learn with our models only on the training set and use the test set in order to estimate the error using the *Root Mean Square Error*. Moreover, we have also used k-cross validation (see next section for the values of k) to tune the parameters like lambda of the different polynomial degrees. We repeated the experiment 30 times with different seed in order to split the data in a unique random way for each trial. Finally, we made varied the lambda from 10^{-5} to 10^{5} with 500 values between.

1.3.2 Model comparison

The figure XXX represents the *RMSE* for each models. The first model used in the constant one, which will allow us to compare improvements with others model. It simply returns the mean of the outputs and doesn't depend on the inputs. Our second model does a ridge regression on the overall data using 10 fold cross validation, because we have a lot of data, splitting them into 10 fold seems reasonable. As expected, this is much better than previous model because we take into account the input features. Next model is similar to the second one except that we add normalization on the real inputs (it doesn't make sense to also normalize categorical variables). Normalization is generally a good thing and often a necessity depending on the algorithms used (e.g. gradient descent ones). It improves only slightly but because normalization is a good practice, we keep it.

At this stage, we want to try to separate our data into cluster. The fourth model consists of using a constant model as in first but this time separating the data into the three clusters identified during the exploratory data analysis. We can observe a big improvement, justifying our choice to split the data in clusters. The next model is about doing a ridge regression per cluster, each one having its own lambda and its own normalization. We also use k-cross validation where the first cluster has a $k_1 = 10$ (1937) because we have a lot of data, the second $k_2 = 6$ (563 samples) and the last $k_3 = 4$ (273) because it doesn't have so much data and so we have to choose a smaller k. As expected we obtained another big improvement justifying again the choice of splitting the data.

The sixth model consists of using dummy encoding for the categorical variables. As explained in the exploratory data analysis, we use $k - 1$ features to represent a k categorical variable. This is a common practice. Unfortunately, we didn't get major improvements.

The two next models is about using polynomial basis functions for each cluster. Categorical variables aren't concerned, they are just replicated for each degree, because it doesn't make any sense to build polynomial for them. We normalize only when we did our feature transformations, in order to avoid loosing information. We have to be careful about the polynomial to avoid the case where $D > N$ (especially for the cluster two and three, because there are much smaller than the first one). The seventh model doesn't use dummy encoding and the eight yes.

For the seventh model, the degrees of the polynomial for the clusters are $3, 6$ and 3. When we add higher degrees, we can directly observe overfitting because suddenly the training error decreases to a value near zero and the test errors grows exponentially, so with this manner we know that our degrees are not to high. The eighth model is similar to the seventh one, except that this time it uses dummy encoding and has lower degrees : $3, 5$ and 2, due to the number of input variables (92 vs 76). We can see that these feature transformations significantly improve the prediction accuracy compared to others model. However, dummy encoding seems to bit a bit more efficient.

1.3.3 Feature transformations

In hope to minimize the error, we tried different feature transformations. The first try was add polynomial of them self as $X = [XX^2X^3...]$ for which each cluster has is own degree. We made varied the different degrees and also the lambdas in order to keep the lambda which minimize the test error. We also try feature transformations where the degree is a root, but unfortunately it was worst. We also wanted to try to just pow features with different degrees (without add polynomial of themselves) but we couldn't do it due to a lack of time.

We also thought about removing some uncorrelated features, we tried to remove the most uncorrelated then remove also the second one etc, but unfortunately we didn't get improvements. The reason is that maybe a feature may not be correlated directly with the input, however a combination of uncorrelated feature might be highly correlated.

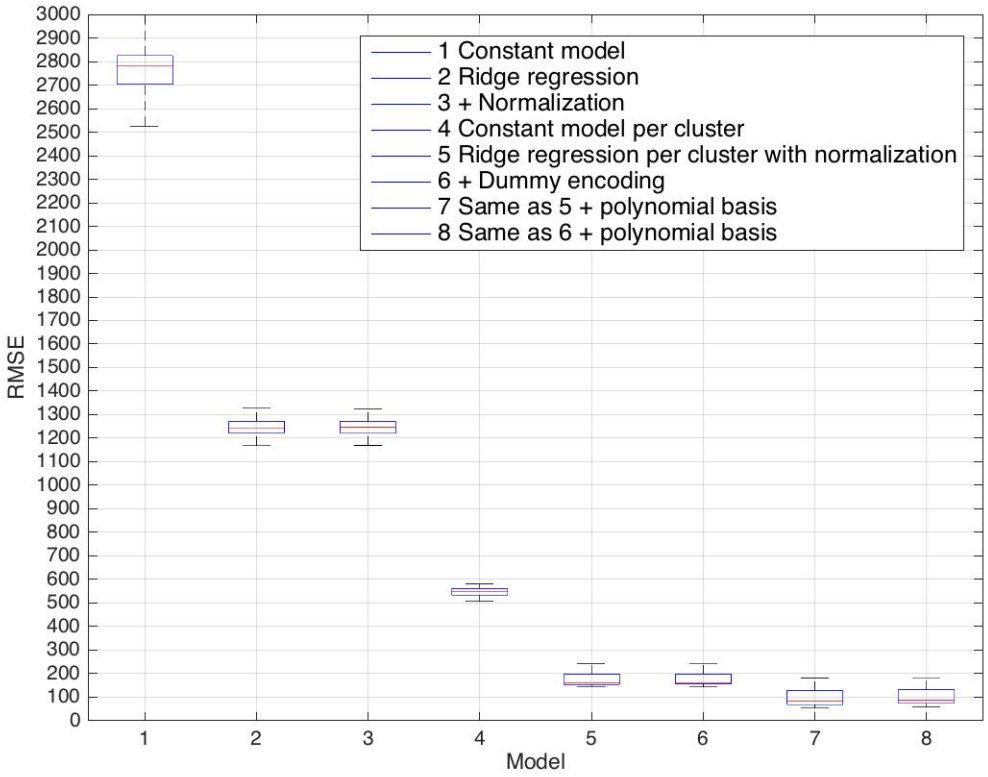


Figure 2: Boxplot of our models, using the RMSE. Each model is represented with a box where we can see their mean and their standard deviation.