736 Assignment 3

Brian McCrindle - 001406300 - mccrinbc

March 24rd, 2020

Question 1

1A

The AutoMPG data is comprised of 7 distinct features where the relationships between these variables are unknown. Our goal is to develop a relationship between the miles per gallon (MPG) and the predictor variables through a set of Gaussian Radial Basis Functions (RBFs). For this implementation, we ignore the potential benefits of regularization for now. We normalize all continuous and discrete variables using a Z-Scoring method by assuming that each predictor variable comes from an underlying Normal distribution and as such, center their distributions around zero and dividing by the distributions standard deviation.

A common issue in Machine Learning is determining how many basis functions to use and which basis functions will be effectively model the true mapping. As such, the assumption is that one of the basis functions will be able to properly describe the true relationships [1]. To determine the appropriate number of basis functions, we iterate through [5:95] in increments of 10. The follow figure illustrates the RMS training and testing errors as a function of the number of RBFs used.

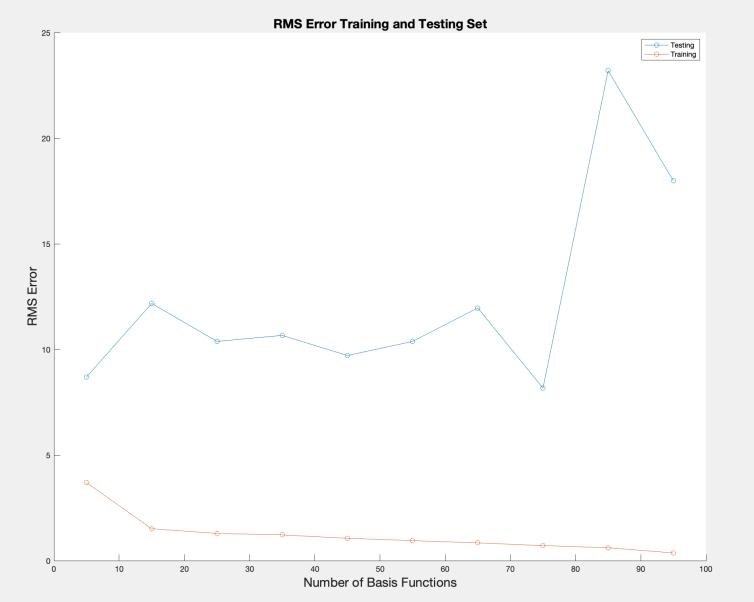

Figure 1: Training and Testing RMS Error vs. Number of Basis Functions

In model evaluation, we are looking for a model that is able to have a low training error, generalize well, and provide higher errors with the testing data relative to training but still low on the absolute scale. This is a good sign that our model has not overfit. A model with low training and drastically high test error is a sign of overfitting. Underfitting is seen with high testing and training error. In this case, we see that the

training error tends towards zero with an increasing number of basis functions, but the testing error trends upwards with this increase. The point of the lowest testing error occurs at 75 basis functions.

1B

In this implementation, we fix the number of basis functions to 90 and incorporate L2-Regularization to the loss function to help with overfitting. We iterate through λ regularization values from [0,0.01,0.1,1,10,100,1000]. The figure below shows the RMS error for the training and testing sets on a semi-log plot.

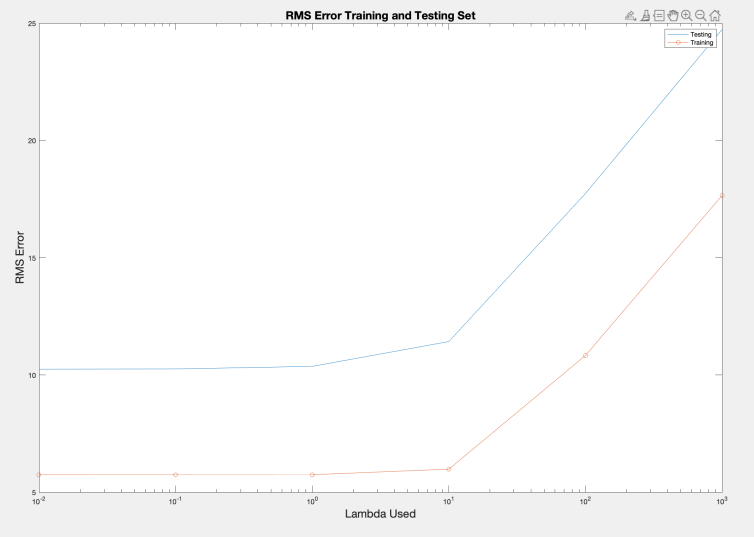

Figure 2: Train and Test Error vs. λ Value

From what we gather from the plot above, the dataset is not clearly overfit to the training data since the test error is higher but still within the same order of magnitude as the training error. Therefore, the most optimal regularization value is $\lambda = 0.001$ (or as close to zero as we can get).

Question 2

In this situation, we're looking to determine the year in which a song was released based on predictor variables related to timbre. Since there a considerable number of features within the dataset that can be used and we're not employing any feature reduction scheme (such as Principle Componant Analysis (PCA)), we will use lasso regularization to prevent overfitting. We use Matlab's built-in *lasso()* function to provide the appropriate weights for each of the predictor variables. For this example, we're interested in determining the optimal value of the regularization coefficient λ. The figures below show the RMS training and testing error as a function of the regularization value [2].

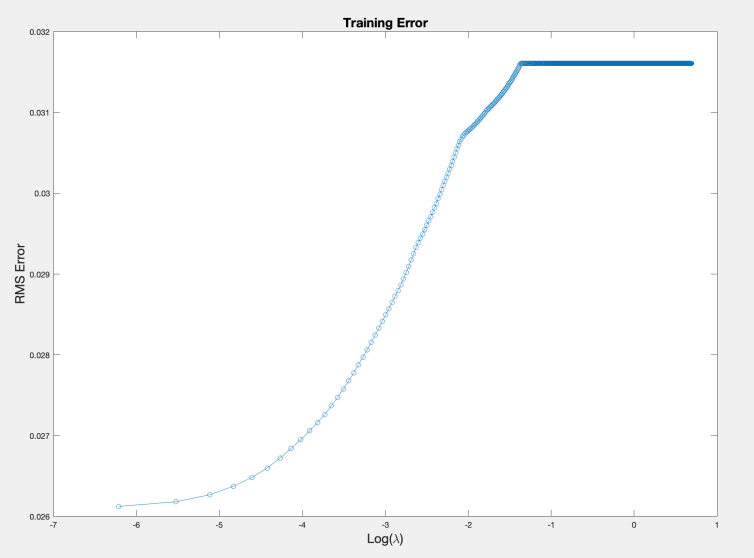

Figure 3: RMS Training Error vs. $\log(\lambda)$

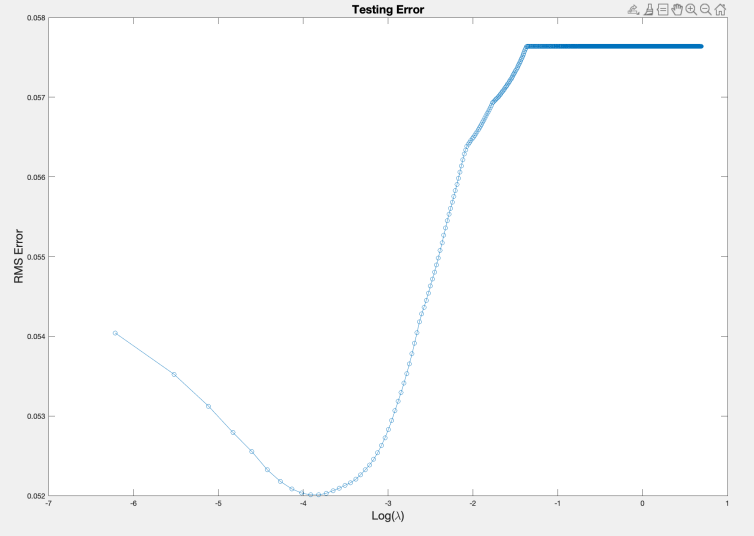

Figure 4: RMS Testing Error vs. $\log(\lambda)$

Seen through the testing error, we will achieve a global minimum when the regularization value is **0.0220** which results in training and test RMS errors of 0.0272 and 0.0520 respectively. Along with this, we can show the total number of non-zero coefficients placed over the feature space.

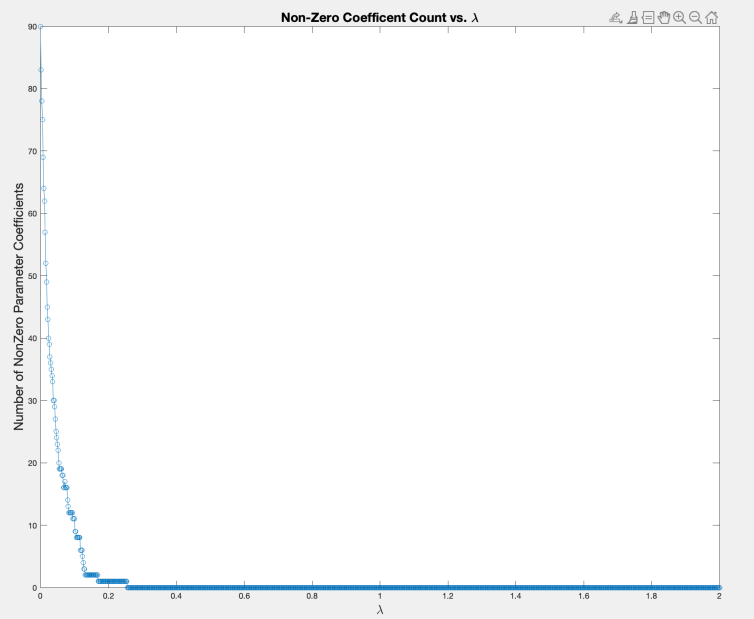

Figure 5: Number of Non-Zero Coefficients vs. λ

We see that with a regularization value of 0.0220, the majority of the features are assigned zero weights, leading to the conclusion that only a subset of the features are important for prediction. As such, if we were to employ methods such as PCA, we could potentially bump up our predictive performance with less chances of overfitting.

Question 3

3A

With five relevant features related to the different portions of the crabshell, we can train a logistic regression model for prediction purposes. Using Matlab's *rescale()* function, we map each of the five features to an output range of [-1,1]. We apply a 75/25 train/test split using random shuffling. We set a default seed in Matlab for reproducability purposes.

As we did before, we use a set of radial basis functions (RBFs) to map the features into a non-linear space to find a linear separator. We use 5 basis functions where $\phi_0 = 1$ to account for the bias. To determine the optimal weights for the logistic regression, we determine the maximum likelihood estimate (MLE) using the Newton-Raphson method [3].

$$W_{new} = W_{old} - (\Phi^T R \Phi)^{-1} \Phi^T (y - t) \tag{1}$$

Where W, Φ, R, y, and t are the weight estimates, a diagonal matrix of likelihoods, the label estimates, and the true labels, respectively. To make sure we find the minimum for W, iterate over an $\mathcal{N}$ number of training EPOCHS, each time passing in the randomly chosen training set. With the optimal W, we apply this onto our testing data, transform it through the Sigmoid() activation function, and compute the RMS error between the predicted and true labels. When doing this, we find that we arrive at the following confusion matrix.

$$\begin{bmatrix} 18 & 6 \\ 14 & 12 \end{bmatrix}$$

Where indices (1,1), (1,2), (2,1), and (2,2) are the True Positive, False Negative, False Positive, and True Negative positions, respectively. With this, we have an accuracy of 60% which is better than random guessing.

3B

We can compare the performance of the logistic regression classifier to a Support Vector Machine (SVM). In the simplest SVM case, a hyperplane can be developed to separate a binary feature space by maximizing the margin between the hyperplane and the support vectors. In cases where the feature space in unable to be linearly separated, or cannot be separated with perfect classification, we can again apply a non-linear kernel to transform the feature space and specify the degree in which we accept potential misclassifications. This is known as a *soft* margin. The soft margin will help the algorithm converge and prevent overfitting. This is generally more useful since the amount of training samples is not vast enough to cover the entire feature space [4].

For this implementation, we use a set of RBFs with a soft margin to maximize our probability of finding the most representative hyperplane. Non-linear RBFs in particular map the finite feature space into an infinite-dimensional feature space where the data is *always* linearly separable. When applying these transforms and conditions through Matlab's *fitcsvm* and *predict* functions, we arrive at a confusion matrix as follows.

$$\begin{bmatrix} 9 & 15 \\ 16 & 10 \end{bmatrix}$$

With these results, we can see that we have an accuracy 38%, which is worse than random guessing. This is clearly an improper fit due to either the algorithm not converging or an insufficient amount of training data for the correct margin to be learned.

References

[1] Volker Tresp, *Basis Functions*, https://www.dbs.ifi.lmu.de/Lehre/MaschLernen/SS2016/Skript/BasisFunctions2016.pdf

[2] Neil Radford, *http://www.utstat.utoronto.ca/ radford/sta414.S11/week1b.pdf*, http://www.utstat.utoronto.ca/ radford/sta414.S11/week1b.pdf

[3] Christopher Bishop, *Pattern Recognition and Machine Learning*, https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf

[4] StackExchange, *Soft-Margin SVM*, https://stats.stackexchange.com/questions/94003/does-using-a-kernel-function-make-the-data-linearly-separable-if-so-why-using

[5] Sharam Shirani, *2018-prml-slides-3/4-shahram: Regression, Classification*, AvenueToLearn.