Analysis of Regression Predictors for the Wine Quality Dataset

Paul Streli
CID: 01103106
EEE, Imperial College London
ps4715@ic.ac.uk

1. Introduction

The dataset of choice for this report is the Wine Quality Data Set, which is provided by the UCI Machine Learning Repository [1] and was originally used for a machine learning research paper by Cortez et al. [2]. The dataset is based on 1599 red wine and 4898 white wine variants of the Portuguese *Vino Verde* wine. For each sample, the following eleven physicochemical attributes were measured: *fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol*. Moreover, the *quality* was determined by taking the "median of at least 3 evaluations made by wine experts" [1], who gave a score between 0 and 10.

The goal of this report is to use machine learning techniques to find a suitable hypothesis, that is able to predict the quality of a wine based on a given set of input features. Since the output ranges from 0 to 10, it was chosen to estimate the quality using regression algorithms and empirical risk minimisation. The project was realised in Matlab.

2. Data import

The first task was to import the individual white wine and red wine datasets to matlab and combine them in a single table. To be able to profit from the advantages of both matlab matrices and tables, the different datasets are stored and transformed into different formats during the execution of the programme. Each row in the table corresponds to a single wine sample and each column corresponds to a feature or the wine quality output. For reproducibility, the random seed is set to a fixed integer. As the final hypothesis should work for the combined wine dataset, it is important to treat the samples as coming from a single dataset. As the type of wine is known and could potentially be useful for prediction, it is added to each sample as additional attribute in the first column in the form of:

$$\text{red wine} \mathrel{\widehat{=}} 0\,,\ \text{white wine} \mathrel{\widehat{=}} 1$$

Even though the original data description states that the wine quality ranges from 0 to 10, the data set does not contain any samples with a wine quality of 0, 1 or 2 (very poor quality) or 10 (maximum quality) (Fig. 1). Moreover, the great majority of samples have a quality rating between five and seven. This is not suprising considering the fact that the median of several ratings was taken for each sample and that valuations potentially vary due to individual taste preferences. Therefore, it will be difficult to predict the correct output for wines with very high or low ratings, since the chosen data set does not contain enough samples of these categories to learn from. This bias will not become apparent from the test error, since the test set will not include these categories of wines as they do not appear in the general data set.

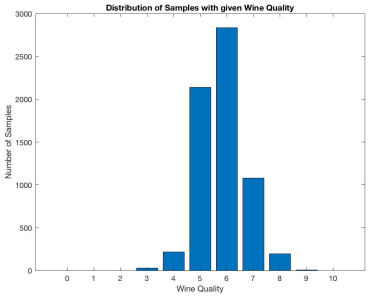

Figure 1 Distribution of Wine Qualities

3. Selection of a suitable Loss Function

Since the performance of the final predictor is determined by its test error, a loss function needs to be chosen that reflects the deviation of the prediciton from the real output. For all further analysis, the squared error is taken as loss function of choice. It is defined as $(h(\boldsymbol{x}) - f(\boldsymbol{x}))^2$, where $\boldsymbol{x}$ is the input feature vector, h is the selected hypothesis and f is the true underlying function. The test error is then defined as the mean squared error between the predicted outputs of the final hypothesis trained over the whole training set and the actual test outputs. The cross-validation error is the mean squared error between the predicted outputs of a hypothesis trained on the CV-training set and the CV set outputs, while the training error is the error between the predicted outputs of a hypothesis trained on a training set and the actual outputs of this training set.

The squared error is suitable for the following reasons:

- It is the standard error for regression problems. Thus, it is used by most regression algorithms for training, which makes it easier to compare their performances.
- It is symmetric and intuitive to interpret.
- It enables closed-form solutions for linear- and ridge regression and thus speeds up processing time.

Even though, the quality values in the dataset are all discrete integers, the use of the squared error with the chosen algorithms will lead, if not rounded later on, to real-valued outputs with arbitrary precision. However, the real-valued outputs can be seen as another advantage, as they allow finer ratings, that are intuitive to interpret. The finer ratings add information and allow better differentiation between future test samples, that are otherwise expected to be mainly given the quality values of 5, 6 and 7. Thus, it was decided to allow real-valued quality ratings and therefore not to round the predicted outputs.

4. Splitting into Training and Test Set

In order to find the final hypothesis, several hypotheses-sets and learning algorithms were considered. To optimise the final hypothesis, while trying to avoid overfitting and being able to test for generalisation, it was important to split the combined wine data set into a training and a test set. This is done by randomly selecting 20% of the samples in the combined wine dataset and putting them into a separate table. These 1299 samples could be later used to calculate the empirical test error for a given hypothesis. As the empirical test error is an unbiased estimator of the true test error, it gives a good indication of how well the predictor function generalises, assuming there is no bias in the chosen dataset. Using 80% of the dataset for training and 20% for testing is an often-used heuristic in machine learning and seems a reasonable choice for the relatively large-sized total dataset with 6497 samples.

We also stored the outputs in different vectors from the input feature matrices to make it easier to pass them to functions later on. As good practise, we also normalised the training set input features and stored them in a new matrix. The corresponding feature means and standard deviations were also saved and used to normalise the test set features. Moreover, we created a vector with the centered training outputs and stored the corresponding mean of the wine qualities.

5. The Constant Base Predictor

A constant baseline predictor is a very simple predictor, that is used in order to get a first idea of the error magnitude, we are faced with in this problem, and to have a benchmark for comparisons with more complex hypotheses later on. One of the simplest statistical predictors is the mean of the training outputs (i.e. wine qualities), which is therefore chosen as our constant base predictor. The training error, which is now a biased estimator for the population variance of the outputs, the 10-fold-cross-validation error, which is calculated for comparison with other cross-validation errors later on, and the test error are shown in Table 1:

Method	Constant Base Predictor - Mean of training qualities: 5.8192
Training error	0.7611
Cross-validation error	0.7615
Test error	0.7680

Table 1 Constant Base Predictor

Note that the cross-validation is a good estimate of the test error.

6. Linear Regression without transformation

6.1. Without Regularisation

The linear regression algorithm offers a closed-form solution of the form $w = (X^T X)^{-1} X^T y = X^+ y$, where X is the input feature matrix and y is the output vector. The function tries to find the coefficients w for a multivariable linear equation, that minimises the squared error.

To find a linear baseline, all twelve normalised attributes were initially used as features without any transformation. There exists a rule of thumb that states that the number of samples should be at least ten times bigger then the number of variables computed via linear regression. As the considered model will have one coefficient for each feature and another constant coefficient, at least 130 data points are needed. The 5198 samples used for training therefore seem more than sufficient.

The 10-fold-cross-validation error using linear regression (Table 2) shows some improvement to the previous constant base predictor. This indicates that there exists some overall linear correlation between the input features and the wine quality. Further improvement might be yield through regularisation.

6.2. Using Regularisation

In the following, regularisation with the three different penalties ridge regression (A1.1), lasso (A2.2) and elastic net (A2.3) is tested. Due to the large number of samples compared to the moderate number of twelve input features only minor improvements due to regularisation are expected. However, certain features might be found discardable by the feature selection property of lasso and elastic net. For example, in comparison to the other feature coefficients linear regression assigned a very small coefficient to citric acid. This indicates that this attribute has a very small effect on the quality rating. This is underlined by the fact that the matlab *fitlm* function, used to calculate the linear regression, associates a high p-value of 0.673 for the F-statistic to it. The higher the p-value the more likely the feature is insignificant and has a coefficient of zero. [3] Thus, it might not be selected by elastic net or lasso.

Table 2 shows the results obtained using the three different methods. 10-fold-cross-validation was used to find the minimum cross-validation error and the corresponding optimal λ and α. Afterwards, the corresponding model was trained over the whole dataset with its optimal parameter settings and the test error was evaluated on the previously selected test set. For hypotheses selection, it is important to only look at the cross-validation error. The test error should be only used to check the generalisation capability of the chosen hypothesis. Ridge regression, lasso and elastic net do not show any significant improvement to plain linear regression. Furthermore, Figures A2.1 and A2.2 (see App.) indicate that the CV error increases as the strength of the penalty term, which is controlled by λ, rises. Below a certain λ -threshold the CV error seems to stay constant for lasso and ridge regression. Thus, the best parameters shown in Table 2 are expected to vary with changing CV splits in the region, where the CV error is almost constant. A similar conclusion can be drawn for the elastic net. The best value of α almost completely recovers ridge regression (due to α

being almost zero) and a small λ is suggested. Contrary to our initial expectation, no coefficient was found to be zero after applying all three penalties. In summary, the findings suggest that the regularisation performs best if the penalty is set to be as small as possible, therefore reducing the regularising effect until it is almost redundant.

Method	Linear Reg. (no regular.)	Ridge	Lasso	Elastic net
CV error	0.5442	0.5457	0.5449	0.5435
Test error	0.5211	0.5216	0.5216	0.5215
Best parameter(s):	n.a.	$\lambda = 30.5$	$\lambda = 7.4 * 10^{-4}$	$\lambda = 0.0046$, $\alpha = 4.36 * 10^{-4}$

Table 2 CV and test error for linear regression with different penalties

The linear regression and ridge regression were implemented using the *fitlm* and the *ridge* function respectively. The cross-validation was carried out with *crossval.* The *lasso* function was adopted for cross-validation and calculation of elastic net and lasso.

7. Linear Regression using Feature Transform

A potential way to further improve prediction and minimise the test error is to add additional features. One idea is to use the product of two individual features as a new feature, that is representing the interaction between two attributes. Potentially, this could be done for all possible feature combinations. In the further report, those features will be called *interaction features*. In addition, quadratic features could be used. Looking at the p-values of the *fitlm* function, which can be set to use the models "*quadratic"* and "*interactions",* to get a first understanding of the significance of those features for linear regression, it is found that the quadratic features almost all have a very high p-value indicating that they do not add much improvement in finding a better linear model. While a quite big proportion of the interaction features have high p-values as well, there are quite a few with very small ones. For this reason and the fact that it seems intuitive that the interplay of certain physicochemical attributes has a big influence on the human taste, it was decided to add the interaction features to the data matrices.

Following a similar approach as before, the datasets get normalised and stored in separate matrices. If the original and all possible interaction features are used, the model has 78 input variables. This might seem like a feature explosion, that could lead to overfitting, but referring to the previous rule of thumb, there are still enough samples for training. Moreover, the effects of overfitting can be minimised if correct regularisation is applied.

Table 3 shows the results for linear regression, ridge regression, lasso and elastic net. Although there are many more features, the CV error for the linear regression model increased in comparison to the linear regression model with the original features. There is also a bigger difference between the test error and the CV error. This indicates, that the unregularised linear regression tends to overfit the data, due to the large number of features.

Ridge regression tends to keep coefficients small, which helps in avoiding extreme overfitting. Lasso's feature selection property might discard unneeded features, while elastic net tries to utilise both benefits and applies a mixture of the ridge regression and lasso.

Indeed, the cross-validation error of the ridge regression is significantly smaller than the CV-errors of previous models. The norm of the resulting coefficient vector decreased from 142.286 for the linear regression model to 1.084 for the ridge regression model, showing the postitive regularising effect. Moreover, Figure 4 shows how the CV-MSE-error varies with changing λ. As the function first decreases and then increases, there is a clear minimum, which shows that the interesting region of λ is covered and an optimal λ can be selected.

Similar results can be found for lasso. While the graph does show a minimum, it is not as distinct as for the ridge regression case (Fig. A2.3). Nevertheless, the feature selecting regularisation of lasso caused 29 of the 78 coefficients to be zero for the selected hypothesis, and the CV error is slightly smaller than for the ridge case.

Finally, the elastic net penalty was applied. After the cross-validation procedure over several (λ, α)-pairs, the optimal α of 0.5 indicates that the optimal regularisation applies the lasso and ridge penalty with similar strength. The CV error is smaller than all previous cross-validation errors, indicating the overall strength of the elastic net penalty. 24 features were set to have a coefficient of 0 and the norm of the coefficient matrix is 1.575, underpinning the combined regularising effect of elastic net.

Method	Lin. Reg. (no regular.)	Ridge	Lasso	Elastic net
Cross-validation error	0.6254	0.5198	0.5191	0.5169
Test error	0.4886	0.4927	0.4923	0.4922
Best parameter(s):	n.a.	$\lambda = 30.52$	$\lambda = 5.3 * 10^{-4}$	$\lambda = 8.66 * 10^{-4}$, $\alpha = 0.5$

Table 3 CV and test error for linear regression with interaction features and different penalties

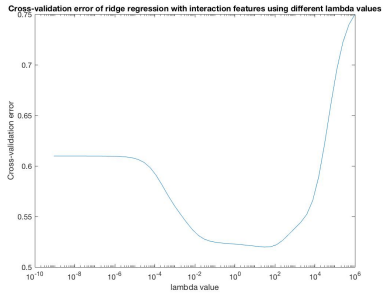

Figure 2 CV-error of ridge regression with interactive features

8. Support Vector Machine Regression

Support Vector Machine is a more advanced machine learning algorithm. It was proposed in lectures for classification. However, there exists a version for regression as well. The SVM regression algorithm uses the

following loss function, called ε-insensitive loss function (notation follows the lectures) [4]:

$$L(y, h(\boldsymbol{x}, w)) = \begin{cases} 0, & if\ |y - h(\boldsymbol{x}, w)| \leq \varepsilon \\ |y - h(\boldsymbol{x}, w)| - \varepsilon, & otherwise \end{cases}$$

The empirical risk is the mean of the loss function over all samples in the training set. SVM regression tries to find a fitting linear function in the high-dimensional space resulting from the Kernel transform, that minimises the empirical risk, while also minimising $\|w\|^2$(A1.4 for math. formulation).

Matlab implements SVM regression with the *fitrsvm* function. It allows the use of a linear $K_{lin}(x, x') = \boldsymbol{x}\,\boldsymbol{x}'$. and the gaussian RBF kernel $K_{gauss}(x, x') = \exp(-\gamma\,\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$. Both were tested using the original twelve features. The gaussian kernel performed significantly better (Table 4) and was therefore used for further optimisation. *fitrsvm* allows to optimise for C, γ and ε, instead of applying the *fitrsvm* default parameters [5]. This optimisation, which applies 5-fold-cross-validation, was done for the SVM regression gaussian kernel, but did not lead to a smaller CV error. This indicates that the default values are close to optimal. Finally, the SVM with gaussian kernel was trained over the whole training set with optimised and default parameters and the test error was evaluated. SVM regression with gaussian kernel led to a lower CV and test error in comparison to the linear regression models.

Kernel for SVM regression	linear	Gaussian (default)	Gaussian (optimised)
Cross-validation error	0.5493	0.4676	0.4691
Test error	-	0.4177	0.4428

Table 4 Cross-validation and test error for different SVM regression methods

9. Neural Networks

Another attempt was to use Neural Networks to solve the regression problem. Matlab allows the implementation of neural networks with an easy to use GUI tool (called by *nnstart*). The neural network was trained in fitting mode with the training set of the original twelve features as inputs. The toolbox allows different settings for the number of neurons in the hidden layer. Several numbers were tested and the best cross-validation error was found with 100 neurons in the hidden layer. The training error was not significantly deviating from the CV error with these settings, indicating that no overfitting took place (Fig. A2.4). This also fits the rule of thumb for the maximum number of hidden neurons (see A1.5). The CV error and the final test error can be found in Table 5.

10. Bag of Trees

Supervised ensemble learning is another concept, which was briefly introduced during lectures. A possible matlab function that implements supervised ensemble learning is *Treebagger*. The function is set to use bootstrap aggregation to combine several regression decision trees (A1.6). [7] This reduces overfitting and improves generalisation. [8] Moreover, for each decision split only one third of the features are considered to further decorrelate the different trees as in random forest. 10-fold-cross-validation was used for the optimal number of trees. 100 was found to work best for this problem. This setting gave a smaller CV-error than the CV error of 0.4278 for the model with 10 trees.

The use of Bag of trees was also encouraged due to the good results of Malloy [9], who worked on the same dataset. Indeed, bag of trees with 100 trees, the 12 original normalised input features and default *Treebagger* settings showed the best performance of all algorithms considered in this report and is therefore the preferred choice.

Method	Neural Network	Bag of Trees
Cross-validation error	0.4382	0.4042
Test error	0.4608	0.3647

Table 5 Cross-Validation and Test Error for Neural Network and Bag of Trees

11. Conclusion

The performances of the considered algorithms gradually improved over the course of the report. The hypothesis, which was found to work best for the given dataset, can be found by using bag of trees. For the final bagged trees model, there was an improvement of 46.92% in cross-validation error and 52.51% in test error to the constant base predictor. This is followed by the neural network model and the SVM regression model with gaussian kernel. While linear regression with different penalties and the use of feature transformation led to a significant performance boost compared to the constant base predictor, it was not as successful as the more advanced algorithms introduced later on. This shows that the underlying function for this problem tends to be more complex than a multivariable linear function with or without interaction features. Interestingly, the test error always tended to be smaller than the CV error. The only exception was for the neural network, where the test error was slightly bigger. The trend of lower test error can probably be explained due to the random split of the dataset, that assigned only very few of the less frequent higher and smaller wine qualities to the test set. A different random seed could potentially lead to slightly different results. Nevertheless, this underlines the fact, that learning could potentially be more refined if the wine qualities were more widely spread and there were more samples with exceptional wine qualities to train on.

For further research, it would be interesting to use wine samples from different regions and to introduce the price of the wine as additional feature.

In conclusion, the report showed that physicochemical attributes can be used to make an improved prediction on the quality of a wine if the correct hypothesis is used. However, as individual taste varies and thus the quality of a vine will be rated differently by different people, a definite prediction on such a subjective quantity will remain difficult to obtain.

12. Pledge

I, Paul Streli, pledge that this assignment is completely my own work, and that I did not take, borrow or steal work from any other person, and that I did not allow any other person to use, have, borrow or steal portions of my work. I understand that if I violate this honesty pledge, I am subject to disciplinary action pursuant to the appropriate sections of Imperial College London.

References

[1] Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Cortez, P., Cerdeira A., Almeida F., Matos T. and Reis J.. Modeling wine preferences by data mining from physico-chemical properties.
In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

[3] Mathworks. (n.d.). Interpret Linear Regression Results. Available from: https://uk.mathworks.com/help/stats/understanding-linear-regression-outputs.html [Accessed 18th June 2018]

[4] Tripod. (n.d.). Support Vector Machine Regression. [online] Available at: http://kernelsvm.tripod.com [Accessed 17 Mar. 2018].

[5] Mathworks. (n.d.). Fit a support vector machine regression model. Available from: https://uk.mathworks.com/help/stats/fitrsvm.html?s_tid=doc_ta#shared-Epsilon [Accessed 18th June 2018]

[6] Hagan, M., Demuth, H., Beale, M. and De Jesus, O. (n.d.). Neural Network Design. Available at: http://hagan.okstate.edu/NNDesign.pdf#page=469 [Accessed 21 Mar. 2018].

[7] Mathworks. (n.d.). Create bag of decision trees. Available from: https://uk.mathworks.com/help/stats/treebagger.html [Accessed 21st June 2018]

[8] Tat, M. (2017). Seeing the random forest from the decision trees: An explanation of Random Forest. Available at: https://towardsdatascience.com/seeing-the-random-forest-from-the-decision-trees-an-intuitive-explanation-of-random-forest-beaa2d6a0d80 [Accessed 20th June 2018]

[9] Machine Learning with the UCI Wine Quality Dataset. [online] Rstudio-pubs-static.s3.amazonaws.com. Available at: https://rstudio-pubs-static.s3.amazonaws.com/175762_83cf2d7b322c4c63bf9ba2487b79e77e.html [Accessed 20 Mar. 2018].

[10] Wikipedia. (n.d.) Bagging. Available from: https://de.wikipedia.org/wiki/Bagging [Accessed 20th June 2018]

[11] n.a. (2006). Regression Trees. Available at: http://www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf [Accessed 21 Mar. 2018].

APPENDIX

A1. Additional mathematical Theory

A1.1 Formulation of Ridge Regression

Minimize $\widehat{R_n}(\boldsymbol{w}) + \frac{\lambda}{n}\|\boldsymbol{w}\|_2^2$
(see Lecture 3, Slide 35)

A1.2 Formulation of Lasso

Minimize $\widehat{R_n}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_1^1$
(see Lecture 3, Slide 48)

A1.2 Formulation of Elastic net

Minimize $\widehat{R_n}(\boldsymbol{w}) + \frac{\lambda(1-\alpha)}{2n}\|\boldsymbol{w}\|_2^2 + \frac{\lambda\alpha}{n}\|\boldsymbol{w}\|_1^1$
(see Lecture 3, Slide 51)

$\widehat{R_n}(w)$: empirical risk
n: number of samples
$\boldsymbol{w}$: coefficient vector
α, λ: parameters

A1.4 Formulation of SVM Regression

The SVM Regression problem can be formulated as follows, where ξ_i and ξ_i^* are slack variables to determine the deviation of training samples outside the ε-insensitive zone (kernelsvm):

$$\text{min. } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

$$\text{s.t. } \begin{cases} y_i - h(\boldsymbol{x}, w)_i \le \varepsilon + \xi_i^* \\ h(\boldsymbol{x}, w)_i - y_i \le \varepsilon + \xi_i \\ \xi_i, \xi_i^* \ge 0, i = 1, \dots, n \end{cases}$$

The problem can be also transformed into the dual problem, similar to the case seen in lectures:

$$h(x) = \sum_{i=1}^{n_{SV}}(\xi_i - \xi_i^*)K(\boldsymbol{x}, \boldsymbol{x}')$$

A1.5 Rule of thumbs for maximal number of Hidden Neurons [6]

$$N_h = \frac{N_s}{\alpha * (N_i + N_o)}$$

- N_h: Number of hidden neurons
- N_s: Number of training samples
- N_i: Number of input neurons
- N_o: Number of output neurons
- α: Scaling between 2 and 10 (way to define how import good generalisation is)

For the Neural Network considered in the report, we find:

$$N_h = \frac{5198}{4 * 13} = 100$$

A1.6 Regression Trees

Regression trees are created by recursive partitioning (clustering) the input space, until the spaces are small enough to describe them with a simple model. The simplest model, used in classic regression trees, is the sample mean of all the sample outputs inside the subspace. [11] At each decision split, we try to split the space using the variable that reduces the residual sums of squares the most, i.e. minimises the following sum:

$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i - \hat{y}_{R_j})^2$$

R_j: sub-spaces of data
$\hat{y}_{R_j}$: sample mean of the sub-space R_j
There exists a greedy algorithm to solve this problem. To improve generalisation bagging is recommended. [8]

A2. Additional Figures

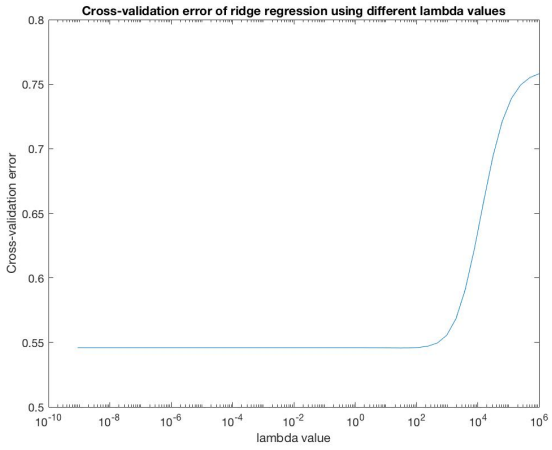

Figure A2.1 CV error of ridge regression

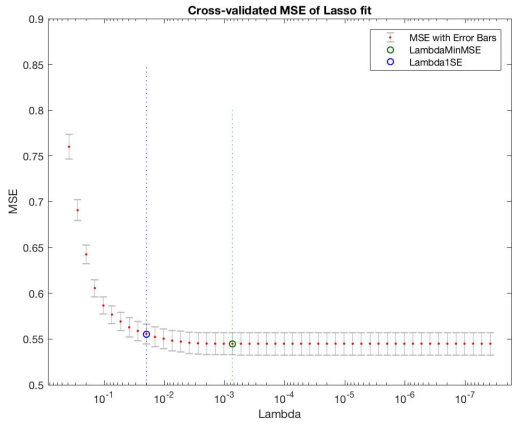

Figure A2.2 CV error of lasso

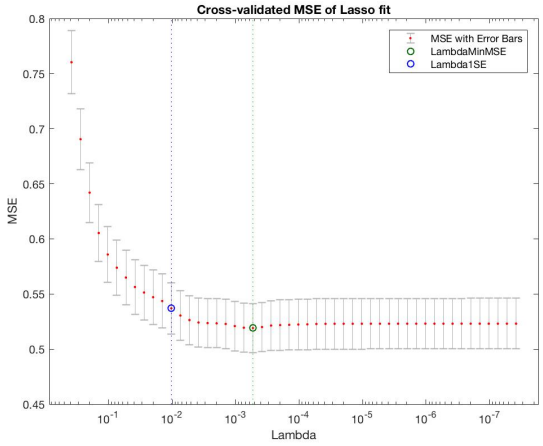

Figure A2.3 CV error of lasso with interaction features

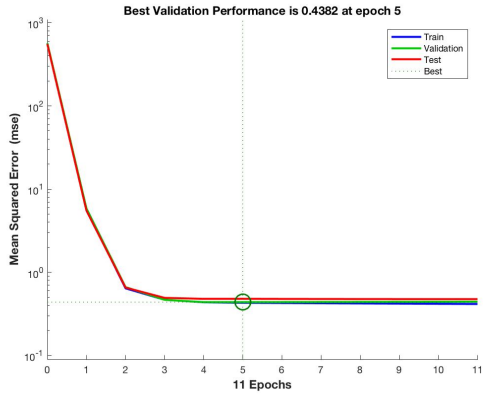

Figure A2.4 Neural Network Training Performance