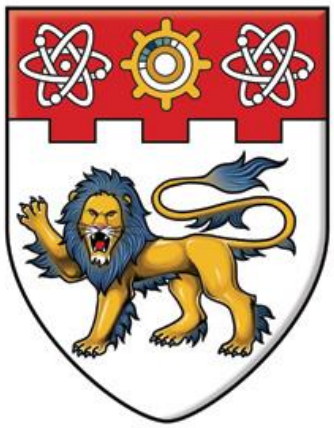


NANYANG TECHNOLOGICAL UNIVERSITY

CZ4042 Neural Networks

Individual Assignment

By:

Name: Bansal Ankur

Matriculation No.: U1322989K

Table of Contents

Objective

The objective of this assignment is to design, learn and experiment with the neural network for spam identification, a classification problem and California housing approximation problem.

Spam Classification

For spam classification problem we are to train a neural network and minimize the error of misclassification for the provided data. The given data is divided into train and test dataset. The training data is provided as P_train dataset and its target output is mapped to T_train dataset. However, the data provided is not processed and must be processed (regularized or normalized) to scale the network inputs i.e. P_train and targets i.e. T_train. The following approach is used to scale the network inputs:

Z Score Normalization:

Z-Score normalization is one of the most commonly used approach using mean and standard deviation. The inputs are normalized such that they have zero mean and unity standard deviation. The **mapstd** function of Matlab's Neural Network Toolbox is used to perform the necessary task.

The mean of the feature in an individual feature vector is calculated using the following formula,

$$Mean = \frac{Sum\ of\ all\ data\ values}{Total\ number\ of\ values}$$

The standard deviation is calculated using the following formula,

$$\sigma = \sqrt{\frac{\sum (x-\mu)^2}{N}}$$

$$\sigma = Standard\ deviation$$

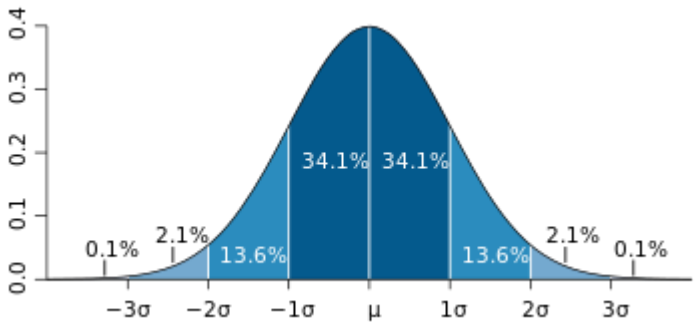
$$\sum = Summation$$

$$x = Value\ in\ the\ dataset$$

$$\mu = mean\ of\ values\ in\ the\ dataset$$

$$N = number\ of\ values\ in\ the\ dataset$$

In the normalized data, each feature value falls in the region represented by the following figure:



1. Vary the Learning Rate

The learning rate was varied and the following values were tested, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 and 1. The settings for the model are as follows:

1. Number of hidden layers = 1
2. Max_fail = 25
3. Maximum number of epochs = 1000
4. Training Data Split = 70:30
5. Training method = traingd (Gradient Descent)

The graphs for training and validation error vs number of epochs are shown below.

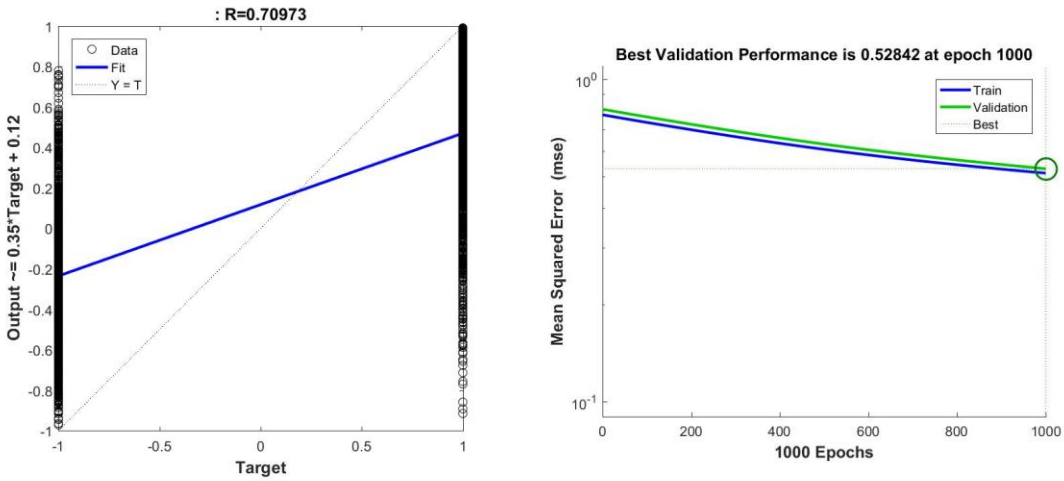


Figure 1: (a) Regression Plot (b) Performance Plot for learning rate 0.005

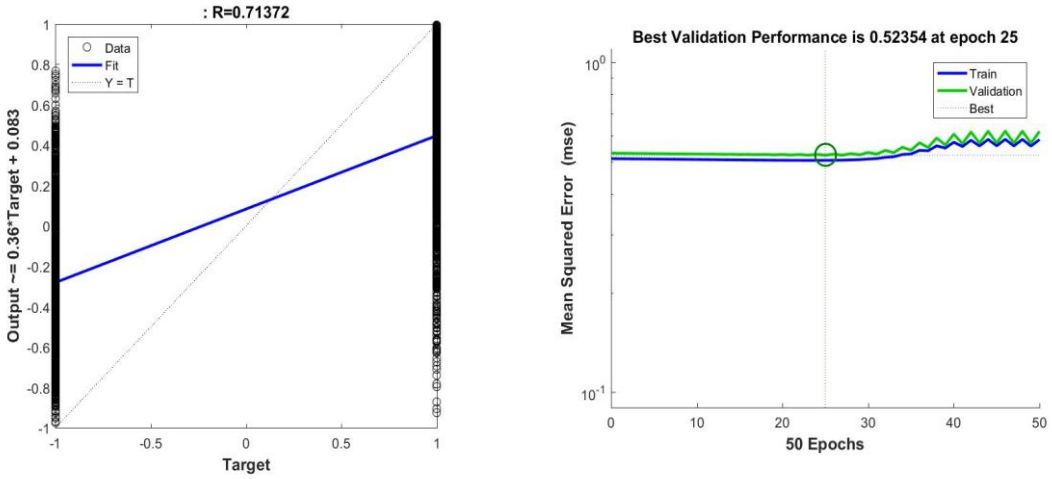


Figure 2: (a) Regression Plot (b) Performance Plot for learning rate 0.01

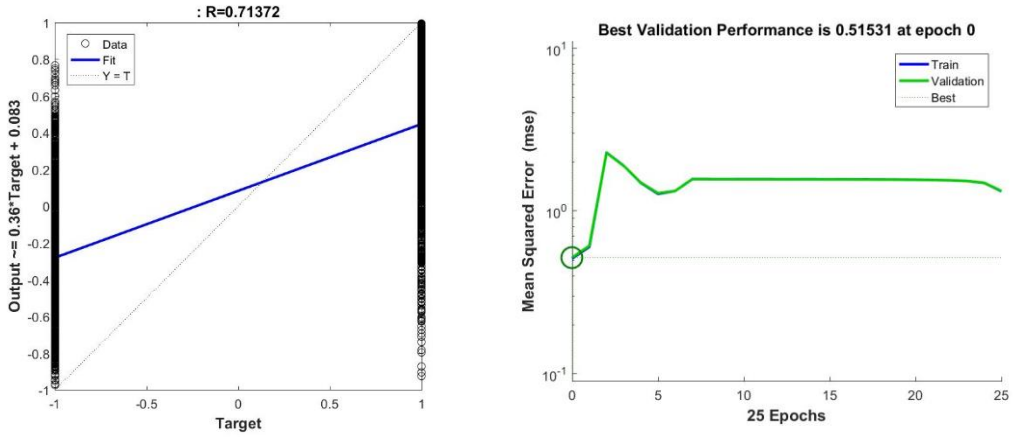


Figure 3: (a) Regression Plot (b) Performance Plot for learning rate 0.05

Analysis

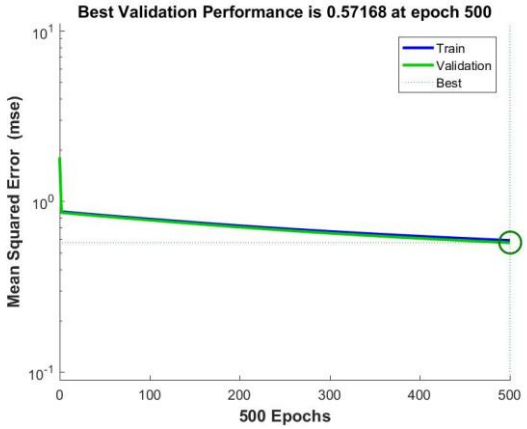
As the learning rate is increased, the training function i.e. gradient descent converges to the local minima of the mean squared error faster within the specified limit for epochs and a set limit for the maximum possible error. With a small value for learning rate, the network does not learn much within the maximum specified epochs as can be observed from Figure 1 (the plot for a learning rate value of 0.005). However, as we increase the learning rate the network starts to learn faster and reaches local minima for the mean squared error within the specified number of epochs (as can be observed from the graph for learning rate value of 0.01). But as we increase it further it tends to overshoot i.e. for a large value of learning rate the function tends to oscillate back and forth between the optimum value of the cost function. Therefore, for the above settings, the optimum value for learning rate should be somewhere around 0.01. Also, the limit for maximum error

acceptable can be reduced to make it run for greater number of epochs in the hope to reduce the error further.

2. Vary the Number of Hidden Layers

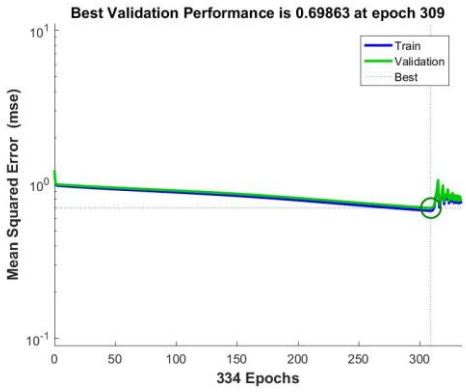
The following settings were used while the number of hidden layers was varied from 1 to 3 with each layers having constant number of neurons (10 in this case).

1. Learning Rate = 0.01
2. Max_fail = 25
3. Maximum number of epochs = 500
4. Training Data Split = 70:30
5. Training method = traingd



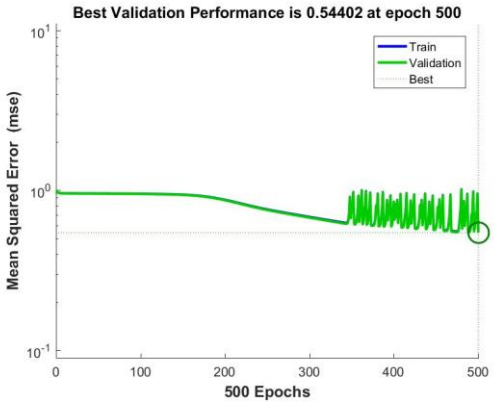
Miss Classification Error
0.5489

Figure 4: (a) Performance Plot (b) Missclassification Error for neural network with one hidden layer



Miss Classification Error
0.5196

Figure 5: (a) Performance Plot (b) Missclassification Error for neural network with two hidden layers



Miss Classification Error
0.4196

Figure 6: (a) Performance Plot (b) Missclassification Error for neural network with three hidden layers

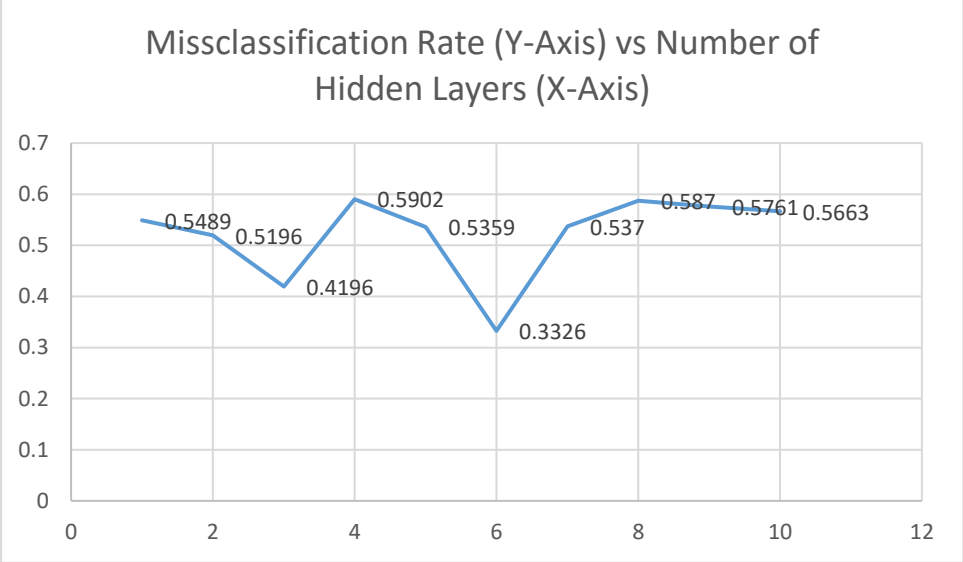


Figure 7: Plot of Missclassification Rate Vs Number of Hidden Layer

Analysis

The above graph represents the variation in miss classification rate with increase in number of hidden layers while the size of each layer is fixed at a constant value of 10. It can be noticed that the accuracy increases by a certain amount i.e. miss classification error reduces for increase in number of hidden layers from 0 to 1 or 1 to 2 (in this case it reduces further when number of hidden layers is increased to 3). However, in general there is no definite pattern for increase or decrease in miss classification rate when we further increase the number of hidden layers, and this can be clearly observed from the plot above. In practice usually one hidden layer is required for a really good accuracy, and too many hidden layers only decrease the effectiveness of the back propagation algorithm. The ***test set error*** will shoot up when you use more Hidden layers though you might have got near perfect accuracy for **train sets**.

3. Vary Size of Hidden Layer (Number of Neurons in a Hidden Layer)

The following settings were used while varying the size of the hidden layer (**only one hidden layer is used**).

1. Number of Hidden Layers = 1
2. Max_fail = 25
3. Maximum number of epochs = 500
4. Training Data Split = 70:30
5. Training method = traingd

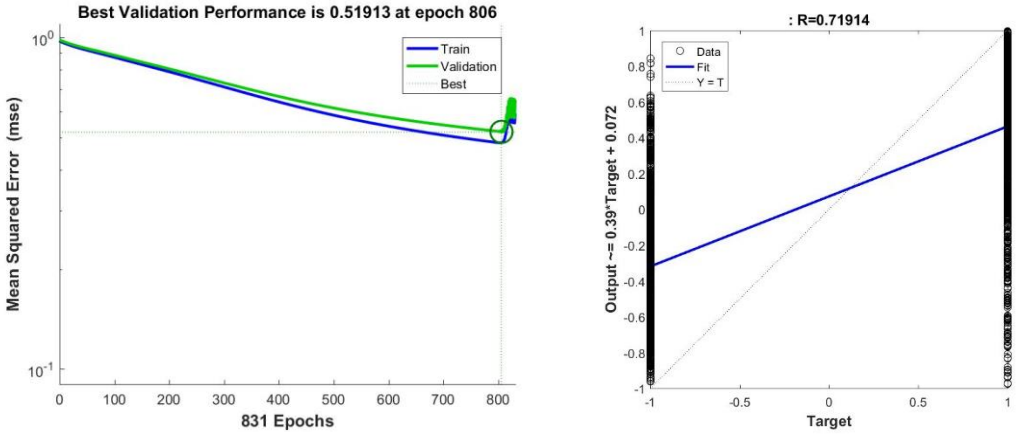


Figure 6: (a) Performance Plot (b) Regression Plot for a neural network with one hidden layer of size 10

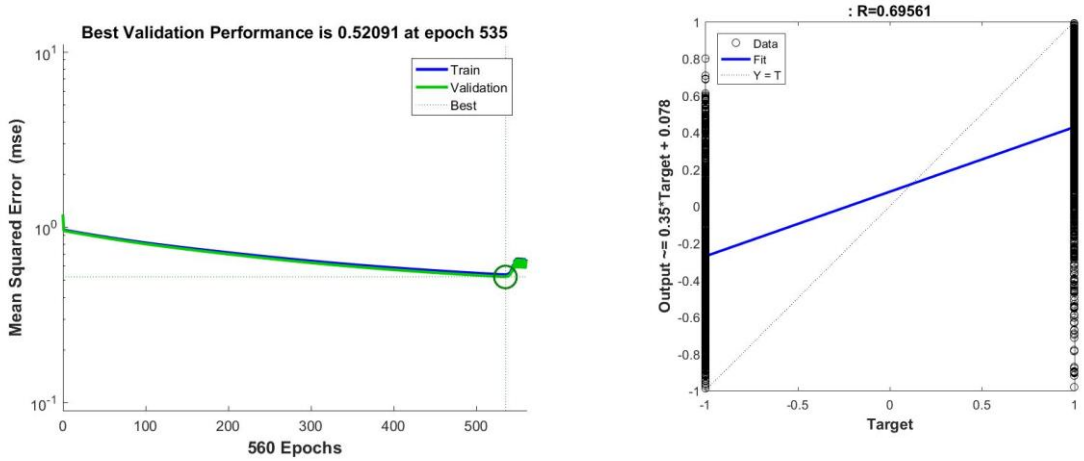


Figure 7:(a) Performance Plot (b) Regression Plot for a neural network with one hidden layer of size 57

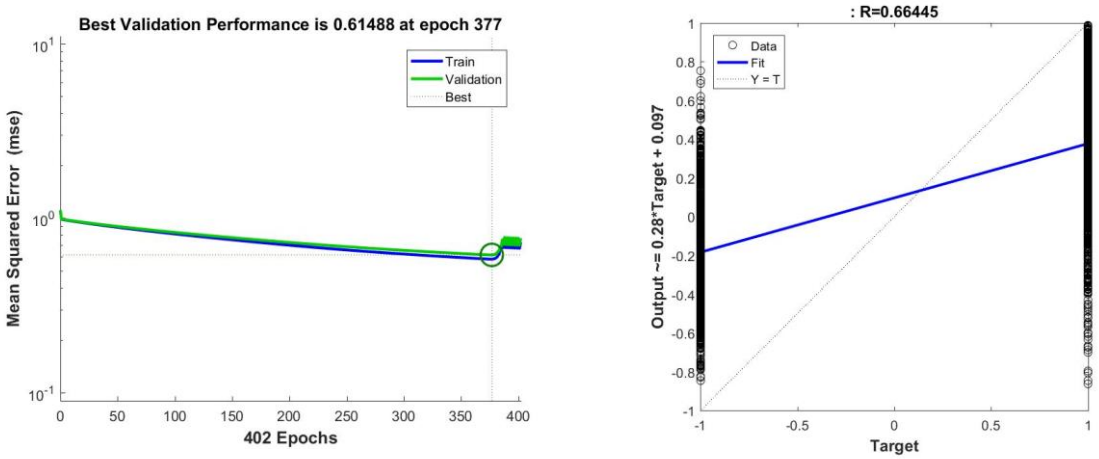


Figure 8: :(a) Performance Plot (b) Regression Plot for a neural network with one hidden layer of size 80

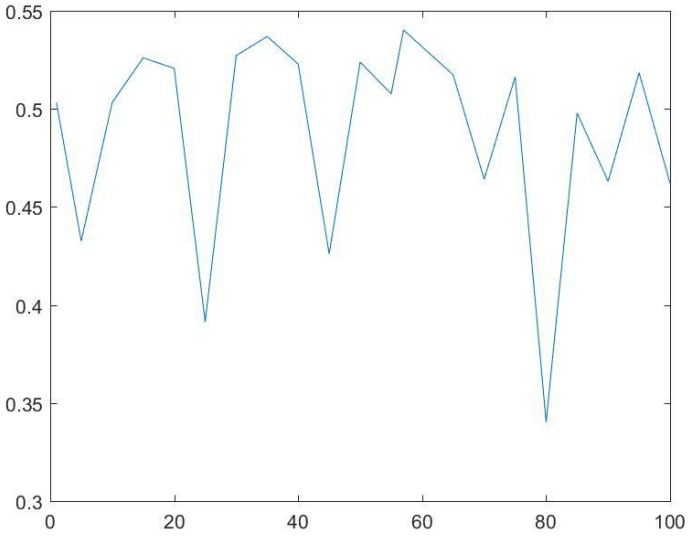


Figure 9: Plot of Missclassification Error Vs Size of Hidden Layer

Analysis

The above graph represents the variation in miss classification rate with increase in size of the hidden layer. It can be noticed that the error rate does not follow any specific pattern with increase in number of neurons in the hidden layer. However, this random variation can be used to apply k fold cross validation for selection of number of neurons. As done during this experiment, we chose the suitable dimensions for our hidden layer and trained the neural network k times (in this case we chose 10 values and thus trained the network 10 times). Finally, we can use the plot in figure 11 to determine the dimension (or number of neurons for the hidden layer) for which the average testing error is minimum.

4. Vary the Epochs

The following settings were used while varying the size of the hidden layer (only one hidden layer is used).

1. Number of Hidden Layers = 1
2. Hidden Layer Size = 10
3. Max_fail = 25
4. Training Data Split = 70:30
5. Training method = traingd

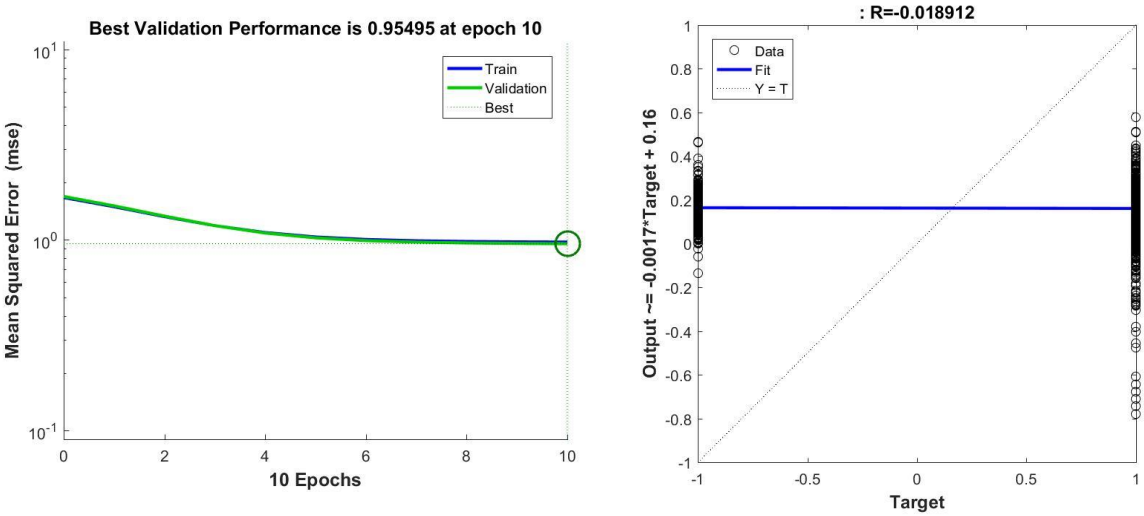


Figure 12: (a) Performance Plot (b) Regression Plot for a neural network with 10 epochs

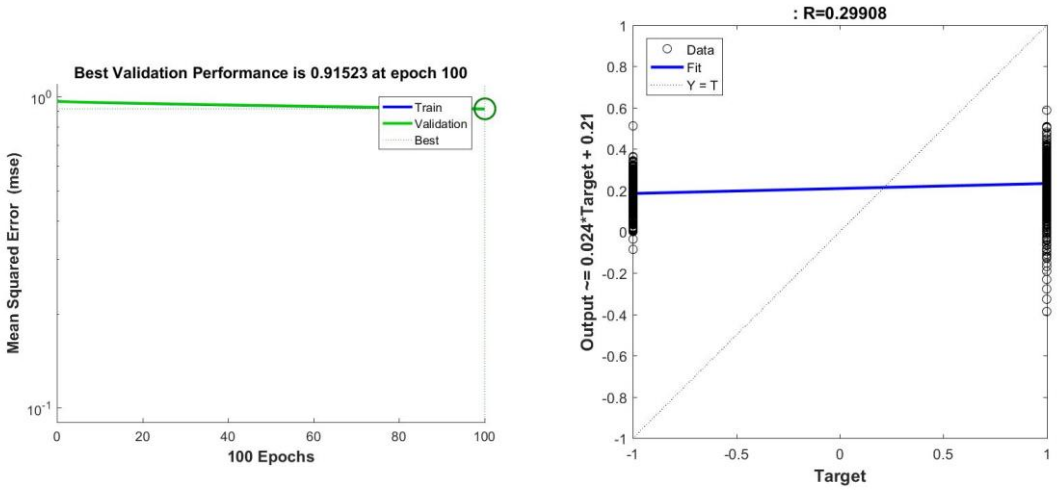


Figure 13: (a) Performance Plot (b) Regression Plot for a neural network with 100 epochs

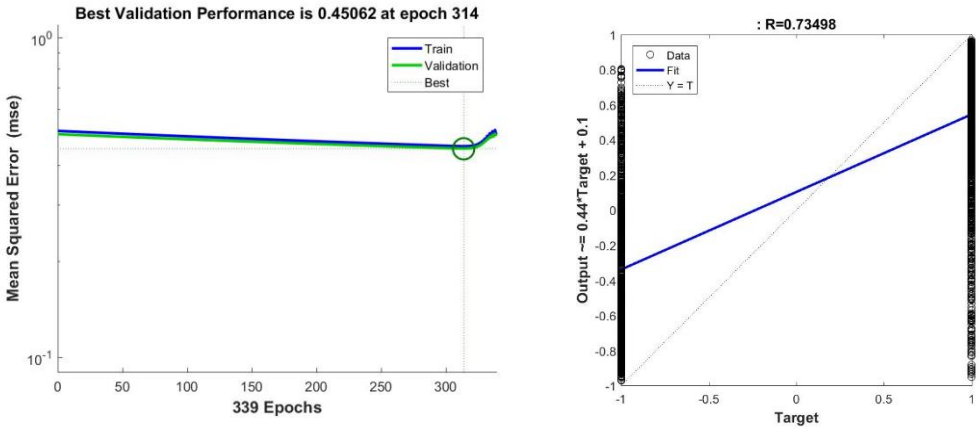


Figure 14: (a) Performance Plot (b) Regression Plot for a neural network with 500 epochs

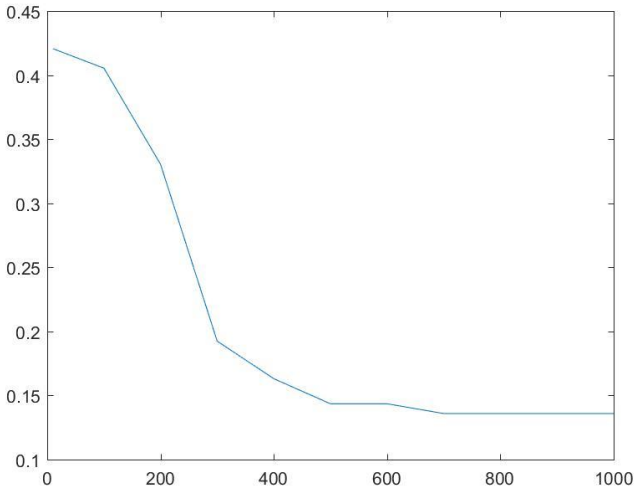


Figure 15: Plot of Missclassification Rate Vs Epochs

Analysis

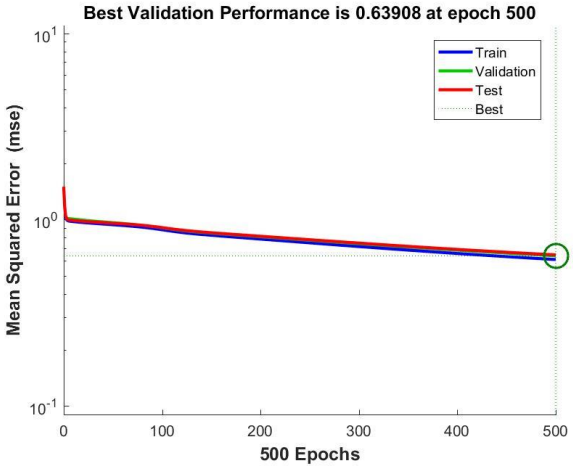
The plot in figure 15 depicts the effect of epochs on miss classification error rate. It can be observed with increase in epochs the mean squared error for the neural network is reducing. To figure out the optimum epochs we need to figure out the epoch at which our validation fails decrease below the maximum desired validation fails. The number of epochs at which our validation fails just surpass the maximum desired validation fails can be fixed as the number of epochs that we use for our neural network. For example, the performance plot in figure 14 for a neural network with 500 epochs generates best validation performance at epoch 339 i.e. the validation fails just surpass the maximum desired validation fails at this epoch number. Therefore, 339 epochs can be an optimum number for the neural network. But however, it can be observed from figure 15 that it is possible to achieve a lower value for miss classification error by further increasing the epoch number which can be tested by training a neural network divided into train,

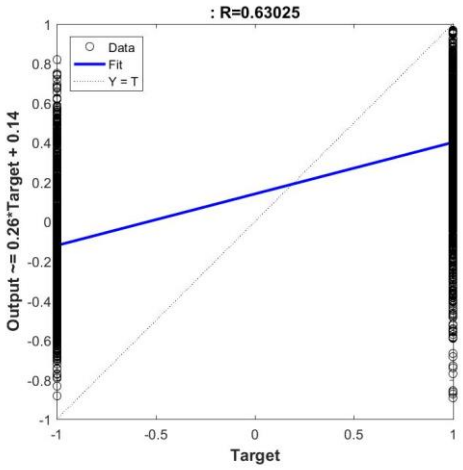
validate and test dataset for epoch number higher than 500. This approach of finding optimum epochs is referred to as **early stopping.**

5. Three Way Split

The following settings were used while varying the size of the hidden layer (only one hidden layer is used).

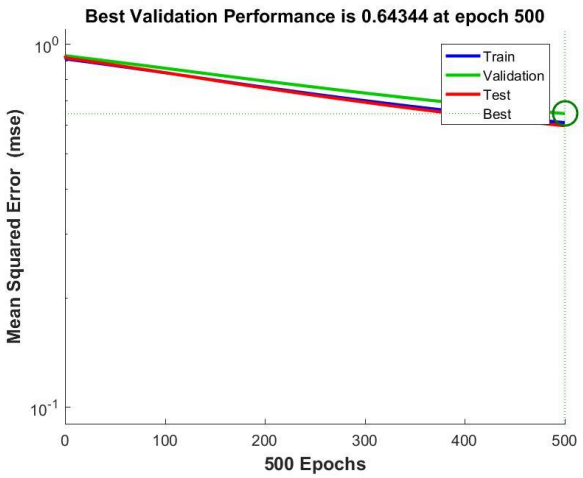
1. Number of Hidden Layers = 1
2. Hidden Layer Size = 10
3. Max_fail = 25
4. Epochs = 500
5. Training method = traingd





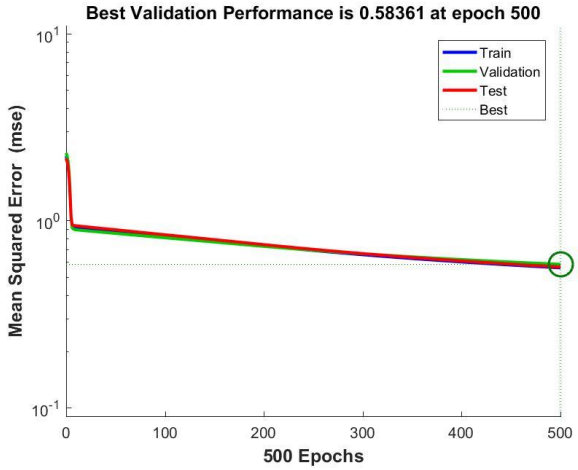
MissClassification Rate
0.1989

Figure 16: (a) Performance Plot (b) Regression Plot (c) Missclassification Rate for a neural network split in a ratio of 70:15:15



MissClassification Rate
0.2152

Figure 17: (a) Performance Plot (b) Missclassification for a neural network split in a ratio of 80:10:10



MissClassification Rate
0.1620

Figure 18: (a) Performance Plot (b) Missclassification for a neural network split in a ratio of 80:0:20

Analysis

In this experiment we vary the data-split ratio between train, validate and test. More data for training essentially means that the model obtained is highly accurate, but it has not been validated. Having minimum validation split ratio is good if the provided dataset is small (in our case it is big enough). Increasing the validation data points may ensure that the model is as accurate as has been properly considered as the best model for data set used, but may be less accurate than the other case because of reduction in data points for training. A less split ratio for test set will simply mean a greater variance in performance statistics of the trained neural network. Therefore, it is essential to find a balance between these sets to obtain the best possible network with least possible miss classification error. From the above three examples of data split 70:15:15 turns out to be the best option because of lower mean squared error. (Even though the error rate is higher than the

80:0:20 split, it is better to choose 70:15:15 because our initial dataset is big enough to have a validation set which can assist in selection of a better ANN architecture).

6. Vary the Max Fail

The following settings were used while varying the size of the hidden layer (only one hidden layer is used).

1. Number of Hidden Layers = 1
2. Hidden Layer Size = 10
3. Epochs = 1000
4. Training Data Split = 60:40
5. Training method = traingd

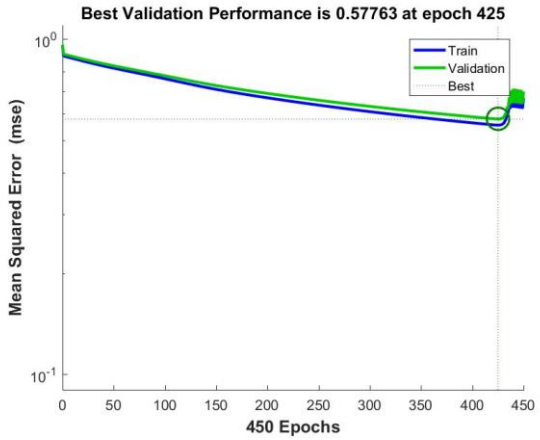


Figure 19: Performance Plot for Max_fail = 25

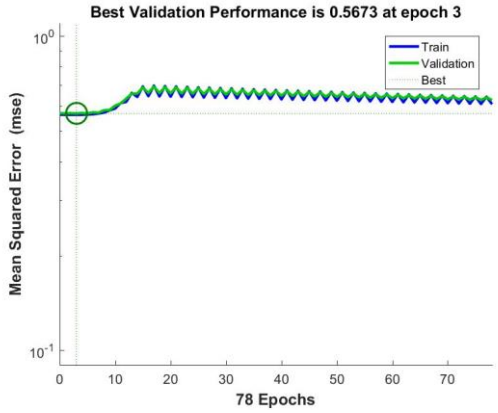


Figure 20: Performance Plot for Max_fail = 75

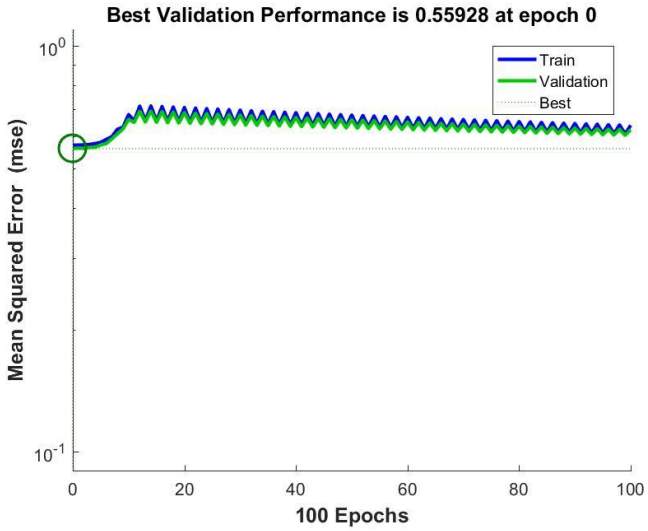


Figure 21: Performance Plot for Max_fail = 100

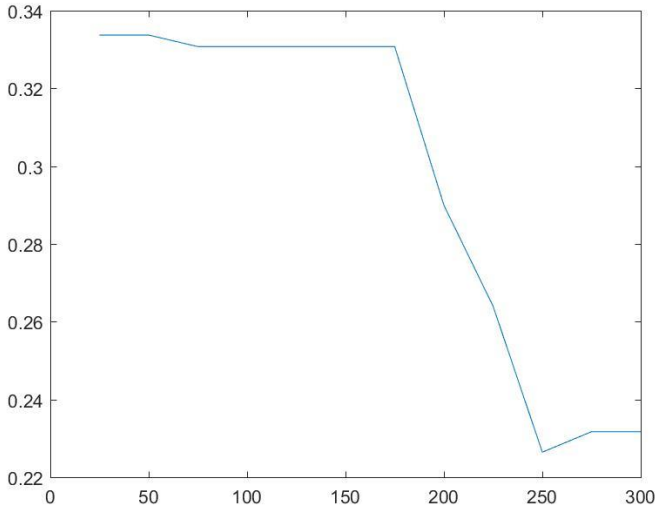


Figure 22: Plot of Missclassification Error Vs Max_fail

Analysis

In this experiment we vary the stopping criterion of neural network training by modifying the value for max_fail i.e. the maximum value for validation fail. From the above plots it can be observed that the increase in value of max_fail ensures that network is properly trained. Thus the time for completing the training of the network increase substantially with increase in value of max_fail, however as the plot in figure 22 suggests it doesn't guarantee improvement in performance or accuracy.

7. Vary Backpropagation function to Bayesian Regularization

The following settings were used while varying the size of the hidden layer (only one hidden layer is used).

1. Number of Hidden Layers = 1
2. Max_fail = 25
3. Hidden Layer Size = 40
4. Epochs = 200
5. Training Data Split = 70:30
6. Training method = trainbr

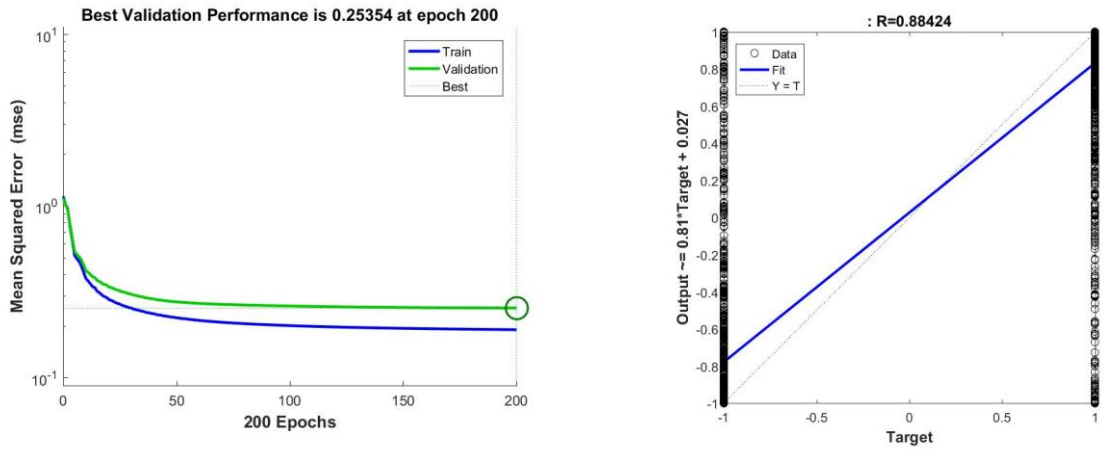


Figure 23: (a) Performance plot (b) Regression Plot for a neural network with size of hidden layer = 40

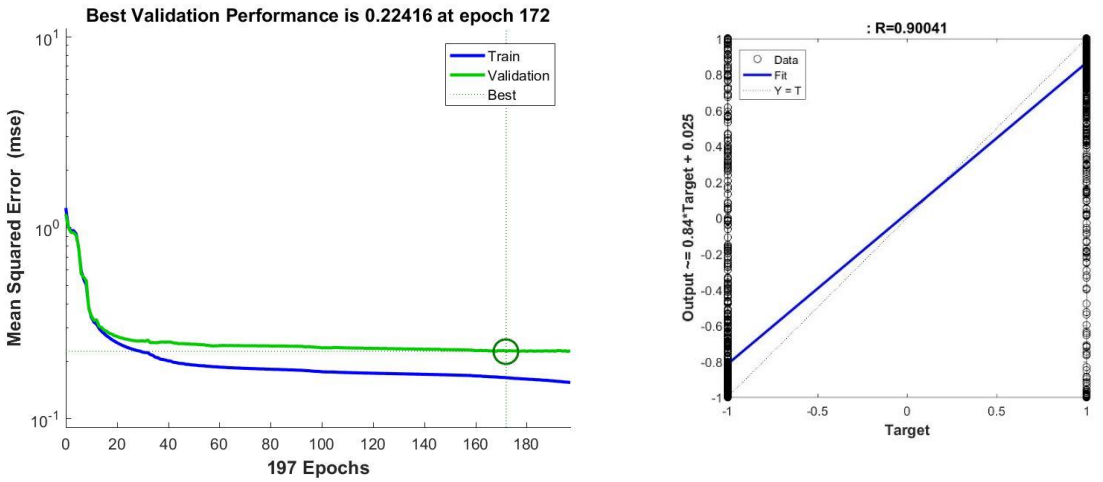


Figure 24: (a) Performance Plot (b) Regression plot for a neural network with hidden layer size = 80

MissClassification Rate (Hidden Layer Size = 40)	MissClassification Rate (Hidden Layer Size = 80)
0.0793	**0.0728**

Analysis

From the above plots and the tables, it can clearly be noted that the value for miss classification error is extremely low for the Bayesian Regression backpropagation technique for this Multi-Layer Perceptron based neural network. From figure 11 in section 3 we can see that in the neural network trained using Lavenberg-Marquardt Algorithm for the same configuration settings the missclassification error for a hidden layer size of 40 and 80 is roughly around 0.45 and 0.32 respectively whereas using Bayesian it is 0.0793 and 0.0728 respectively. Thus, it can be observed that performance of this neural network is around 5 times better than the performance of neural network trained using LMA. In this technique of backpropagation, there is no validation done and the complete data provided in the training data set is used.

8. ANN Architecture Selection

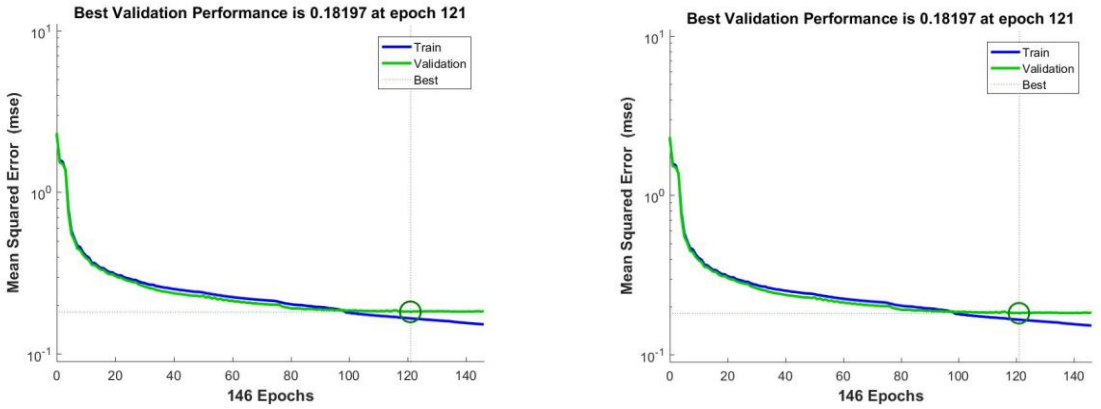


Figure 25: (a) Performance Plot - LMA, Hidden Layer Size = 40 (b) Performance Plot – LMA, Hidden Layer Size = 80

It can be observed from the above plots that the misclassification value for the neural network trained using LMA with a hidden layer of size 80 and maximum epochs 200 is 0.0626 which is even lower than the value got using Bayesian Regularization.

The choice for selection of a ANN architecture has been made on the basis of variation of the neural network used during the training period and based on the testing errors. Of the three different backpropagation methods tried, namely, Lavenberg-Marquardt Algorithm (LMA), Gradient Descent Learning and Bayesian Regularization, LMA outperform the other two.

Extensive testing was performed with a fixed maximum validation check value of 25, hidden layer size of either 40 or 80 which is considerably the optimum based on the explanation provided in section 2 of Spam Classification. Increase in number of validation checks can enhance the performance of the neural network, however, it incurs a substantial time increase which can lead to overfitting of data. The maximum number of epochs is set to 200 while comparing these three backpropagation

algorithms and increasing epochs would also increase the training time. The optimum value for learning rate turns out to be around 0.01 and one hidden layer proves to be sufficient for the neural network.

Approximation Problem

NOTE: Since, root mean square (**RMS**) is ***directly proportional*** to the mean square (**MS**) all the comparisons for approximation has been made using mean square value. The plots present the mean square value instead of root mean square value.

1. Varying Hidden Layer Size

The following settings were used while varying the size of the hidden layer (only one hidden layer is used).

1. Number of Hidden Layers = 1
2. Max_fail = 25
3. Epochs = 200
4. Training Data Split = 70:30
5. Training method = trainlm

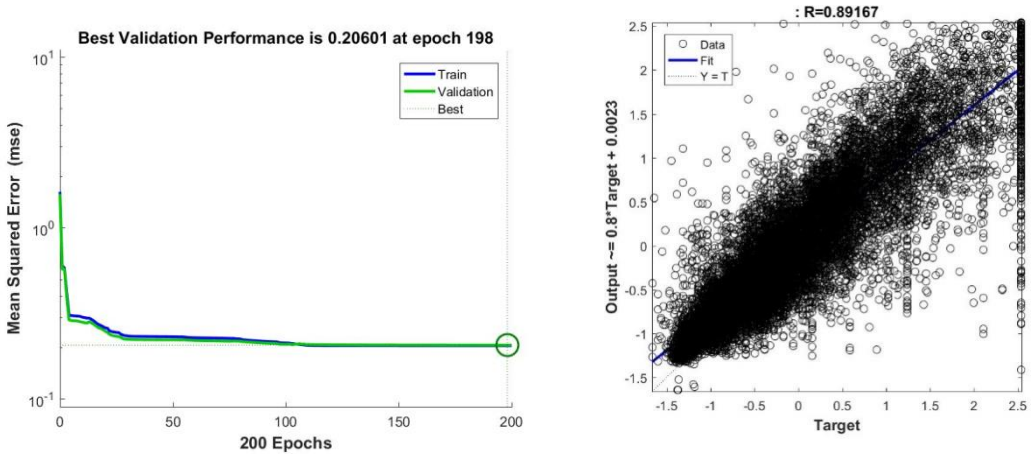


Figure 26: (a) Performance Plot (b) Regression Plot for a neural network with hidden layer size 10

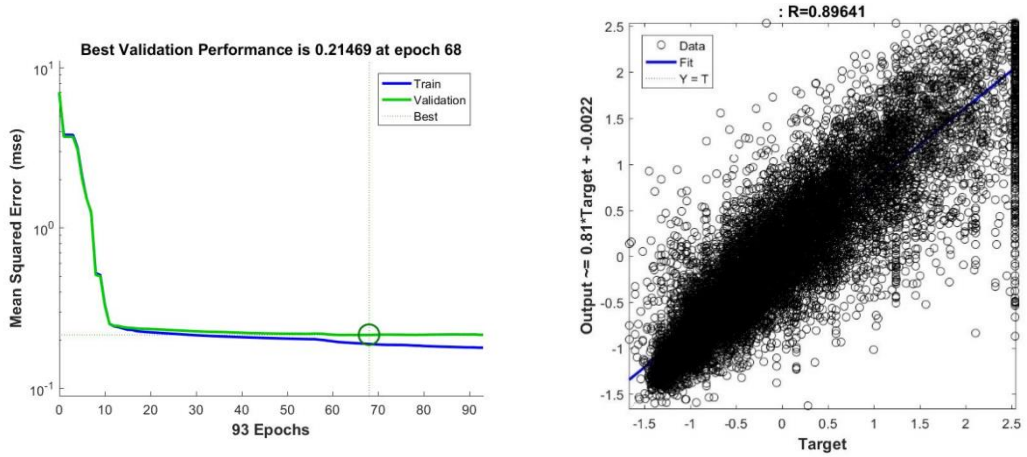


Figure 27: (a) Performance Plot (b) Regression Plot for a neural network with a hidden layer size 50

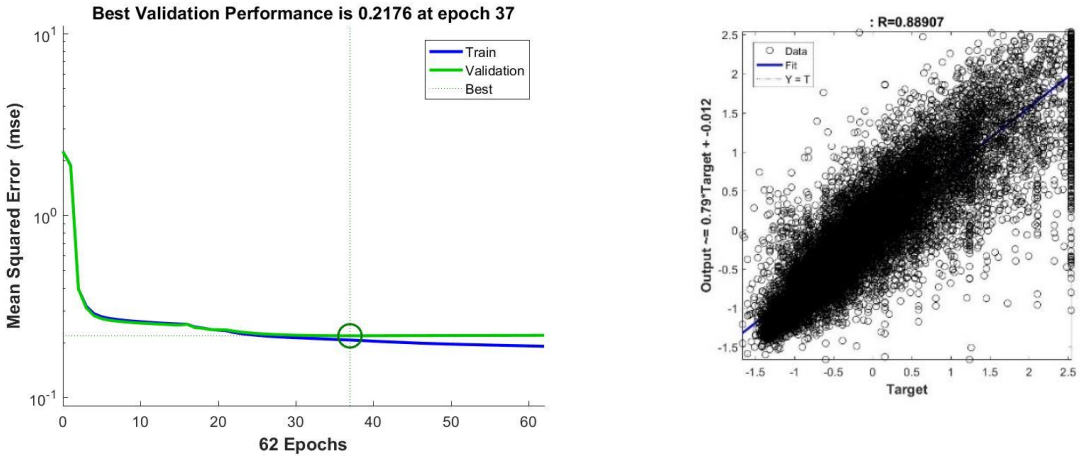


Figure 28: (a) Performance Plot (b) Regression Plot for a neural network with hidden layer size 80

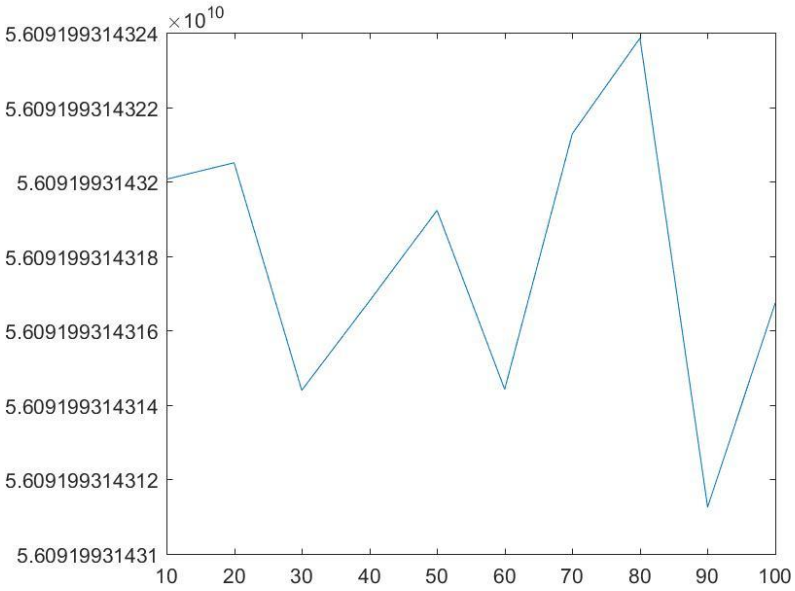


Figure 29: (a) Plot of MS Error Vs Hidden Layer Size

Analysis

The above graph represents the variation in MS error rate with increase in size of the hidden layer. It can be noticed that the error rate does not follow any specific pattern with increase in number of neurons in the hidden layer. However, this random variation can be used to apply k fold cross validation for selection of number of neurons. As done during this experiment, we chose the suitable dimensions for our hidden layer and trained the neural network k times (in this case we chose 10 values and thus trained the network 10 times). Finally, we can use the plot in figure 29 to determine the dimension (or number of neurons for the hidden layer) for which the average testing error is minimum.

2. Different Data Split

The following settings were used while varying the size of the hidden layer (only one hidden layer is used). The data is split into the ratio 60:40 instead of the usual 70:30 in this case.

1. Number of Hidden Layers = 1
2. Max_fail = 25
3. Epochs = 200
4. Training Data Split = 60:40
5. Training method = trainlm

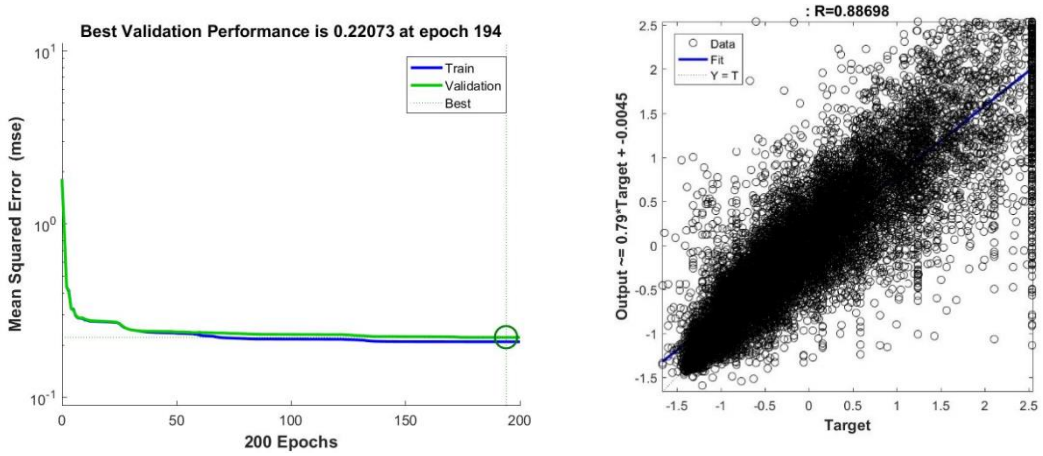


Figure 30: (a) Performance Plot (b) Regression Plot for a neural network with hidden layer size 10

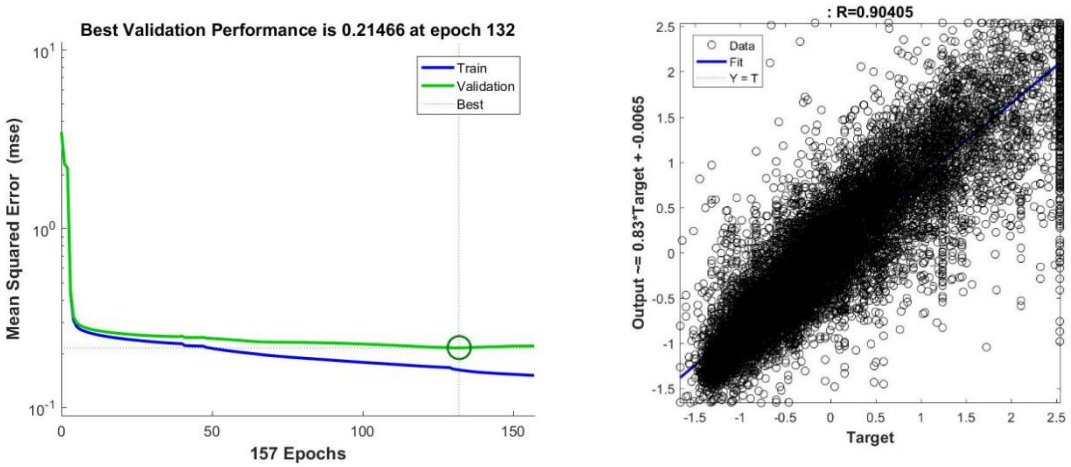


Figure 31: (a) Performance Plot (b) Regression Plot for a neural network with hidden layer size 50

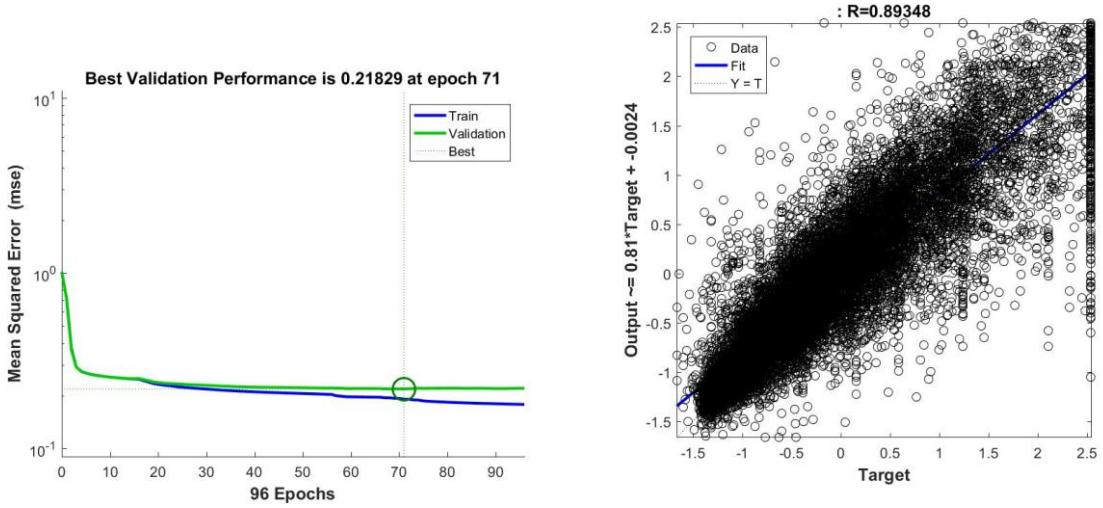


Figure 32: (a) Performance Plot (b) Regression Plot for a neural network with hidden layer size 80

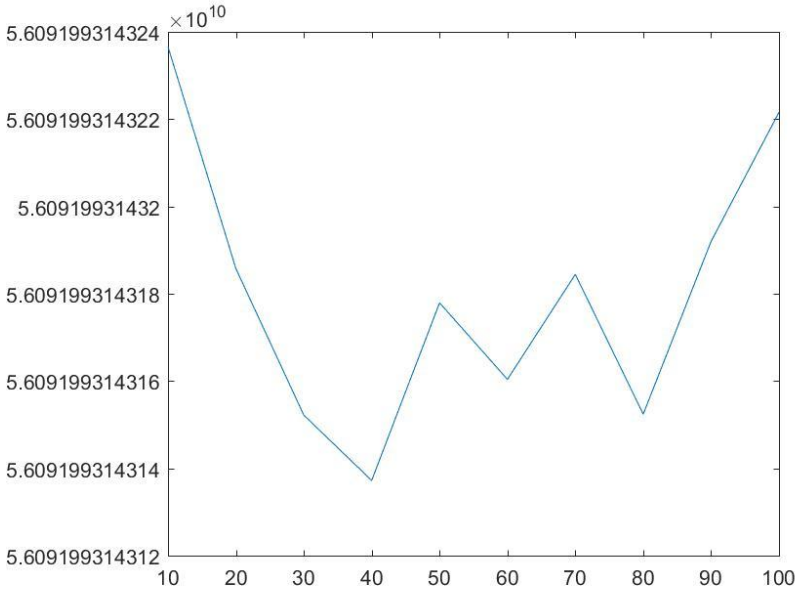


Figure 33: Plot of MS Error Vs Hidden Layer Size

Analysis

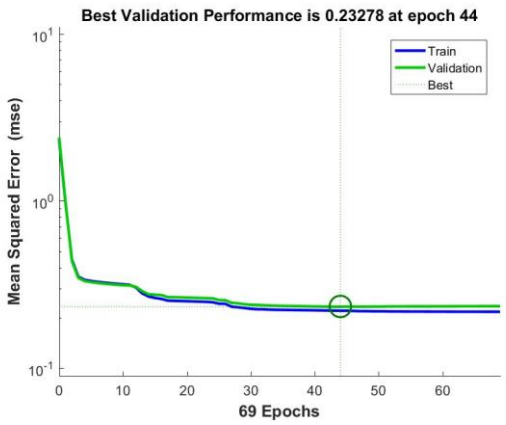
In this type of experiment, changing the data-split ratio, the model is ensured that almost equal number of data is used for both training and validation of the model. More data for training essentially means that the model obtained is more accurate, but the model has not been validated. Increasing the validation data points may ensure that the model is as accurate as has been properly considered as the best model for the data set used, but may be less accurate than the other case because lesser number of data points have been used for training.

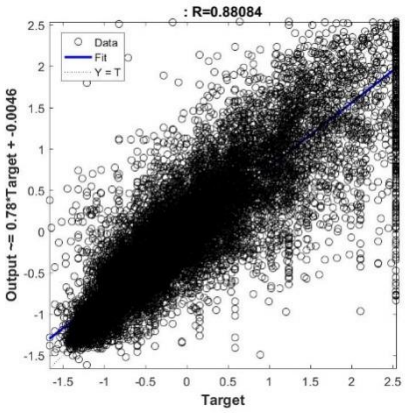
Balancing the ratio between the number of data points used for training and the number of data points used for testing is important to obtain the best model applicable to the current task of classification of the data points into different classes.

3. Varying Max Fail

The following settings were used while varying the max fail (only one hidden layer is used).

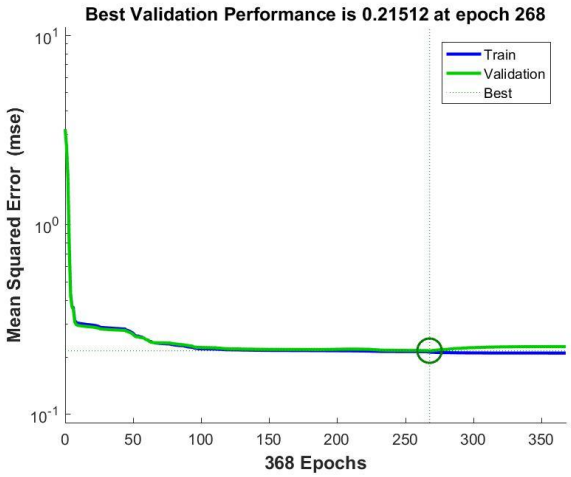
1. Number of Hidden Layers = 1
2. Hidden Layer Size = 10
3. Epochs = 400
4. Training Data Split = 70:30
5. Training method = trainlm

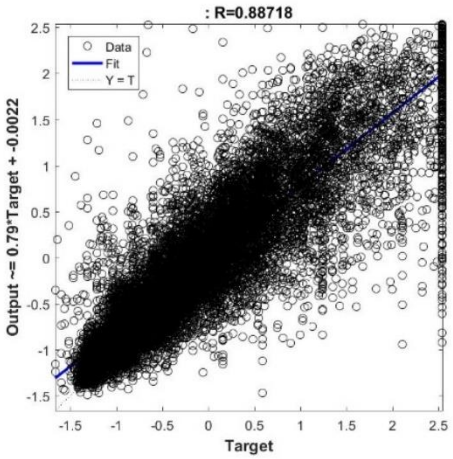




MS Error
56091993143.2196880000

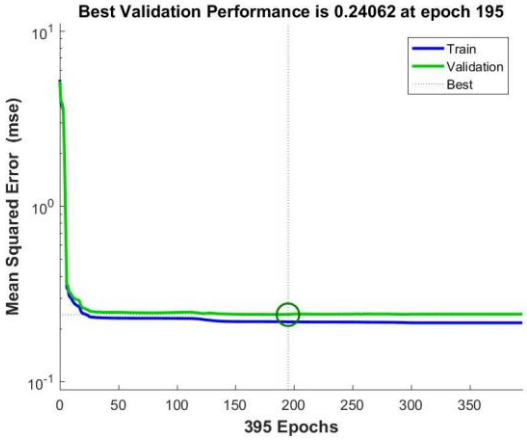
Figure 34: (a) Performance Plot (b) Regression Plot (c) MS Error for a neural network with max_fail = 25

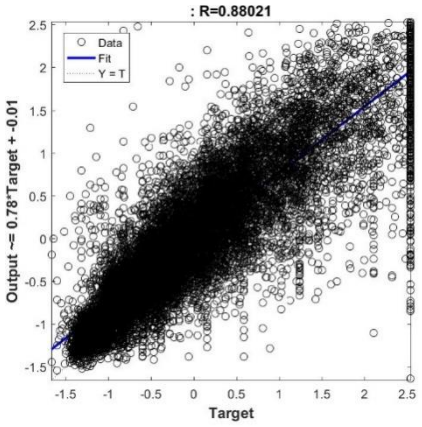




MS Error
56091993143.2284700000

Figure 35: (a) Performance Plot (b) Regression Plot (c) MS Error for a neural network with max_fail = 100





MS Error
56091993143.2480470000

Figure 36: (a) Performance Plot (b) Regression Plot (c) MS Error for a neural network with max_fail = 200

Analysis

In this experiment we vary the stopping criterion of neural network training by modifying the value for max_fail i.e. the maximum value for validation fail. From the above plots it can be observed that the increase in value of max_fail ensures that network is properly trained. Thus the time for completing the training of the network increase substantially with increase in value of max_fail, however it doesn't guarantee improvement in performance or accuracy.

4. Train Function – Gradient Descent and Vary the Learning Rate

The following settings were used while varying the size of the hidden layer (only one hidden layer is used).

1. Max_fail = 25
2. Number of Hidden Layers = 1
3. Hidden Layer Size = [10]
4. Epochs = 200
5. Training Data Split = 70:30
6. Training method = trainlm

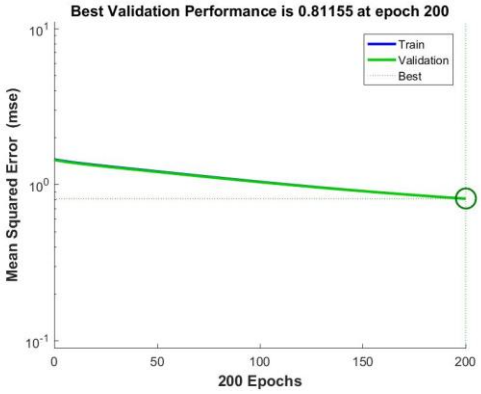


Figure 37: Performance Plot for learning rate 0.001

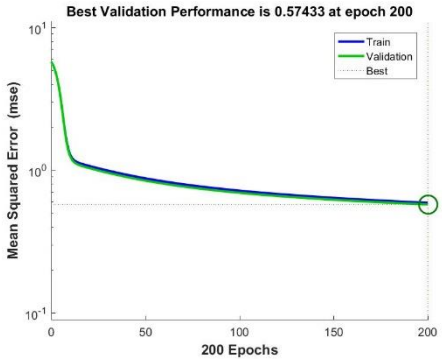


Figure 38: Performance Plot for learning rate 0.003

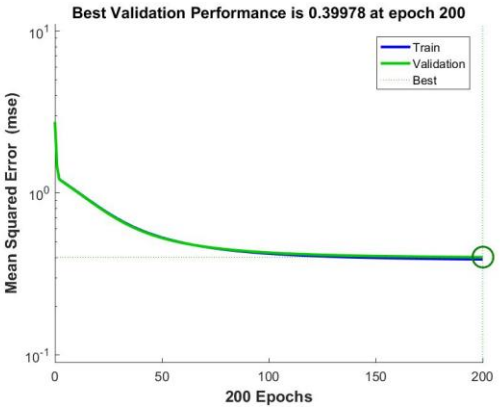


Figure 39: Performance Plot for learning rate 0.01

Figure 40: Performance Plot for learning rate 0.01

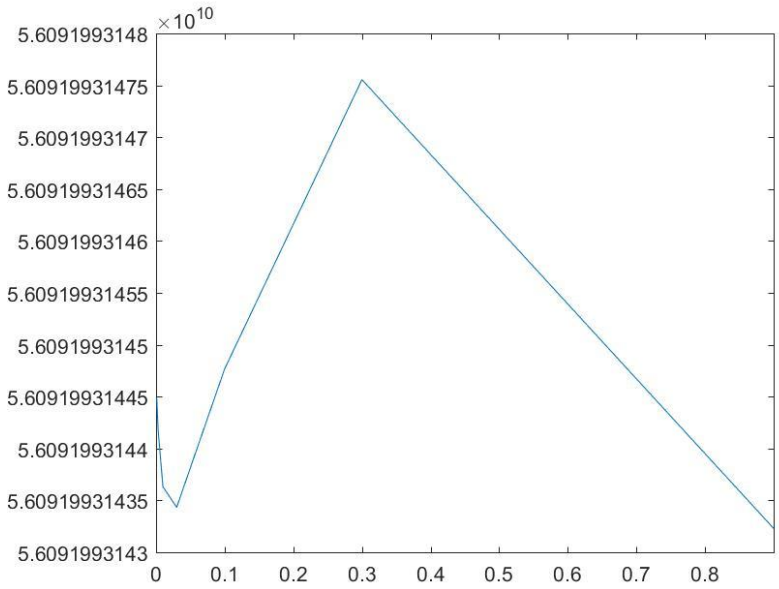


Figure 41: Performance Plot for learning rate 0.003

Analysis

As the learning rate is increased, the training function i.e. gradient descent converges to the local minima of the mean squared error faster within the specified limit for epochs and a set limit for the maximum possible error. With a small value for learning rate, the network does not learn much within the maximum specified epochs as can be observed from Figure 41 (the plot for a learning rate value of 0.005). However, as we increase the learning rate the network starts to learn faster and reaches local minima for the mean squared error within the specified number of epochs (as can be observed from the graph for learning rate value of 0.01). But as we increase it further it tends to overshoot i.e. for a large value of learning rate the function tends to oscillate back and forth between the optimum value of the cost function. Therefore, for the above settings, the optimum value for learning rate should be somewhere around 0.01. Also, the limit for maximum error acceptable can be reduced to make it run for greater number of epochs in the hope to reduce the error further.

5. ANN Architecture Selection

The choice for selection of a ANN architecture has been made on the basis of variation of the neural network used during the training period and based on the testing errors. Of the two different backpropagation methods tried, namely, Lavenberg-Marquardt Algorithm (LMA) and Gradient Descent Learning, LMA outshines.

Testing was performed with a fixed maximum validation check value of 25, and different number and sizes of hidden layer. The maximum validation check was also varied as a stopping criteria for comparisons. Increase in number of validation checks can enhance the

performance of the neural network, however, it incurs a substantial time increase which can lead to overfitting of data. The maximum number of epochs is set to 200 while comparing the backpropagation algorithms and increasing epochs would also increase the training time.

6. Radial Basis Function

The radial basis function can be defined by the following equation,

$$\phi(\boldsymbol{X}) = \phi(||\boldsymbol{X}||)$$

It involves choosing the bias function of the following form,

$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|),$$

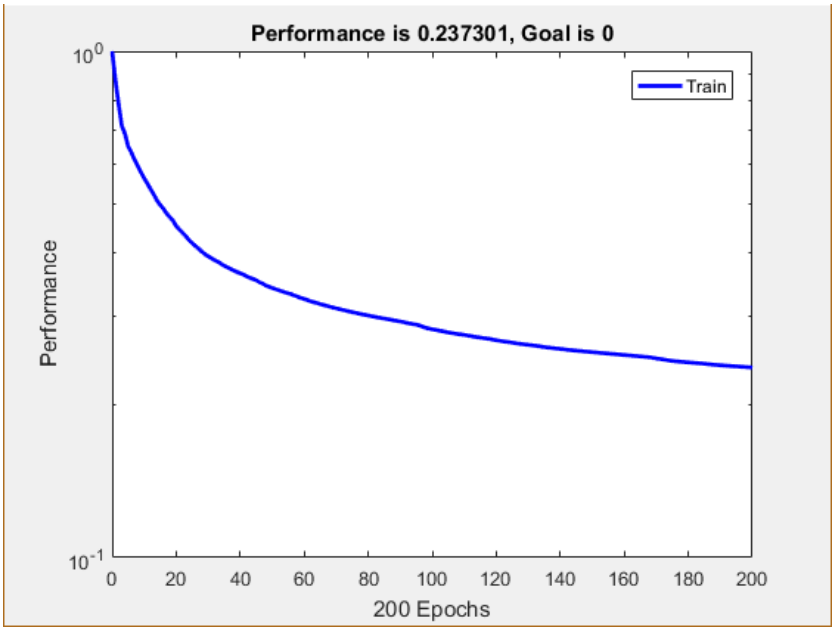
The radial basis function is the linear summation of non-linear inputs which helps in clustering the input results which better helps in the approximation function. In this function approximation format, the strict interpolation approach involves choosing each individual point in the training input as the centre of the cluster. However, the number of input data points L should be much less than the number of input features

$$N<L$$

To find a generalized mapping is considered to be the approximation. The Gaussian basis functions give a suboptimal solution that approximated the multi-dimensional surface.

Training Error

The number of neurons are varied and the corresponding mean square error is noted.



Number of Neurons	Mean Square Error
50	0.999939
100	0.340835
150	0.282458
200	0.254537
250	0.237301
300	0.225163
350	0.208598

In this it can be observed that on increasing the number of neurons in the hidden layer which represent the number of non-linear function that the neural network learns, both the training and testing error are steadily decreasing. This proves that the approximation of the California housing prices better fits the category of this and is much more efficient that using a single linear approximation function as shown in the above method.