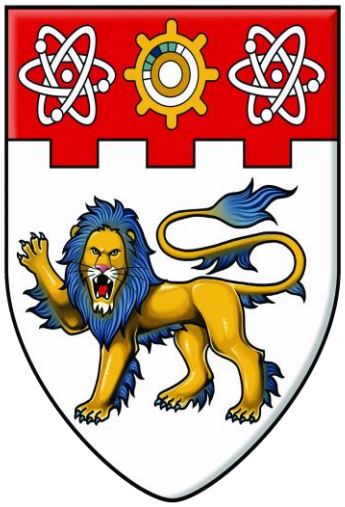


NANYANG TECHNOLOGICAL UNIVERSITY

CZ4042 NEURAL NETWORKS

Assignment

Abstract

This report summarizes the neural networks implemented to solve the problem of classification for SPAM/NOT SPAM emails and the function approximation from the California house pricing dataset to test the new data points in the region and accurately classify the spam emails in the first part and approximate in the second part of the assignment.

Seshadri Madhavan
Matric. No: U1322790J

Table of Contents

Configuration Settings

Training Data

The training data for training the neural network model has been provided in the P_train file of the dataset and its target output has been mapped to the T_train data vector from dataset.

The data provided in the above file is unprocessed and processing needs to be done to convert it into mean normalize the input feature vector. Before feeding the data into the neural network for training and validation the training data is split into two portions for validation and training. The ratio of the data split for each of the individual model has been mentioned in this report.

The neural network model is then trained and the optimum model is decided using the best validation model.

Test Data

The test data for testing the accuracy of the neural network is provided in the P_Test file of test dataset and its target output mapping is provided in the T_Test.

The test data provided is used to check the accuracy of the trained neural network model which is expected to be close to the expected output for the unseen data which has not been used for training.

Data Pre-processing

Training data is to be pre-processed normalizing the data such that the mean of the input feature vector is 0 and the standard deviation of the data is 1. This will ensure that each of the individual feature in the input feature vector falls in this normalised region.
Test data is to be pre-processed using the mean and standard deviation derived after pre-processing the training data. The testing data is not used in the pre-processing stage for the calculation of mean and standard deviation.

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

$$\text{Mean} = \frac{\text{Sum of all data values}}{\text{Number of data values}}$$

Symbolically,

$$\bar{x} = \frac{\sum x}{n}$$

where $\bar{x}$ (read as 'x bar') is the mean of the set of x values,
$\sum x$ is the sum of all the x values, and
n is the number of x values.

The standard deviation of the data is calculated using the following formula,

$$\sigma = \sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

σ = standard deviation

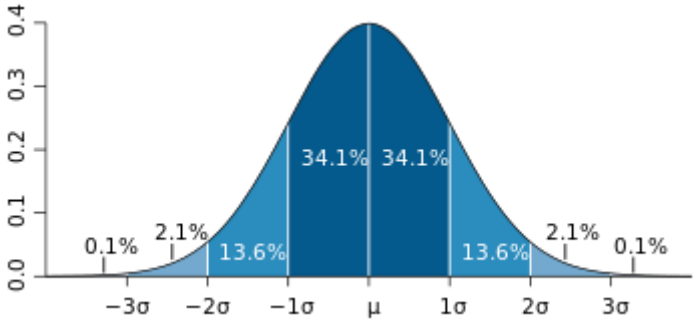
$\sum$ = sum of

x = each value in the data set

$\bar{x}$ = mean of all values in the data set

n = number of value in the data set

After mean normalizing the data, each of the input feature in the feature vector will fall in region which is represented by the following figure,



Performance Metric

For the classification problem, out task is to minimize the error of misclassification of the data.

Training function used

Lavenberg-Marquardt Algorithm

The LMA is an iterative algorithm used for solving the minimization of the non-linear least squares problem which is used for curve fitting in a multi-dimensional data space. The LMA is first initialized with a starting value then converges to the local minimum. As such LMA reaches different convergence points for different values of initialization. As such, only when this initialization point is closer to the Global Minima, we can obtain the best model for the given dataset.

As such for the purposes of training, the trainlm function in MATLAB initializes the guess parameter vector to the following value,

$$\boldsymbol{\beta}^T = (1,1,\ldots,1)$$

For each iteration of the learning, the parameter β is updated using the gradient of the cost function using the approximation of the improved function using the linearization,

Firstly, the value of the following function is calculated,

$$f(x_i, \boldsymbol{\beta}+\boldsymbol{\delta}) \approx f(x_i, \boldsymbol{\beta}) + J_i\boldsymbol{\delta}$$

The output change is then approximated with the corresponding change in β using the above equation and the gradient of this change is calculated,

$$J_i = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$$

Once the algorithm reaches the optimal value of **β** (vector) then the gradient will become 0 and the learning stops.

Bayesian Regularization

Regularization is done to improve generalization of the training input data. The regularization is achieved by modifying the performance function of the trained neural network in an iterative process.

The performance function of training a feedforward function can be defined by the following equation,

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2$$

Generalization of this performance function is improved by adding a term that consists of the sum squares of the network weights and biases.

The regularized performance function can be represented by the following equation,

$$msereg = \gamma mse + (1-\gamma)msw$$

The parameter ϒ is the representation of performance ratio. The sum of weights is represented by the weights equation

$$msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2$$

Gradient Descent Learning

In the gradient descent learning algorithm, the aim like the above few learning mechanisms it to minimize the error which is given by the following equation for a linear system,

$$F(x) = \|A\mathbf{x} - \mathbf{b}\|^2.$$

To minimize the error, we need to minimize the error such that we need to learn in the direction that minimizes the cost function by taking the gradient.

$$\nabla F(\mathbf{x}) = 2A^T(A\mathbf{x} - \mathbf{b}).$$

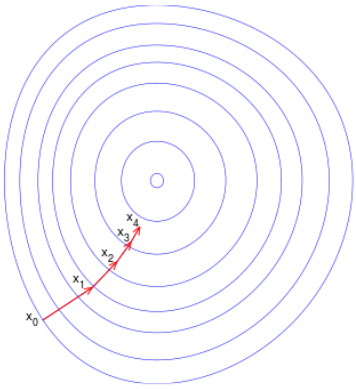


Figure 1: The direction of learning is towards the minimum cost function which in the case of neural networks is to minimize the error

Radial Basis Function

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

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$$

It involves choosing the bias function of the following form,

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi\left(\|\mathbf{x} - \boldsymbol{\mu}_i\|\right)$$

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.

Part 1: Classification Problem

The classification problem is one of the most studied problems in the field of Computer Science with applications in many other fields. It is a problem of identifying which categories of objects the new observation belongs to.

Of the many interesting applications is the classification of email into SPAM and NOT SPAM emails. This is a problem whose application in the real world is clearly identified with it saving countless number of hours for the people every single day. Every email service provided uses classification techniques to separate the emails of their clients.

Of the many classification techniques used is the one with the implementation of classification with the help of a trained neural network model. Neural network models have been successful in separating the handwritten digits for the MNIST dataset.

1. Single Hidden Layer Configurations

Settings:

1. Number of hidden layers = 1
2. Max_fail = 25
3. Maximum number of epochs = 200
4. Minimum Gradient = 1e-20
5. Training Data Split = 70:30
6. Training Method - trainlm

In the following three experiment results shown, varying the number of neurons in the hidden layer. The corresponding training, validation errors have been plotted in the graph and the testing error has been mentioned.

The experiments performed are:

- Number of neurons in hidden layer is 10
- Number of neurons in hidden layer is 46
- Number of neurons in hidden layer is 56
- Variation of the number of neurons and corresponding error plot

Number of neurons in first hidden layer is 10

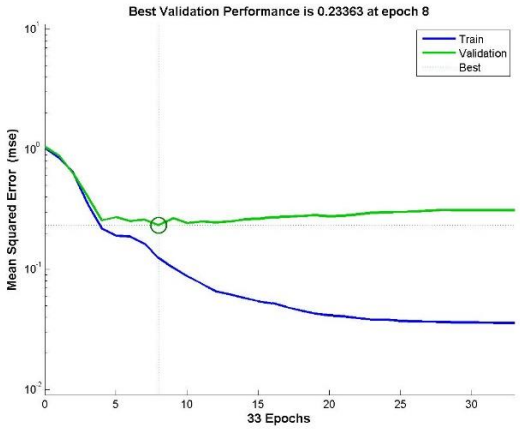


Figure 2: Training and Validation Error vs Number of Epochs

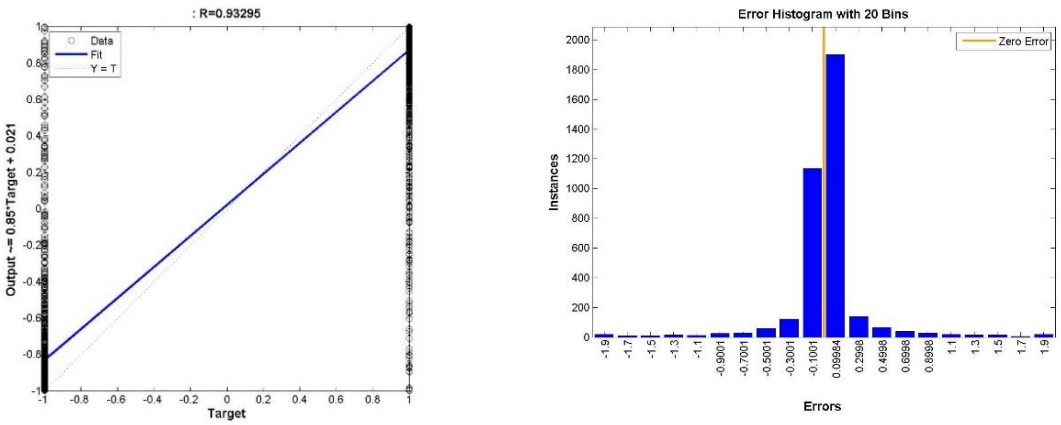


Figure 3: (a) Regression figure (b) Histogram for Neural Network with 1 hidden layer with 10 neurons

Misclassification error
0.0771739130434783

Number of neurons in first hidden layer is 46

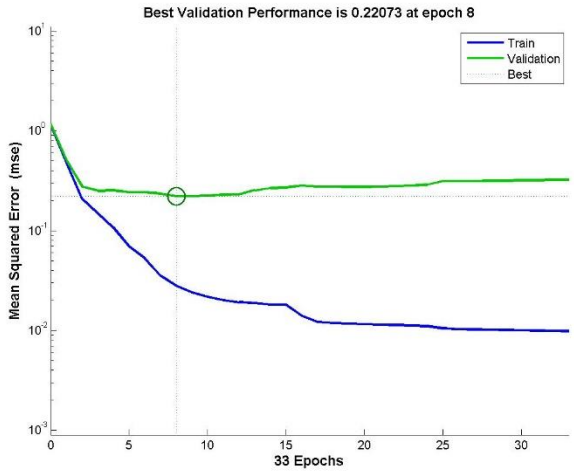


Figure 4: Training and Validation Error vs No. of Epochs for Neural Network with 1 hidden layer with 46 neurons

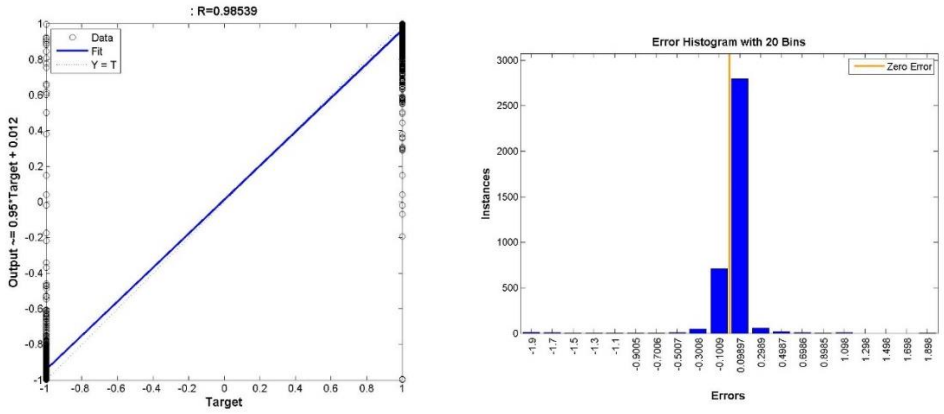


Figure 5: (a) Regression figure (b) Histogram for Neural Network with 1 hidden layer with 46 neurons

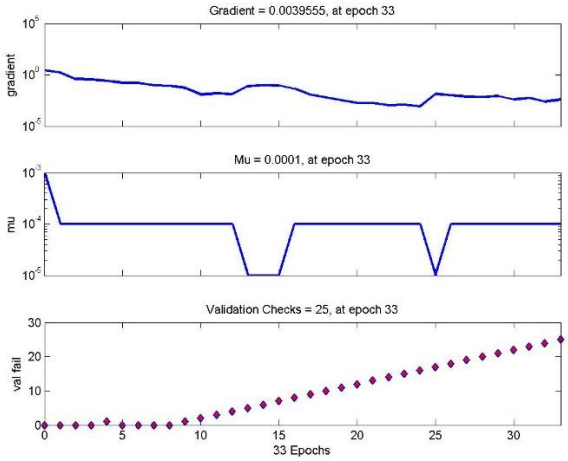


Figure 6: Gradient of the Function trained with Neural Network with 1 hidden layer of 46 neurons

Misclassification error
0.063043478260870

Number of neurons in first hidden layer is 56

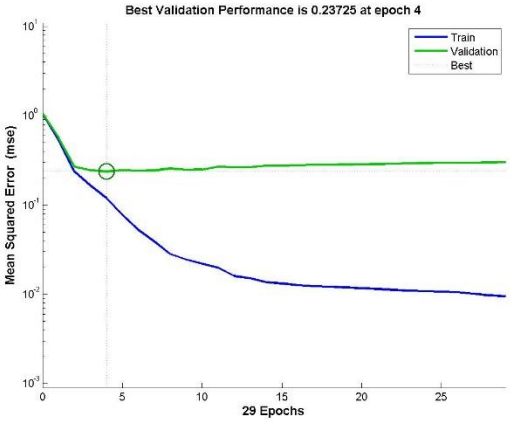


Figure 7: Training and Validation Error vs No. of Epochs for Neural Network with 1 hidden layer with 56 neurons

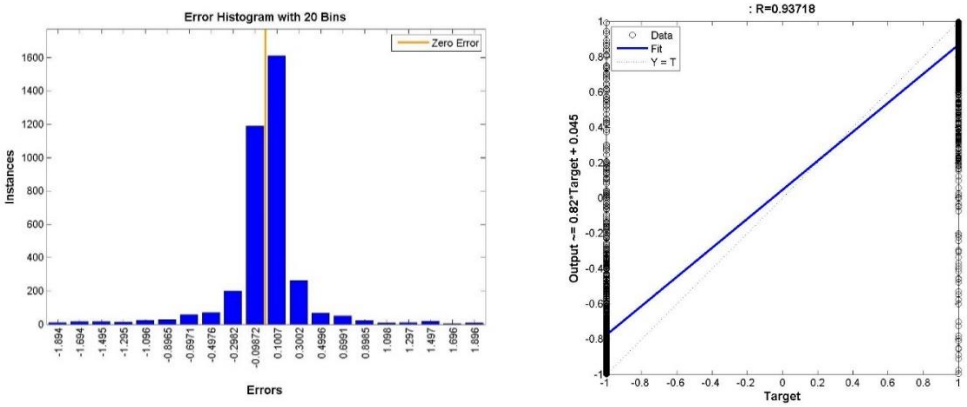


Figure 8: (a) Regression figure (b) Histogram for Neural Network with 1 hidden layer with 56 neurons

Misclassification error
0.0804347826086957

Variation of the number of neurons in the first hidden layer of the single hidden layer neural network

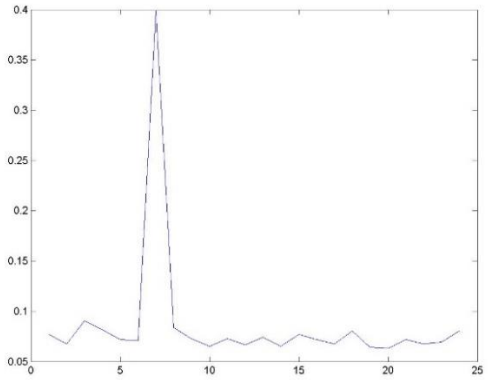


Figure 9: Misclassification Error vs (Number of neurons-10)/10 + 1

Analysis:

The graph shown above is the plot of the Misclassification error vs the performance of the number of neurons in the first hidden layer. The graph shows the stable behaviour when changing the number of neurons except for the spike when the number of neurons is low. On increasing the number of neurons it is expected to over fit the given data. The number of neurons have been changed from 10 to 56 providing 1 neuron for each of the input feature. However the plot does not seem to be showing the same result as expected, however increasing the number of neurons further would over fit the data under the given configuration settings.

2. Changing the Data Split Ration between Training and Validation Data: Data Split Ration changes to 60:40 for Training: Validation

The following settings were used to perform the experiments under this category,

Settings:

1. Number of hidden layers = 1
2. Max_fail = 25
3. Maximum number of epochs = 100
4. Minimum Gradient = 1e-20
5. Training Data Split = 60:40
6. Training Method - trainlm

The number of layer for this type of experiment was kept as a constant and the number of neurons within the layer was changed and the performance was measured. Just like the above category of experiments, the training and the validation plots have been shown. The regression plots of each of the individual networks have also been added in this section.

Number of Neurons in hidden layer = 10

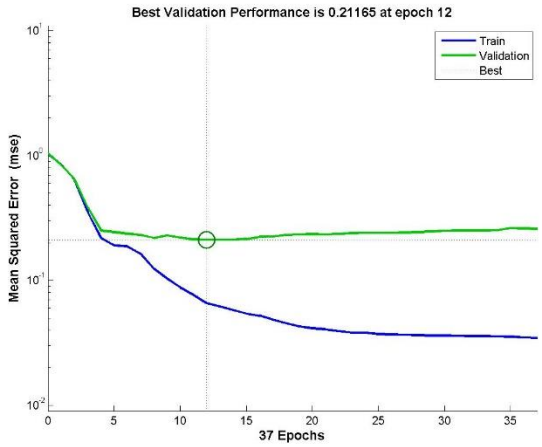


Figure 10: Training and Validation Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons with data split ratio of 60:40

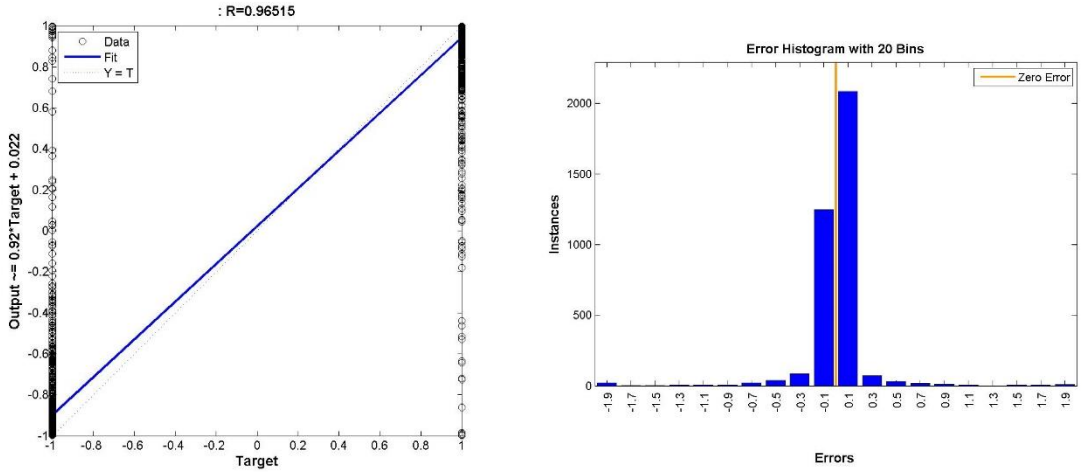


Figure 11: (a) Regression figure (b) Histogram for Neural Network with 1 hidden layer with 10 neurons with data split ration of 60:40 for the Training: Validation

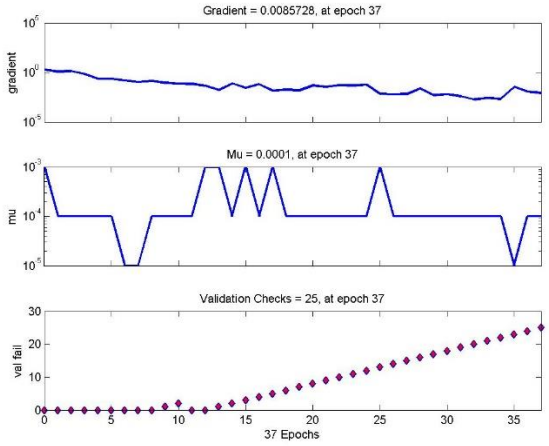


Figure 12: Gradient of the Function trained with Neural Network with 1 hidden layer of 10 neurons data split ration of 60:40 for the Training: Validation

Misclassification error
0.0641304347826087

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 Maximum Number of Validation non-decrease checks using max_fail

The following settings were used to perform the experiments under this category,

Settings:

1. Number of hidden layers = 1
2. Number of neurons in the hidden layer = 10
3. Maximum number of epochs = 400
4. Minimum Gradient = 1e-20
5. Training Data Split = 70:30
6. Training Method - trainlm

Note: In this experiment, the value of the maximum number of epoch has been kept sufficiently large to observe the behaviour of the neural network

The maximum number of validation checks is a stopping criterion in which, if the validation error fails to decrease atleast once in the number of times specified by max_fail. If the validation error decreases, the counter is reset for the next epoch.

The experiments performed are:

1. The value of max_fail = 10
2. The value of max_fail = 30
3. The value of max_fail = 70
4. The value of max_fail = 110
5. Variation of the testing error with the variation in the value of max_fail

Max_fail = 10

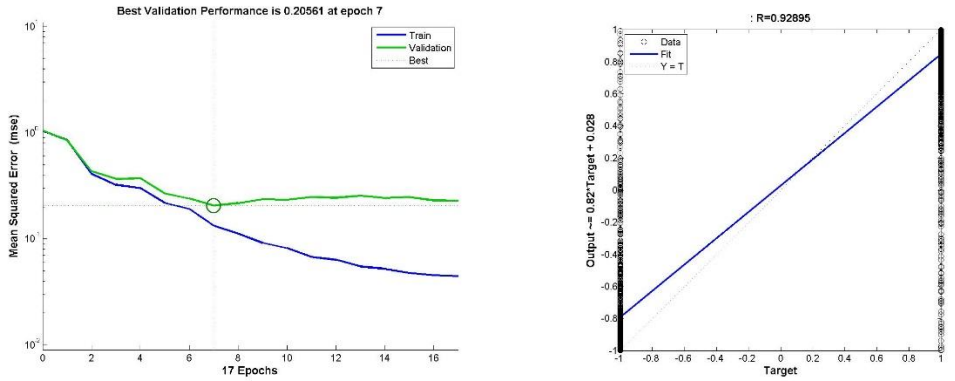


Figure 13: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer and Maximum Number of continuous Validation increase failures is 10

Misclassification error
0.0782608695652174

Max_fail = 30

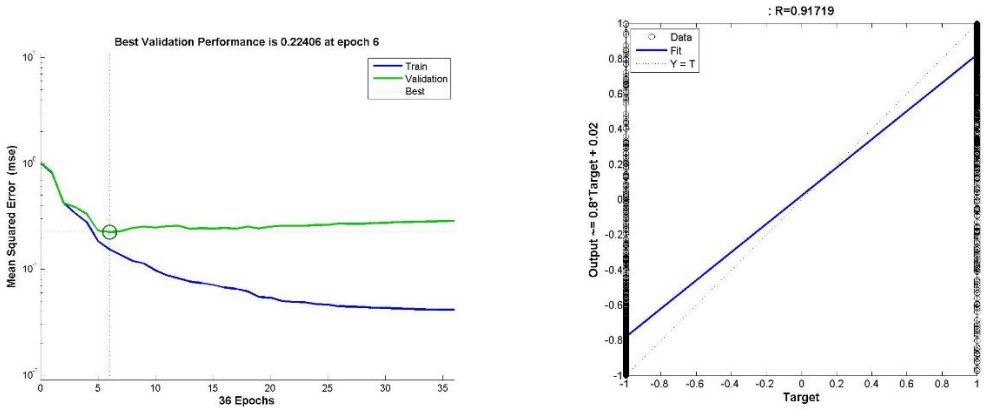


Figure 14: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer and Maximum Number of continuous Validation increase failures is 30

Misclassification error
0.066304347826087

Max_fail = 70

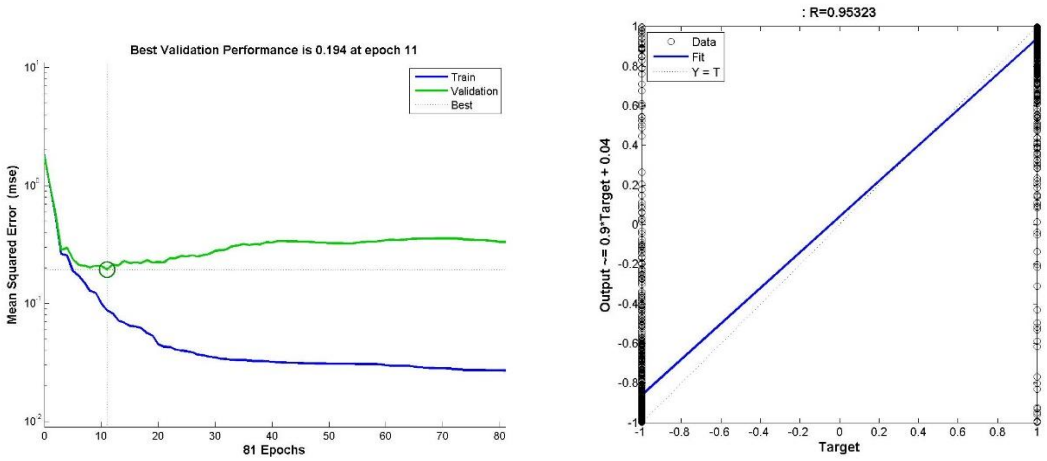


Figure 15: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer and Maximum Number of continuous Validation increase failures is 70

Misclassification error
0.061956521739130

Max_fail = 110

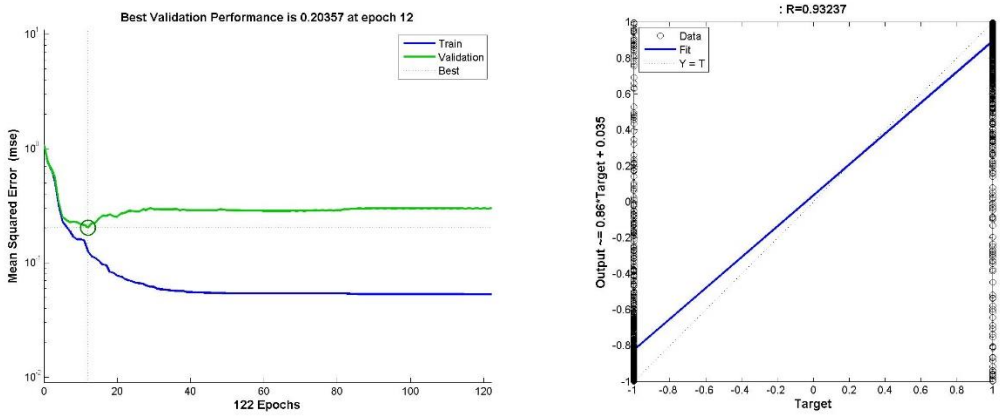


Figure 16: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer and Maximum Number of continuous Validation increase failures is 110

Misclassification error
0.0565217391304348

Plot of the variation of Testing Error with the change in Stopping Criterion (Max_fail)

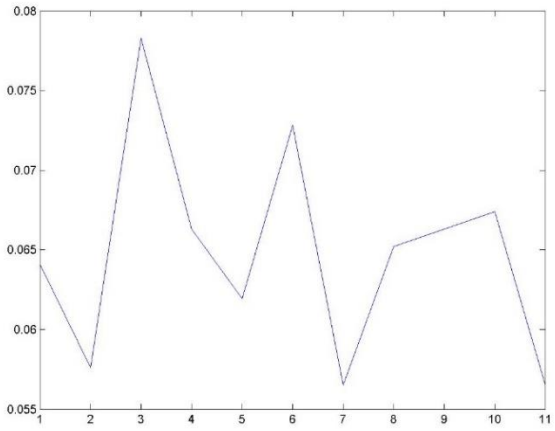


Figure 17: Testing Error vs Max_fail/10 for the Neural Network with 1 hidden layer of 10 neurons

Analysis:

In this experiment, the stopping criterion of training was varied and was studied. It can be observed that increasing the value of max_fail will ensure that network is properly trained and thus the time for completing the training of the network will increase substantially with no guarantee in the improvement of performance as shown in the figure above. However for larger neural networks model it can be observed that increasing the value increase the performance in the beginning and the performance stabilizes.

4. Varying both the number of hidden layers and the number of neurons in each layer

The following settings were used to perform the experiments under this category,

Settings:

1. Max_fail = 25
2. Maximum number of epochs = 100
3. Minimum Gradient = 1e-20
4. Training Data Split = 70:30
5. Training Method - trainlm

Number of Hidden Layers = 2 and the configuration of the layers are [20 20]

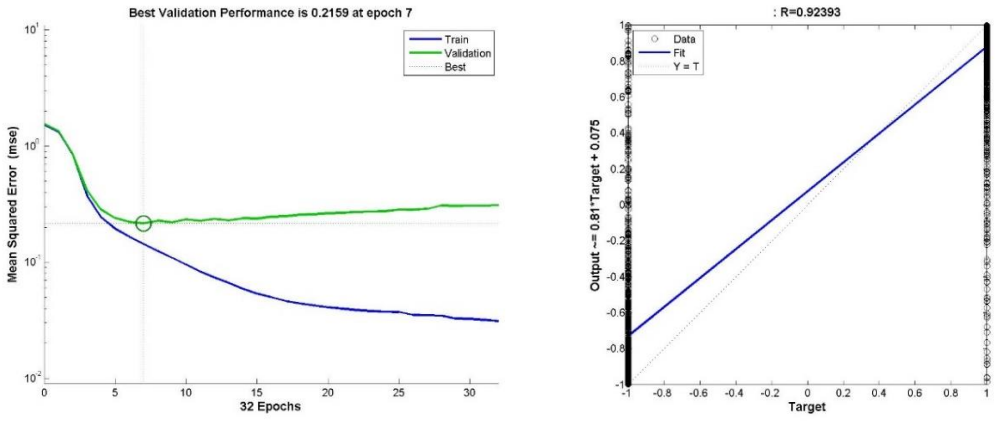


Figure 18: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 2 layer with hidden configuration of [20 20]

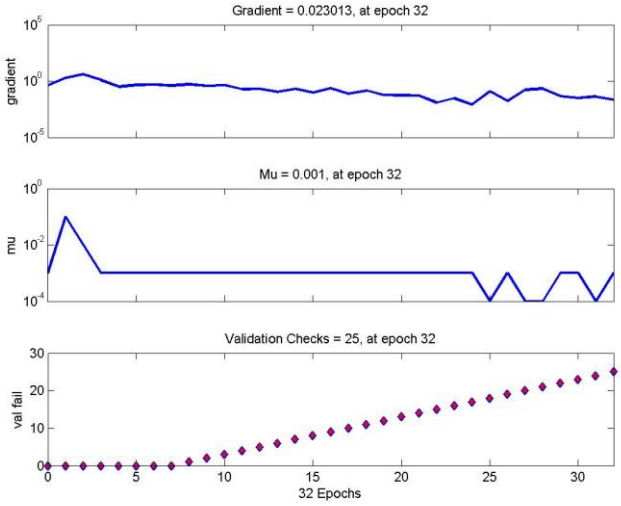


Figure 19: Training State of a neural network with 2 layer with hidden configuration of [20 20]

Misclassification error
0.073913043478261

Number of hidden layer = 2 and configuration of hidden layers [30 30]

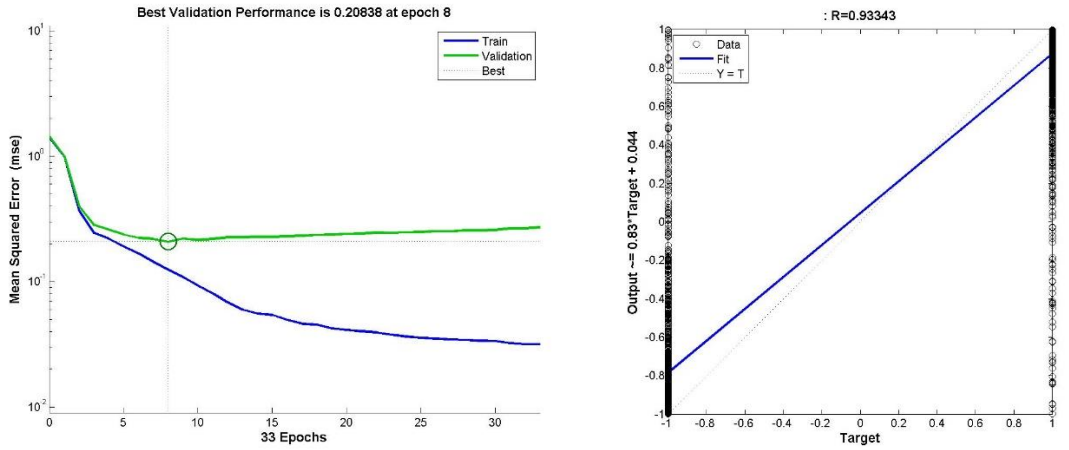


Figure 20: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 2 layer with hidden configuration of [30 30]

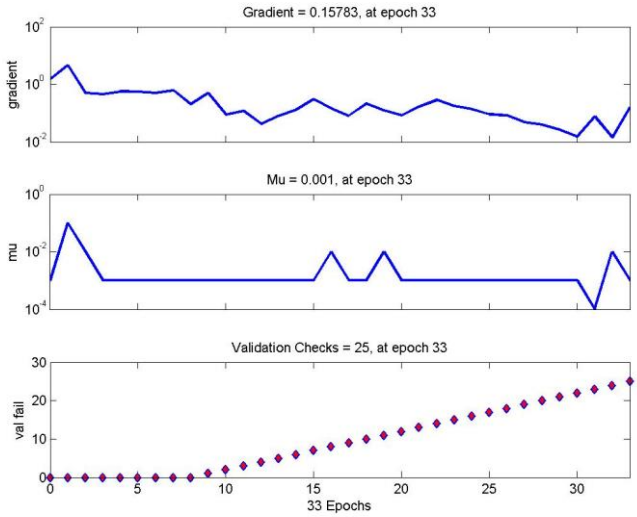


Figure 21: Training State of a neural network with 2 layer with hidden configuration of [30 30]

Misclassification error
0.063043478260870

Analysis:

On increasing the number of layers and the number of neurons in each layer, the performance of the neural network in terms of the reduction of the testing error of the system initially decreases but then the testing error starts increasing.

The idea behind increasing the number of layers in the neural network is to increase the number till the performance of the neural network stops to increase and starts decreasing. Thus, keeping the number of neurons for each layer constant and increasing the number of layers till the error stops decreasing and then we can vary the number of neurons in each layer once the optimum number of layers is found.

Variation of Misclassification rate with number of Neurons in the first layer when the second layer is fixed constant

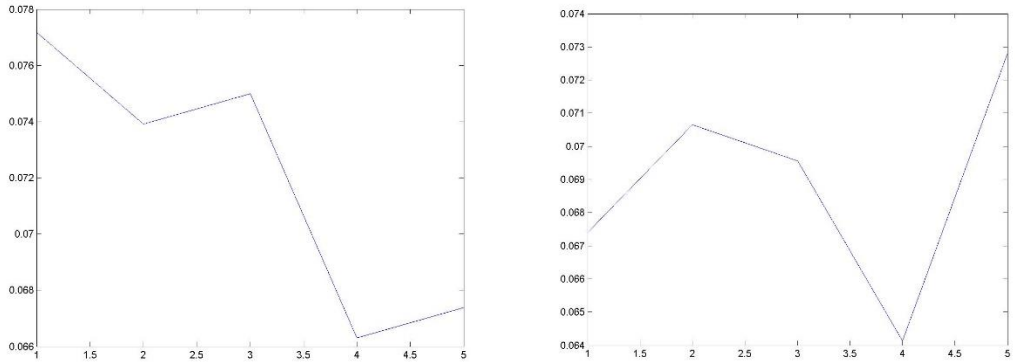


Figure 22: Testing Error (Misclassification Rate) vs Number of neurons in the first layer (a) when the number of neurons in the second layer is fixed at 10 (b) when the number of neurons in the second layer is fixed at 20

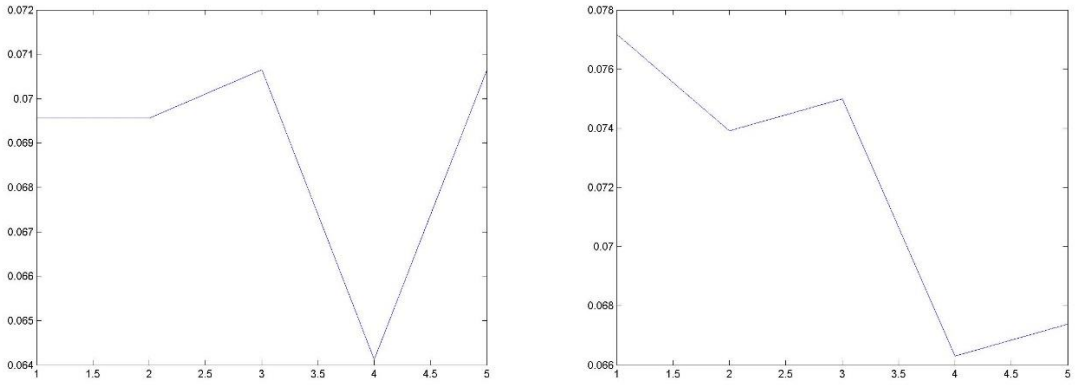


Figure 23: Testing Error (Misclassification Rate) vs Number of neurons in the first layer (a) when the number of neurons in the second layer is fixed at 30 (b) when the number of neurons in the second layer is fixed at 40

Analysis:

Above plotted is the performance (Misclassification Testing Error) of the trained neural network models when the number of neurons in the second layer was kept constant at the number shown below in the caption of each of the figures.

5. Changing the Backpropagation Mechanism: Bayesian Regularization Training

The following settings were used to perform the experiments under this category,

Settings:

1. Number of hidden layers = 1
2. Max_fail = 25
3. Maximum number of epochs = 200
4. Minimum Gradient = 1e-20
5. Training Data Split = 70:30
6. Training Method - trainbr

The concept of Bayesian Regression has been explained in the first part of this report. Using the Bayesian regression backpropagation technique, different neural networks were trained and their testing errors were measured. The Bayesian regression function does not include the validation performance as explained in the first part.

Number of layer = 1 with hidden network configuration of [10]

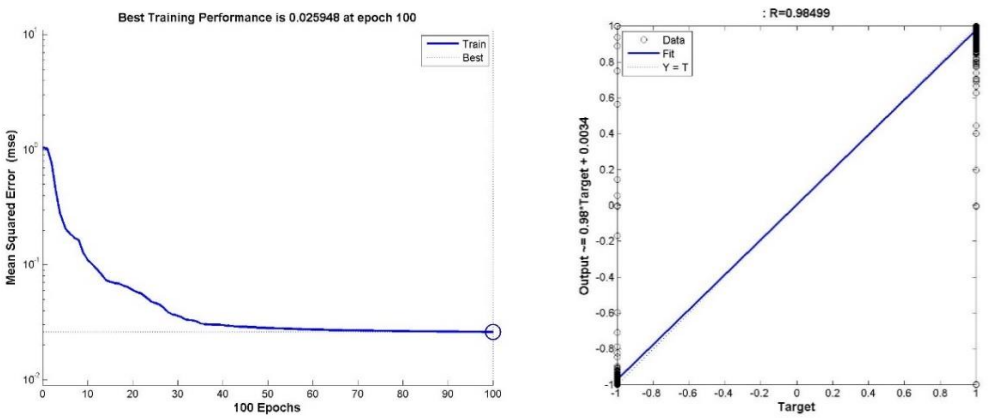


Figure 24: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Bayesian Regularization training

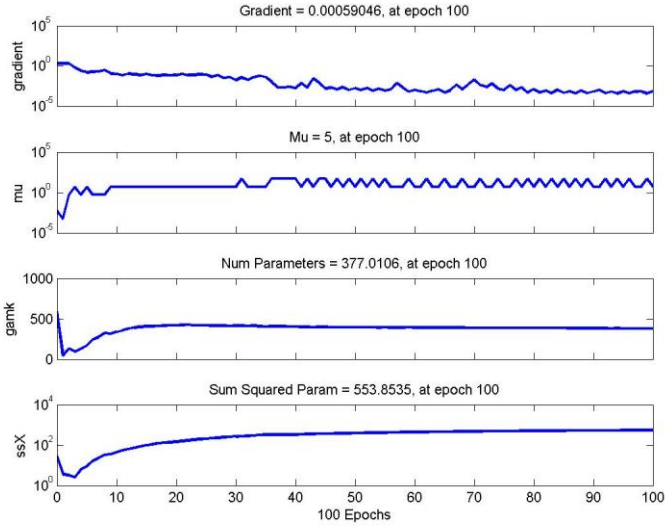


Figure 25: Gradient of the Function trained with Neural Network with 1 hidden layer of 10 neurons and training function of Bayesian Regularization

Misclassification error
0.0076

Number of layer = 1 with hidden network configuration of [40]

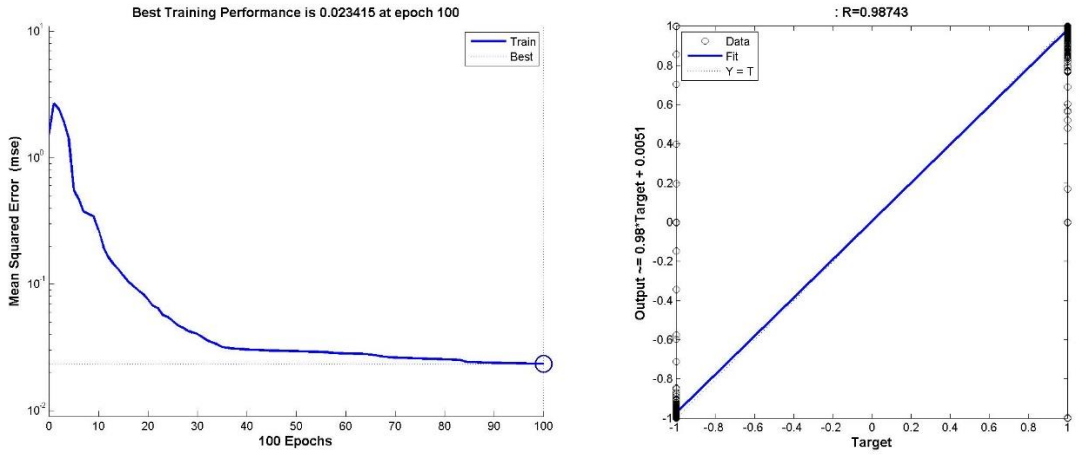


Figure 26: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 40 neurons and Bayesian Regularization training

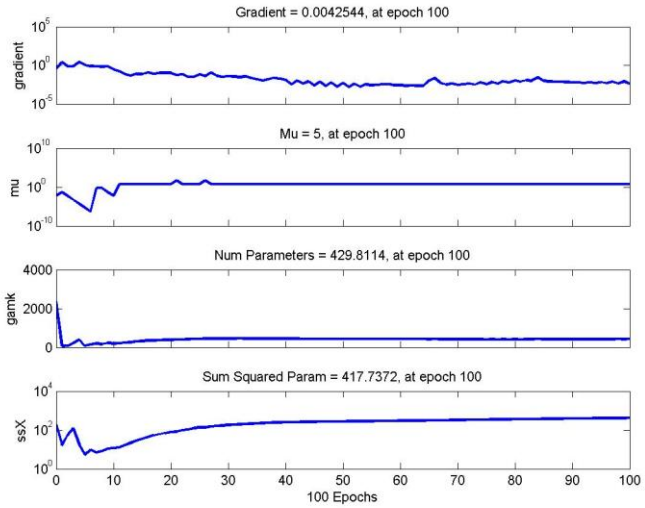


Figure 27: Gradient of the Function trained with Neural Network with 1 hidden layer of 10 neurons and training function of Bayesian Regularization

Misclassification error
0.006521739130435

Analysis:

It can be observed that the Misclassification error on the testing data is significantly much lower for the Bayesian Regression backpropagation technique for this Multi-Layer Perceptron based neural network. It can be observed that the performance of this neural network is 10 times better than that of the corresponding errors obtained by using the LMA backpropagation algorithm. In this technique of backpropagation, there is no validation done and the complete data provided in the training data set is used. Overall, the testing error of the Bayesian Regression based MLP is much better and so is the performance.

6. Training Function – Gradient Descent

Since in the Lavenberg-Marquardt Algorithms and the Bayesian Regularization training algorithms, the training rate parameter cannot be controlled, testing that parameter with the gradient descent learning rule.

The following settings were used to perform the experiments under this category,

Settings:

1. Max_fail = 25
2. Maximum number of epochs = 200
3. Minimum Gradient = 1e-20
4. Training Data Split = 70:30
5. Learning Rate = 0.0001
6. Number of layers = 1
7. Training Method - traingd

The following experiments were performed under this category,

1. Number of hidden layers = 1 with configuration[10] and Learning rate of 0.0001
2. Number of hidden layers = 1 with configuration[10] and Learning rate of 0.05
3. Number of hidden layers = 1 with configuration[10] and Learning rate of 0.26
4. Number of hidden layers = 1 with configuration[10] and Learning rate of 0.51

5. Number of hidden layers = 2 with configuration[10 10] and Learning rate of 0.01
6. Number of hidden layers = 2 with configuration[10 10] and Learning rate of 0.26
7. Number of hidden layers = 2 with configuration[10 10] and Learning rate of 0.51

Number of hidden layers = 1 with configuration [10] and Learning rate 0.0001

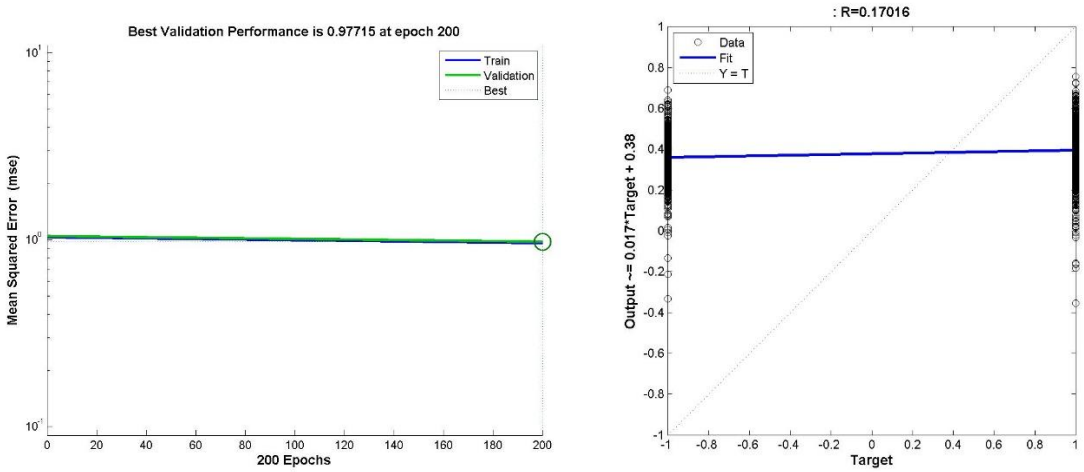


Figure 28: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.0001

Misclassification error
0.578260869565217

Learning rate 0.05

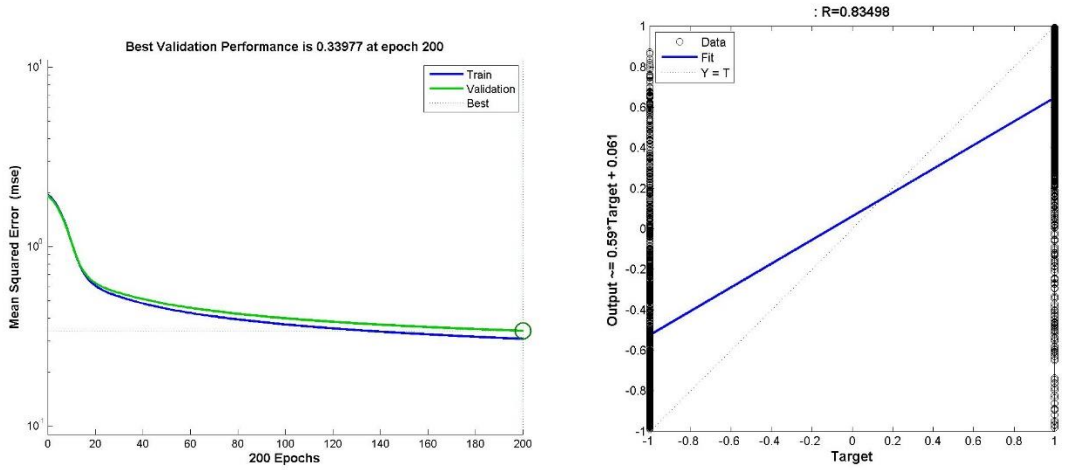


Figure 29: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.05

Misclassification error
0.086956521739130

Number of hidden layers = 1 with configuration [10] and Learning rate 0.26

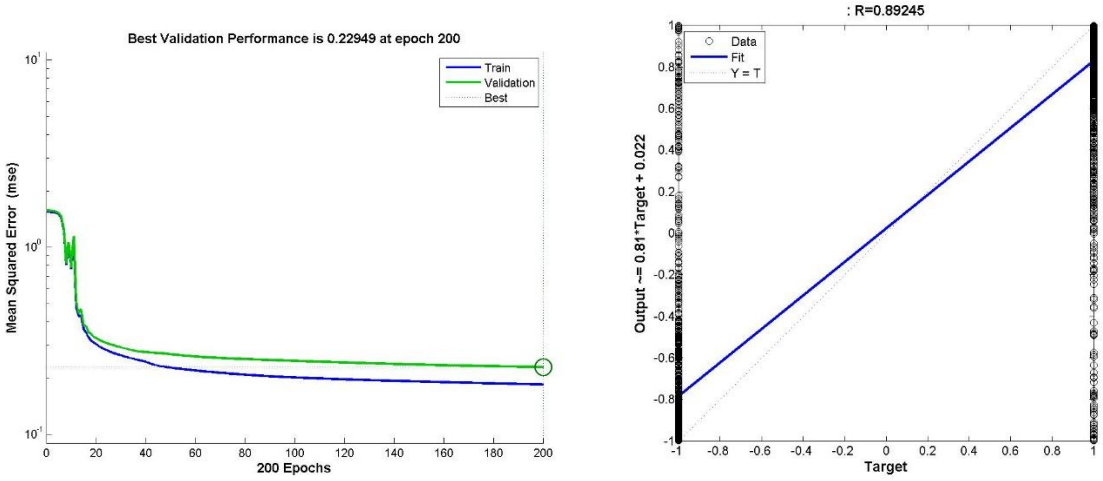


Figure 30: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.26

Misclassification error
0.078260869565217

Number of hidden layers = 1 with configuration [10] and Learning rate 0.51

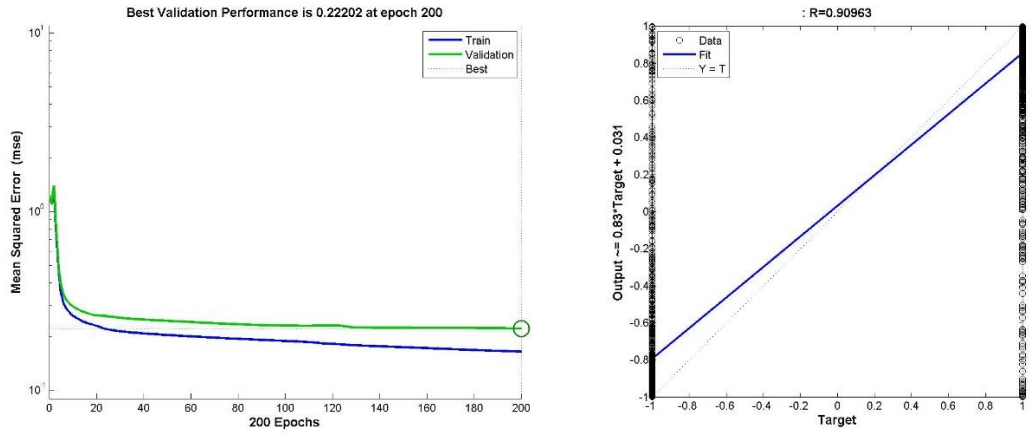


Figure 31: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.51

Misclassification error
0.071739130434783

Number of hidden layers = 2 with configuration [10 10] and Learning rate of 0.01

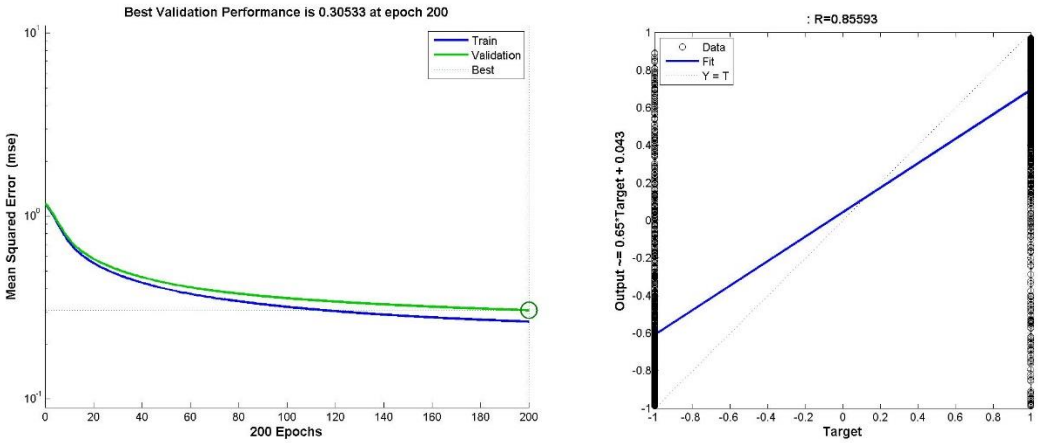


Figure 32: Training Error vs No. of Epochs for Neural Network with 2 hidden layers configuration of [10 10] and Gradient Descent Learning with learning rate of 0.01

Misclassification error
0.089130434782609

Number of hidden layers = 2 with configuration [10 10] and Learning rate of 0.26

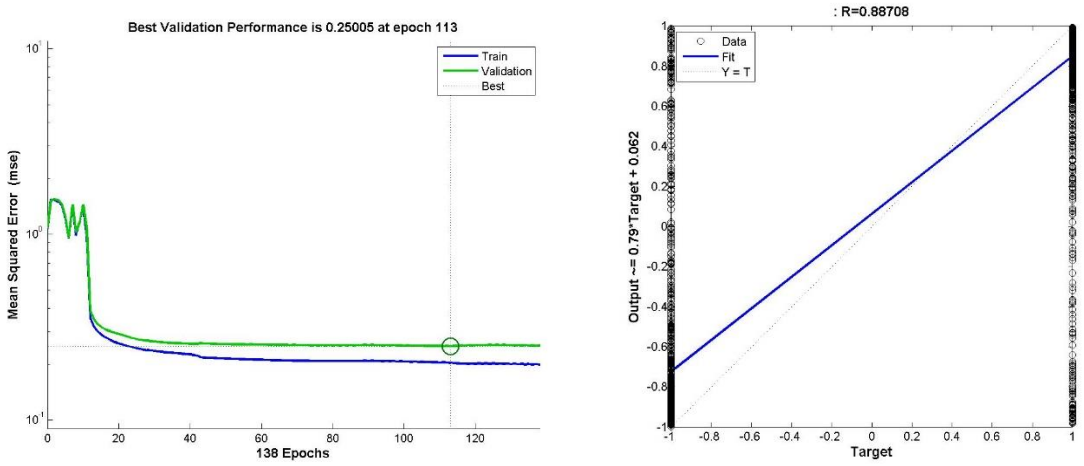


Figure 33: Training Error vs No. of Epochs for Neural Network with 2 hidden layers configuration of [10 10] and Gradient Descent Learning with learning rate of 0.26

Misclassification error
0.067391304347826

Number of hidden layers = 2 with configuration [10 10] and Learning rate of 0.51

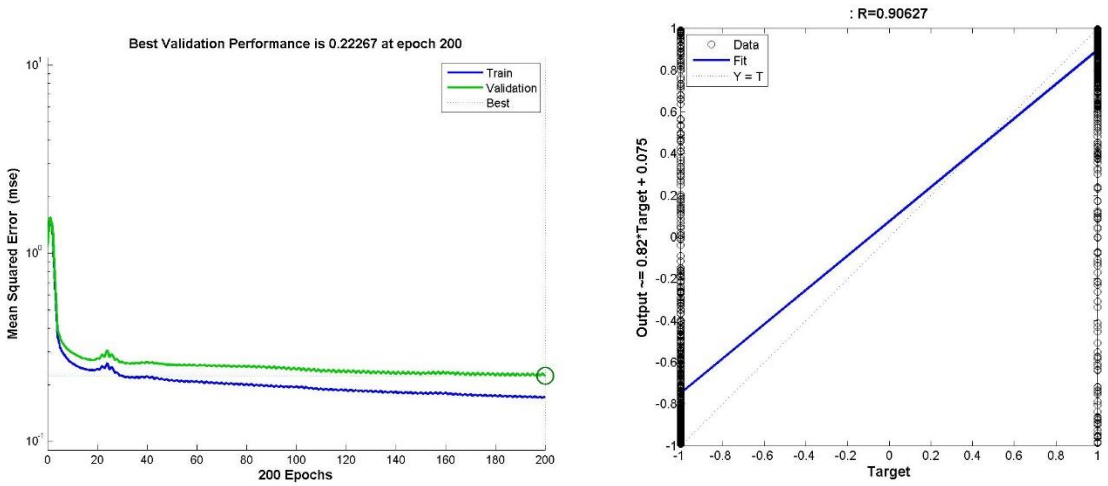


Figure 34: Training Error vs No. of Epochs for Neural Network with 2 hidden layers configuration of [10 10] and Gradient Descent Learning with learning rate of 0.51

Misclassification error
0.072826086956522

Analysis:

When the value of learning rate is very low, then the network does not learn much within the maximum number of specified epochs. On increasing the value of learning rate the network starts learning quickly and might reach the optimum value within the specified number of epochs. But however if the value of the learning rate is too high then the function might oscillate back and forth between the optimum value of the cost function.

Thus it is very important to choose the learning rate appropriately so that the function reaches the optimum value.

Part 2: Function Approximation Problem

The approximation problem is also one of the most studied problems in Computer Science. The function approximation tasks us with choosing the best mathematical model for a particular problem and data set based that better fits it.

Function approximation has various application to the real world activities including the mapping of Stock prices of different stocks. In this part, the task to approximate the function for the fitting the California housing prices to better predict the unseen prices and also help in obtaining profits but buying and selling at the appropriate time.

1. Single Hidden Layer Configurations

The following settings were used to perform the experiments under this category,

Settings:

1. Number of hidden layers = 1
2. Number of neurons in the hidden layer = 10
3. Max_fail = 25
4. Maximum number of epochs = 200
5. Minimum Gradient = 1e-20
6. Training Data Split = 70:30
7. Training Method - trainlm

In the following three experiment results shown, varying the number of neurons in the hidden layer. The corresponding training, validation errors have been plotted in the graph and the testing error has been mentioned.

The experiments performed are:

- Number of neurons in hidden layer is 10
- Number of neurons in hidden layer is 30
- Variation of the number of neurons and corresponding error plot

Number of neurons in hidden layer is 10

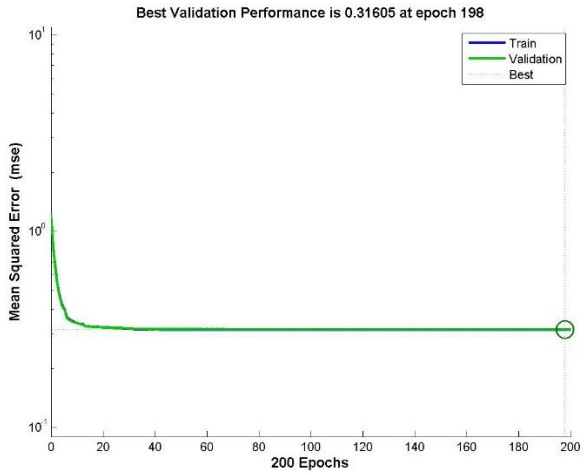


Figure 35: Training and Validation Error vs Number of Epochs

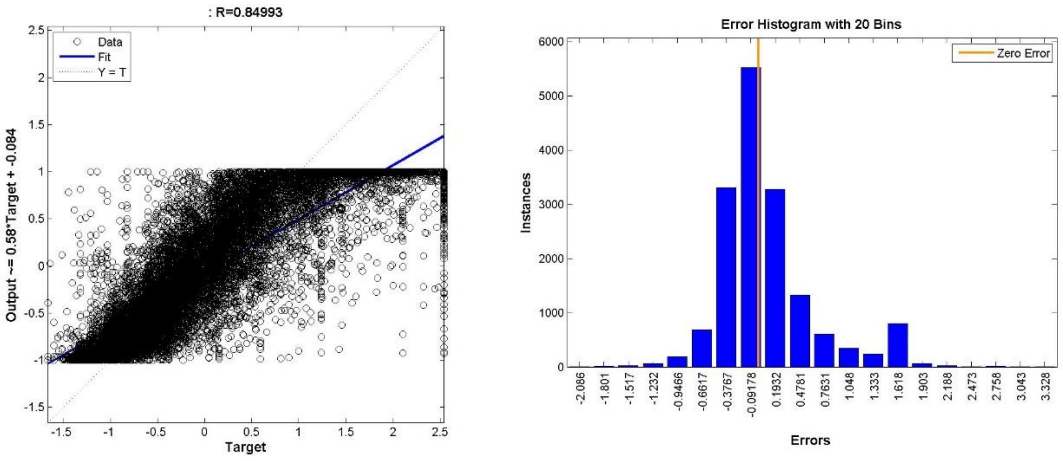


Figure 36: (a) Regression figure (b) Histogram for Neural Network with 1 hidden layer with 10 neurons

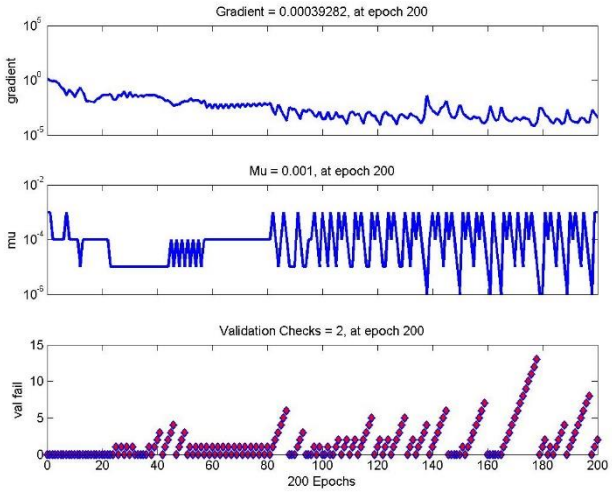


Figure 37: Gradient of the Function trained with Neural Network with 1 hidden layer of 10 neurons

RMSE
65909.1838608107

Number of neurons in hidden layer is 10

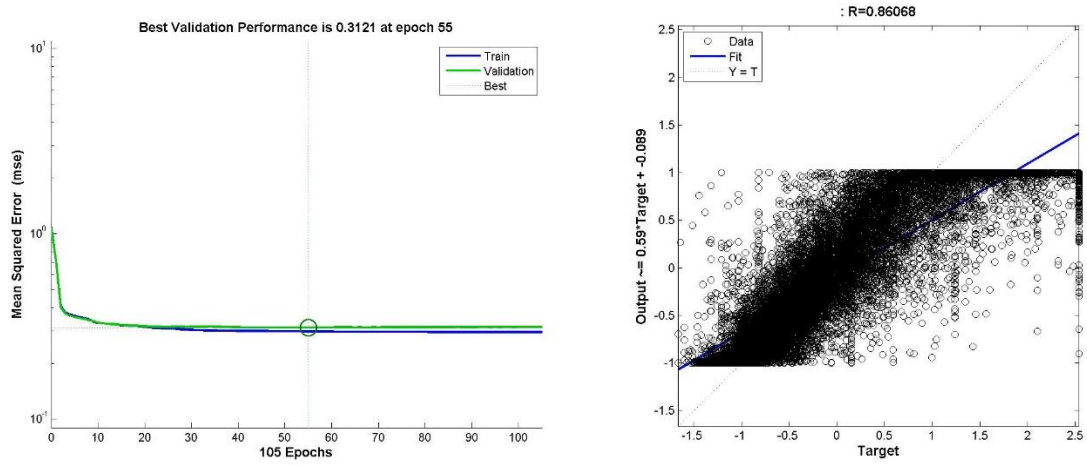


Figure 38: (a) Training and Validation Error vs No. of Epochs (b) Regression Model for function approximation of a neural network with 30 neurons in the single hidden layer

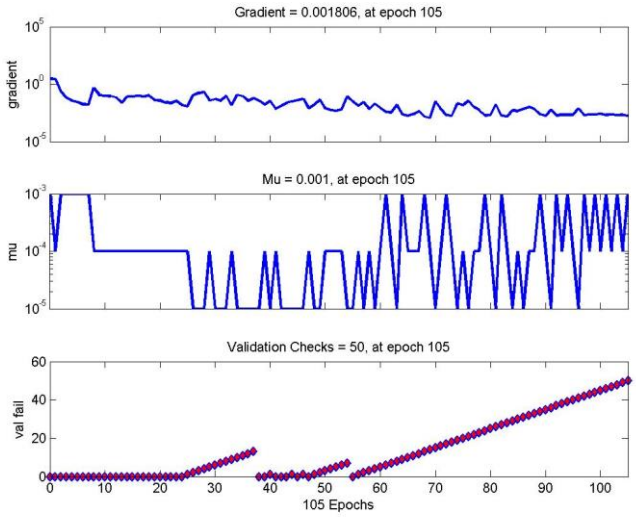


Figure 39: Gradient of the Function trained with Neural Network with 1 hidden layer of 30 neurons

RMSE
6.452296108659939e+04

Variation of the number of neurons and corresponding error plot

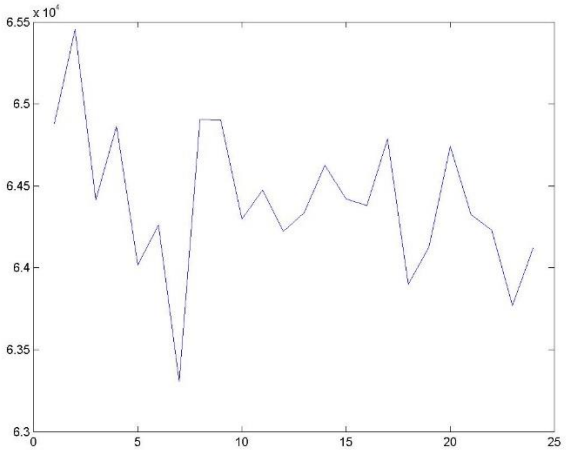


Figure 40: Test Error vs (Number of Neurons in the hidden layer – 10)/2 + 1

Analysis:

The graph shown above is the plot of the Misclassification error vs the performance of the number of neurons in the first hidden layer. The graph shows a steadily decreasing error performance with the increase in the number of neurons in the first hidden layer. The number of neurons in the hidden layer have been increased from 10 to 56 and the Testing error with the points from the test data was noted and has been displayed.

2. Changing the Data Split Ration between Training and Validation Data: Data Split Ration changes to 60:40 for Training: Validation

The following settings were used to perform the experiments under this category,

Settings:

1. Number of hidden layers = 1
2. Max_fail = 25
3. Maximum number of epochs = 200
4. Minimum Gradient = 1e-20
5. Training Data Split = 60:40
6. Training Method - trainlm

The number of layer for this type of experiment was kept as a constant and the number of neurons within the layer was changed and the performance was measured. Just like the above category of experiments, the training and the validation plots have been shown. The regression plots of each of the individual networks have also been added in this section.

Number of neurons in the hidden layer = 10

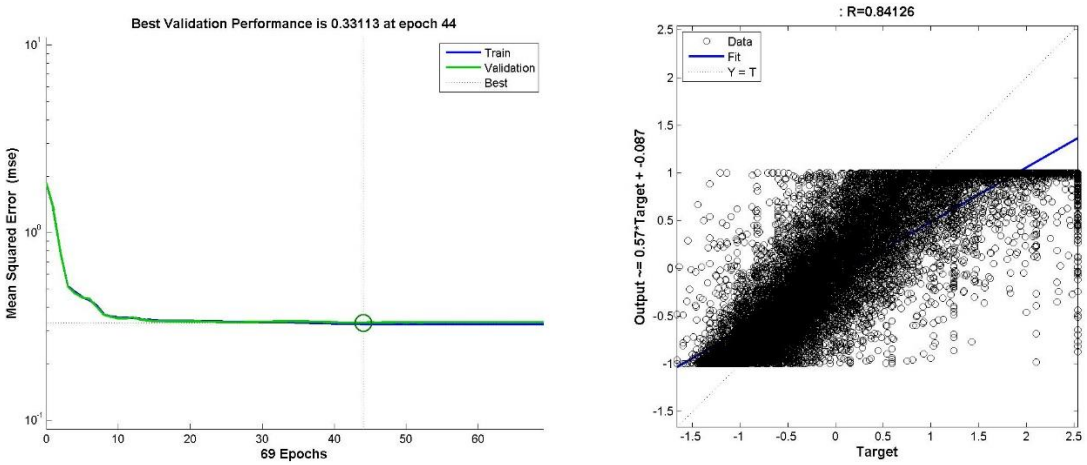


Figure 41: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer

RMSE
6.641306572411611e+04

Number of neurons in the hidden layer = 30

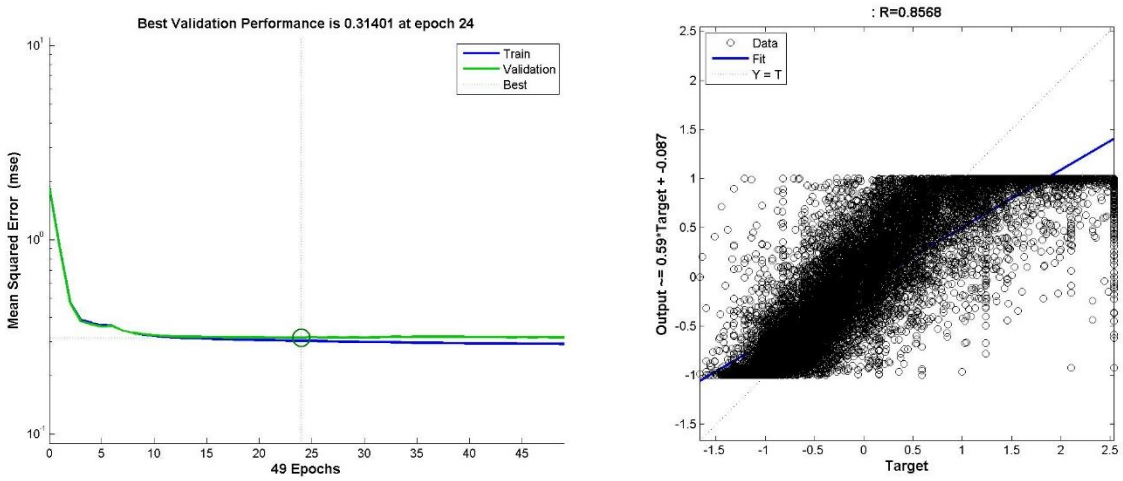


Figure 42: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer

RMSE
6.406994715495104e+04

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 Maximum Number of Validation non-decrease checks using max_fail

The following settings were used to perform the experiments under this category,

Settings:

1. Number of hidden layers = 1
2. Number of neurons in the hidden layer = 10
3. Maximum number of epochs = 400
4. Minimum Gradient = 1e-20
5. Training Data Split = 70:30
6. Training Method - trainlm

Note: In this experiment, the value of the maximum number of epoch has been kept sufficiently large to observe the behaviour of the neural network

The maximum number of validation checks is a stopping criterion in which, if the validation error fails to decrease atleast once in the number of times specified by max_fail. If the validation error decreases, the counter is reset for the next epoch.

The experiments performed are:

1. The value of max_fail = 30
2. The value of max_fail = 60

Value of max_fail = 30

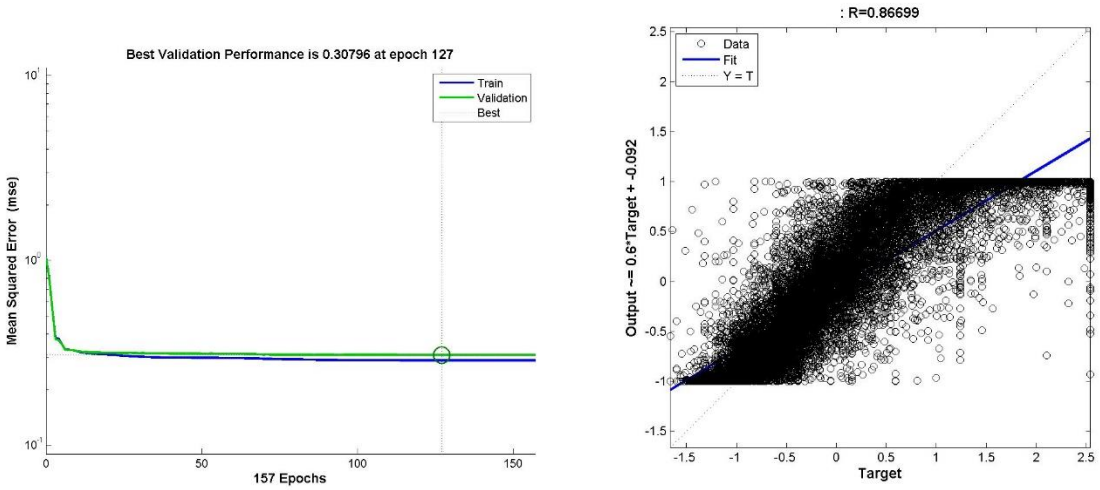


Figure 43: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer and Maximum Number of continuous Validation increase failures is 30

RMSE
6.451097853699548e+04

Value of max_fail = 60

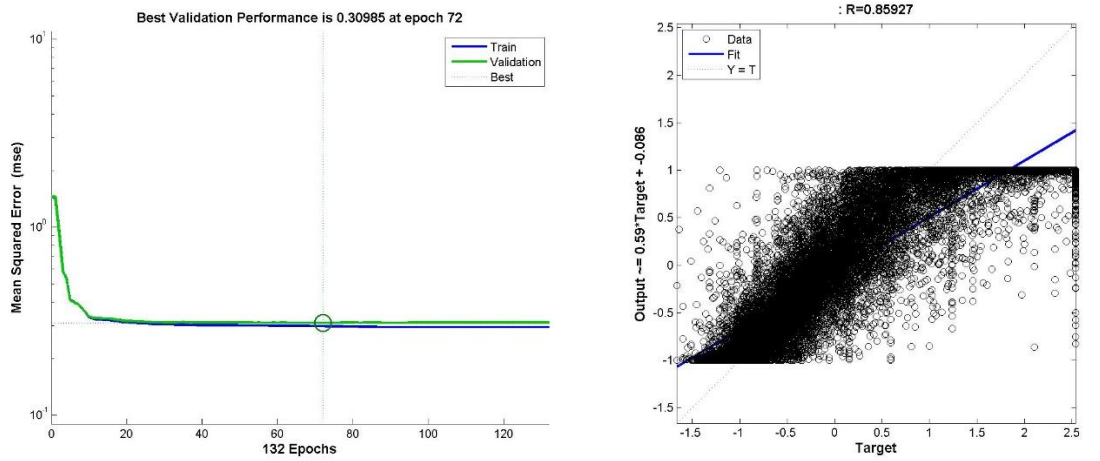


Figure 44: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 1 layer and Maximum Number of continuous Validation increase failures is 100

RMSE
6.388928277055417e+04

Analysis:

In this experiment, the stopping criterion of training was varied and was studied. It can be observed that increasing the value of max_fail will ensure that network is properly trained and thus the time for completing the training of the network will increase substantially with no guarantee in the improvement of performance as shown in the figure above. However for larger neural networks model it can be observed that increasing the value increase the performance in the beginning and the performance stabilizes.

4. Varying both the number of hidden layers and the number of neurons in each layer

The following settings were used to perform the experiments under this category,

Settings:

1. Max_fail = 25
2. Maximum number of epochs = 200
3. Minimum Gradient = 1e-20
4. Training Data Split = 70:30
5. Training Method - trainlm

The experiments performed are:

- Number of layers = 2 and the hidden network configuration is [20 20]
- Number of layers = 2 and the hidden network configuration is [30 30]
- Variation of the number of neurons and corresponding error plot for 1 fixed layer
- Number of layers = 3 and the hidden network configuration is [20 20 20]
- Number of layers = 4 and the hidden network configuration is [20 20 20 20]
- Number of layers = 5 and the hidden network configuration is [20 20 20 20 20]

Number of layers = 2 and the network configuration is [20 20]

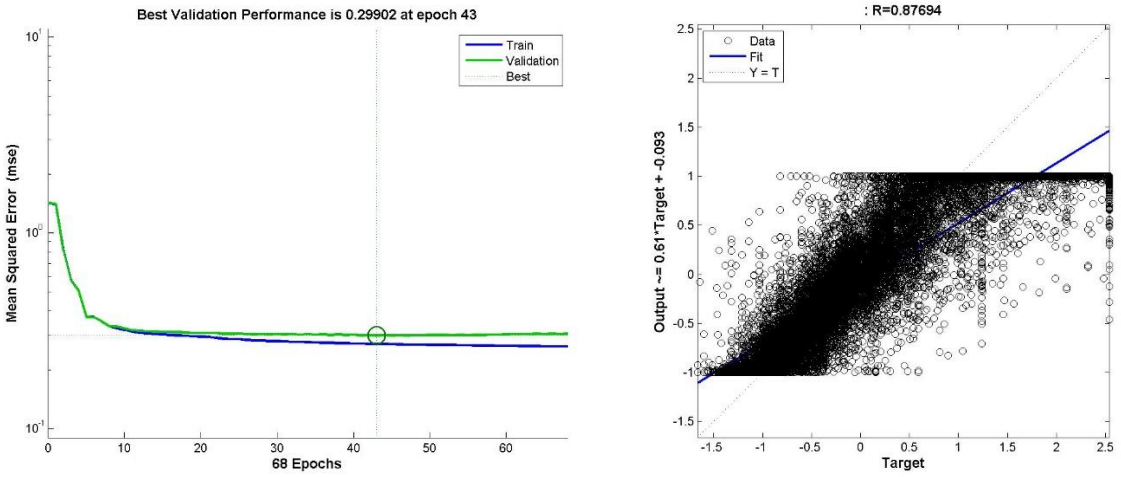


Figure 45: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 2 layer with hidden configuration of [20 20]

RMSE
6.283858088469141e+04

Number of layers = 2 and the network configuration is [30 30]

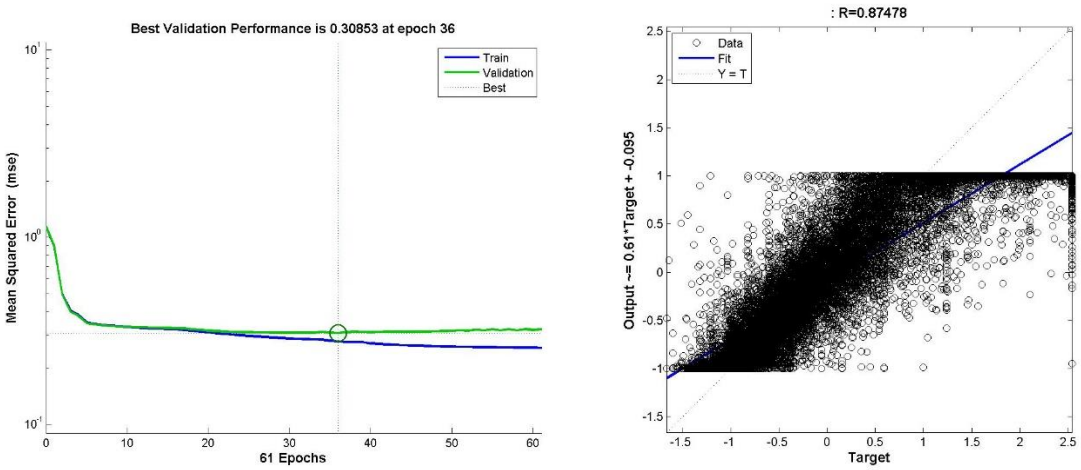


Figure 46: (a) Performance Measure Training and Validation Error vs No. of Epochs (b) Regression output for a neural network with 2 layer with hidden configuration of [30 30]

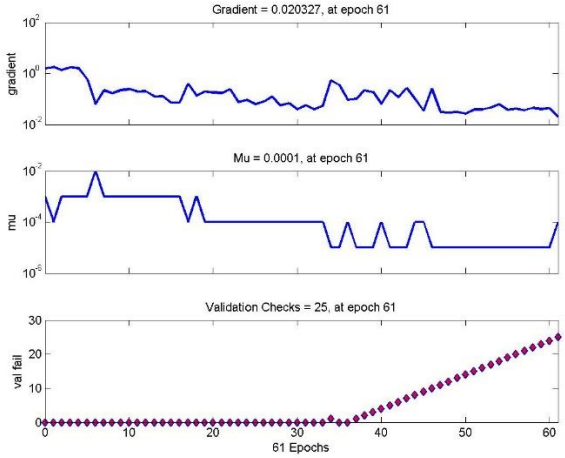


Figure 47: Training State of a neural network with 2 layer with hidden configuration of [20 20]

RMSE
6.357500937533724e+04

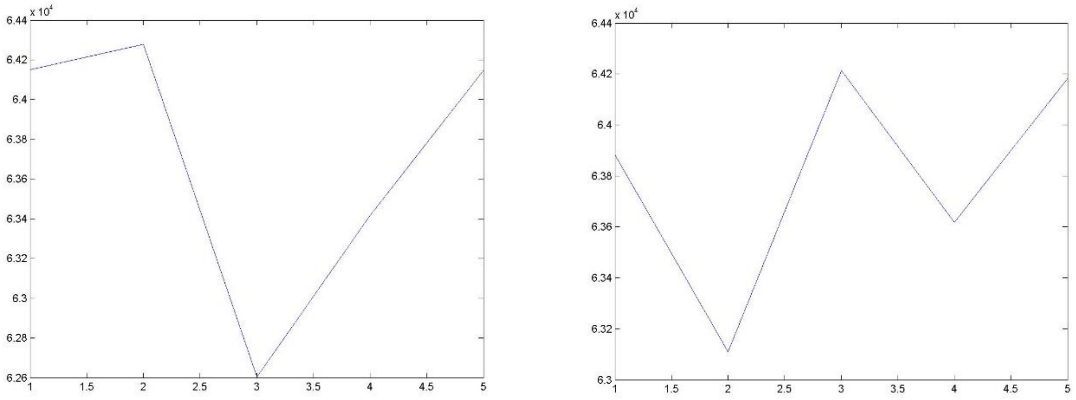


Figure 48: Testing Error (Misclassification Rate) vs Number of neurons/10 in the first layer (a) when the number of neurons in the second layer is fixed at 10 (b) when the number of neurons in the second layer is fixed at 20

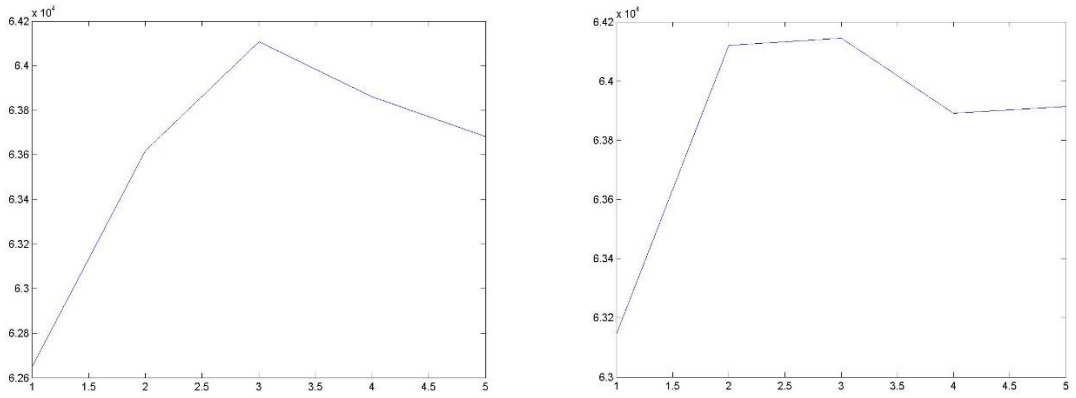


Figure 49: Testing Error (Misclassification Rate) vs Number of neurons/10 in the first layer (a) when the number of neurons in the second layer is fixed at 20 (b) when the number of neurons in the second layer is fixed at 40

Analysis:

Above plotted is the performance (Misclassification Testing Error) of the trained neural network models when the number of neurons in the second layer was kept constant at the number shown below in the caption of each of the figures.

Number of layers = 3 and the hidden network configuration is [20 20 20]

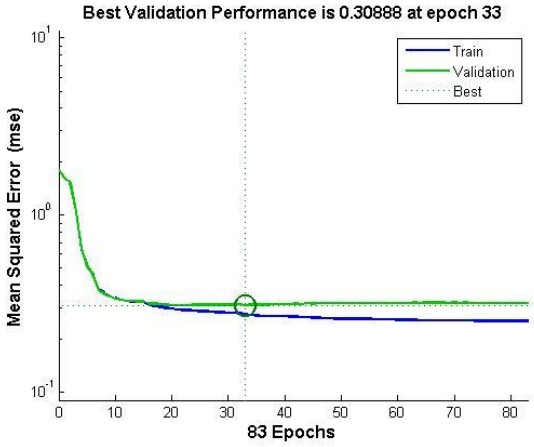


Figure 50: Performance Measure Training and Validation Error vs No. of Epochs for a neural network with 3 layer with hidden configuration of [20 20 20]

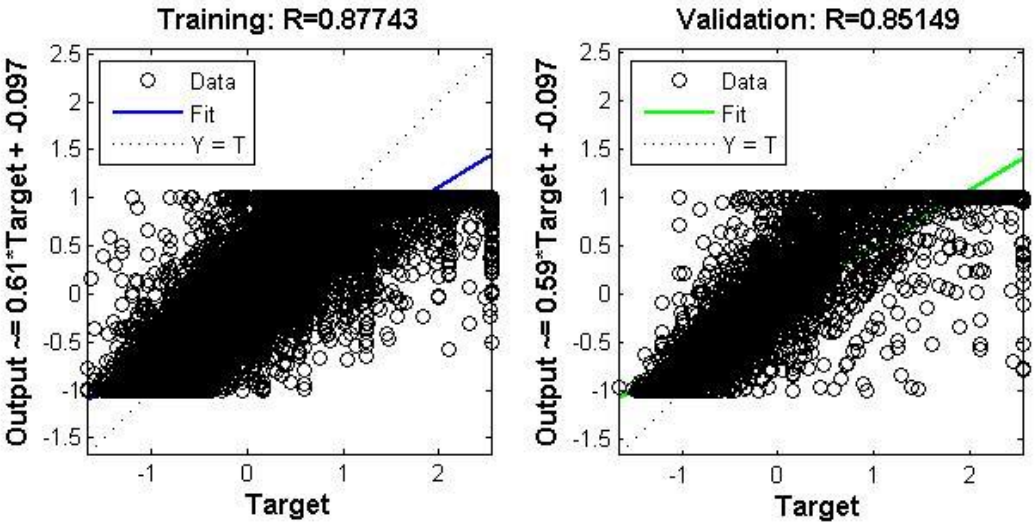


Figure 51: (a) Training Regression (b) Validation Regression for a neural network with 3 layer with hidden configuration of [20 20 20]

RMSE
6.333598399181195e+04

Number of layers = 4 and the hidden network configuration is [20 20 20 20]

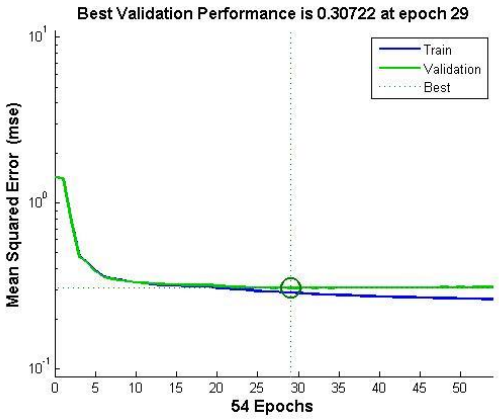


Figure 52: Performance Measure Training and Validation Error vs No. of Epochs for a neural network with 4 layer with hidden configuration of [20 20 20 20]

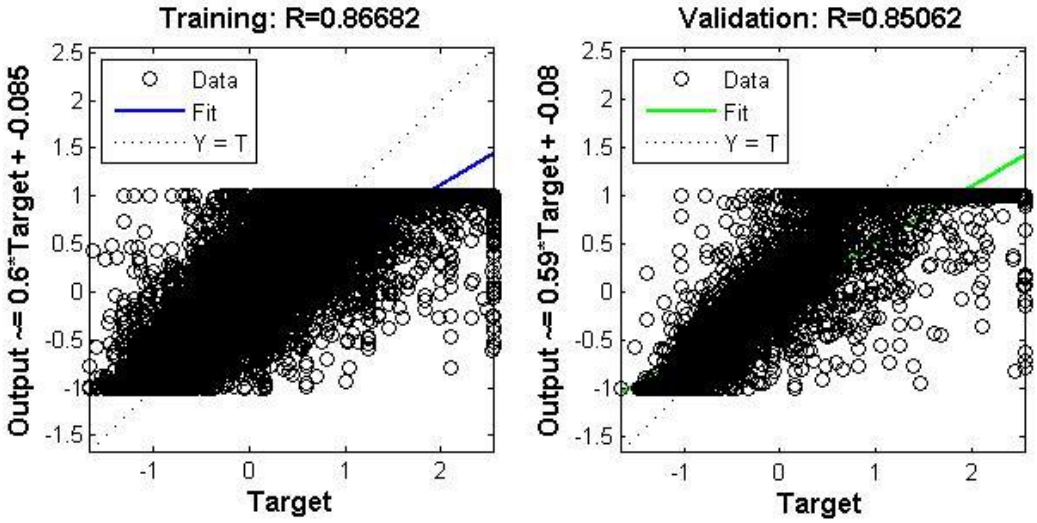


Figure 53: (a) Training Regression (b) Validation Regression for a neural network with 4 layer with hidden configuration of [20 20 20 20]

RMSE
6.363598399181195e+04

Number of layers = 5 and the hidden network configuration is [20 20 20 20 20]

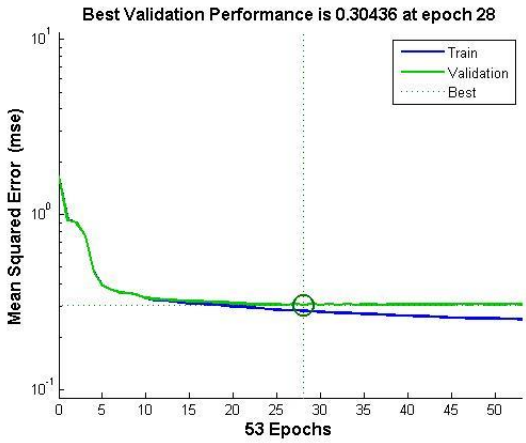


Figure 54: Performance Measure Training and Validation Error vs No. of Epochs for a neural network with 5 layer with hidden configuration of [20 20 20 20 20]

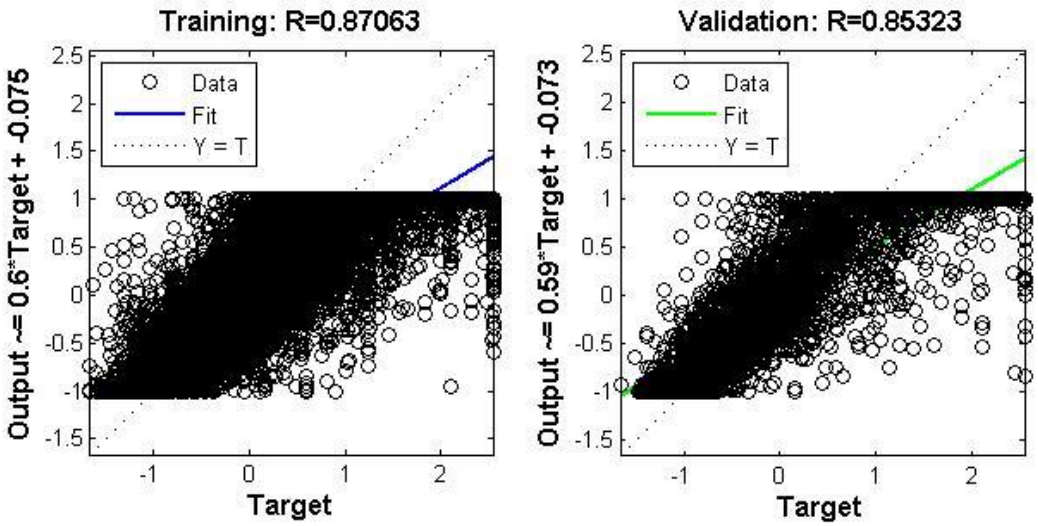


Figure 55: (a) Training Regression (b) Validation Regression for a neural network with 5 layer with hidden configuration of [20 20 20 20 20]

RMSE
63671.5470121589

Analysis:

On increasing the number of layers and the number of neurons in each layer, the performance of the neural network in terms of the reduction of the testing error of the system initially decreases but then the testing error starts increasing.

The idea behind increasing the number of layers in the neural network is to increase the number till the performance of the neural network stops to increase and starts decreasing. Thus, keeping the number of neurons for each layer constant and increasing the number of layers till the error stops decreasing and then we can vary the number of neurons in each layer once the optimum number of layers is found.

6. Training Function – Gradient Descent

Since in the Lavenberg-Marquardt Algorithms and the Bayesian Regularization training algorithms, the training rate parameter cannot be controlled, testing that parameter with the gradient descent learning rule.

The following settings were used to perform the experiments under this category,

Settings:

1. Max_fail = 25
2. Maximum number of epochs = 200
3. Minimum Gradient = 1e-20
4. Training Data Split = 70:30
5. Learning Rate = 0.0001
6. Number of layers = 1
7. Training Method - traingd

The following experiments were performed under this category,

1. Number of hidden layers = 1 with configuration[10] and Learning rate of 0.05
2. Number of hidden layers = 1 with configuration[10] and Learning rate of 0.26
3. Number of hidden layers = 1 with configuration[10] and Learning rate of 0.51

Number of hidden layers = 1 and Configuration of hidden layer [10] and learning rate = 0.05

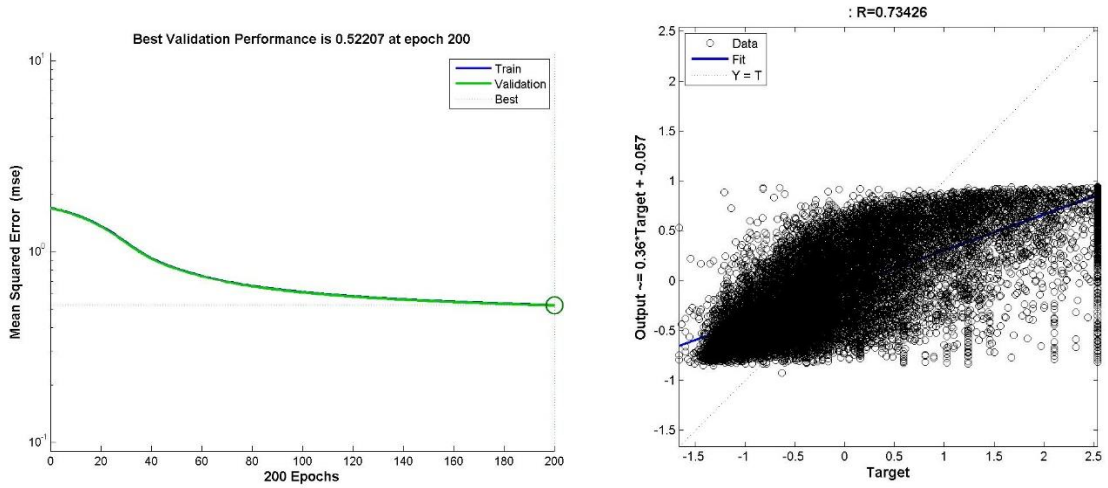


Figure 56: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.01

RMSE
8.686121737980029e+04

Number of hidden layers = 1 and Configuration of hidden layer [10] and learning rate = 0.26

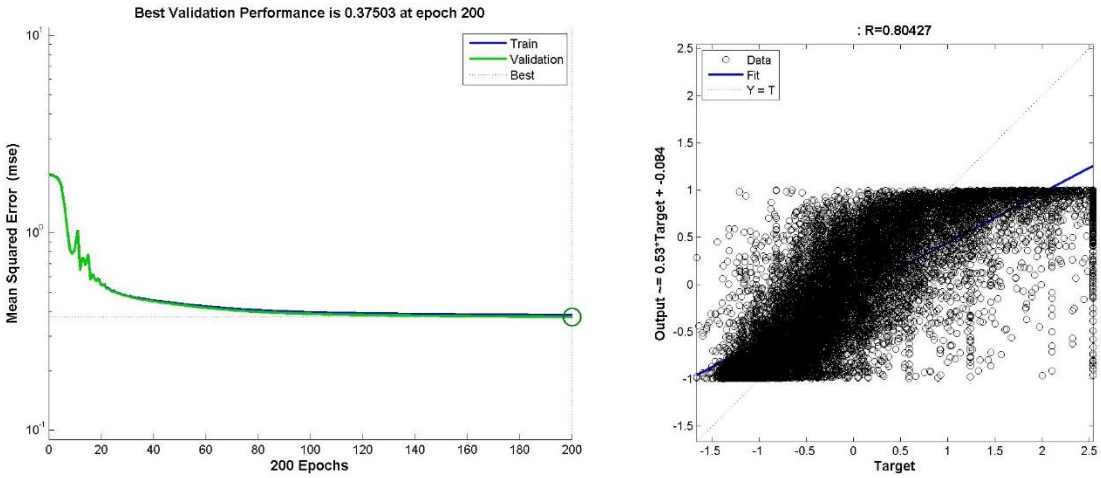


Figure 57: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.26

RMSE
7.266705573430365e+04

Number of hidden layers = 1 and Configuration of hidden layer [10] and learning rate = 0.51

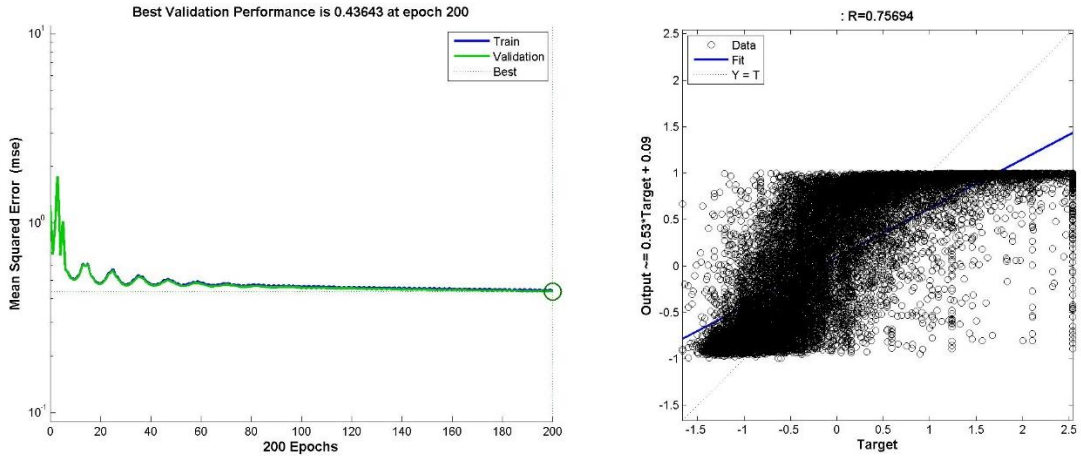


Figure 58: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.51

Misclassification error
7.280546096589715e+04

Number of hidden layers = 1 and Configuration of hidden layer [10] and learning rate = 0.76

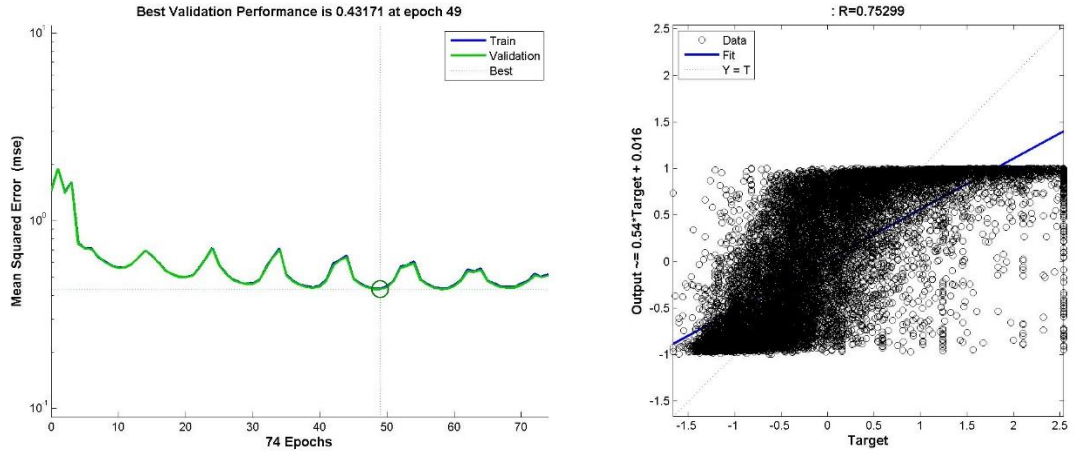


Figure 59: Training Error vs No. of Epochs for Neural Network with 1 hidden layer with 10 neurons and Gradient Descent Learning with learning rate of 0.76

RMSE
7.368971902721983e+04

Analysis:

When the value of learning rate is very low, then the network does not learn much within the maximum number of specified epochs. On increasing the value of learning rate the network starts learning quickly and might reach the optimum value within the specified number of epochs. But

however if the value of the learning rate is too high then the function might oscillate back and forth between the optimum value of the cost function.

Thus it is very important to choose the learning rate appropriately so that the function reaches the optimum value.

Radial Basis Function

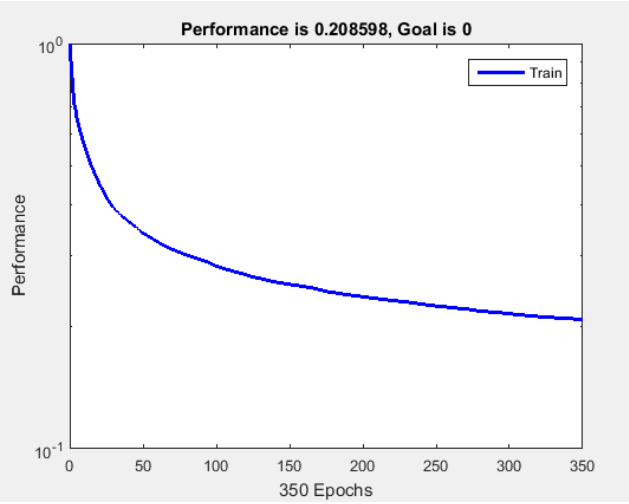


Figure 60: Training Error of a Radial Basis Function

Training Error

Table 1: Training error vs No of neurons of the radial basis function

Number of Neurons	Mean Square Error
50	0.340835
100	0.282458
150	0.254537
200	0.237301
250	0.225163
300	0.215373
350	0.208598

The number of neurons were varied and the corresponding training error was noted

Testing Error

Table 2: Testing error of the corresponding trained neural network model

Number of Neurons	Root Mean Square Error
50	6.7773e+04
100	6.1951e+04
150	5.9938e+04
200	5.8494e+04

The radial basis function works as explained in the first part of this report. 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.