CS6011: Kernel Methods for Pattern Analysis **Even 2015**

Assignment II Report

15/03/2015

Charu Chauhan, CS14M058
Devi G, CS14M011
Ditty Mathew, CS13D018

1 Objective

The objective of this assignment is to understand the application of MLFFNN in both classification and regression tasks.

1. Under classification tasks, the performance of MLFFNN is studied in following 4 cases
   (a) **Linearly separable classes** Here, the performance of MLFFNN with 1 or 2 hidden layers is compared with Bayes classifier using GMM model and perceptron. Number of classes is 3.
   (b) **Non-linearly separable classes** Here, the performance of MLFFNN with 1 or 2 hidden layers is compared with Bayes classifier using GMM model. Number of classes is 2.
   (c) **Overlapping classes** Here, the performance of MLFFNN with 1 or 2 hidden layers is compared with Bayes classifier using GMM model. Number of classes is 3.
   (d) **Image data** Here, the performance of MLFFNN with 2 hidden layers is compared with Bayes classifier using GMM model and perceptron. Number of classes is 5.
2. Under regression tasks, function approximation of Univariate and Bivariate dataset are experimented with MLFFNN and RBF models.
   (a) To find the best performing MLFFNN model with 1 hidden layers for the Univariate dataset
   (b) To find the best performing MLFFNN model with 1 or 2 hidden layers for the Bivariate dataset
   (c) To find the best performing RBF model for both Univariate and Bivariate datasets and to analyse the effect of regularisation parameter and model complexity on the performance of RBF model.

2 Classification tasks

2.1 Linearly separable classes

The given data includes 3 linearly separable classes. Visualisation of the training data is as in Figure 1

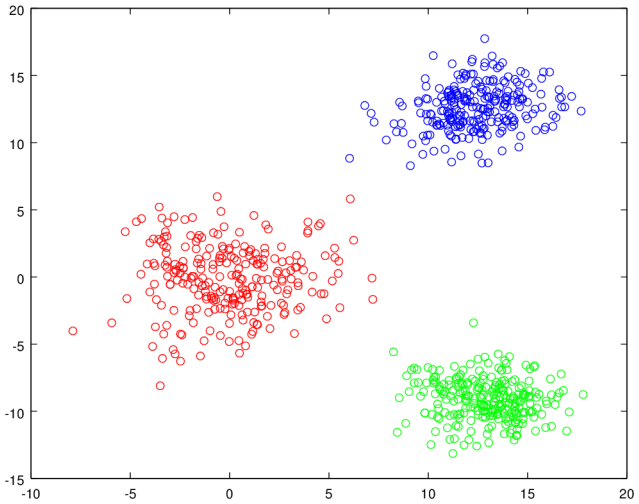


Figure 1: Linearly separable classes

2.1.1 Bayes classifier using Gaussian model

Bayes classifier involves Bayesian reasoning which provides a probabilistic approach to inference. It is based on certain assumptions like the underlying data distribution follows a particular probability distribution and enables to make optimal decisions in classification tasks by making use of the assumed probabilities together with observed data. As per the bayes rule,

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}$$

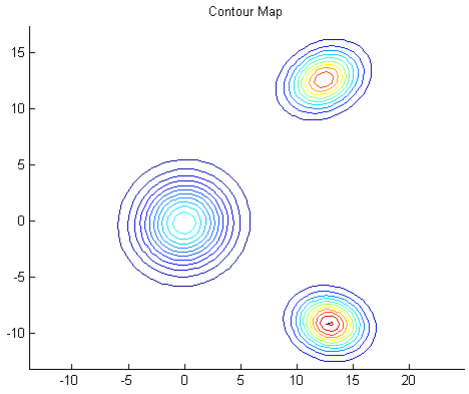
The above formula calculates the posterior probability that a particular point $\bar{x}$ belongs to class ω_j. We will assign $\bar{x}$ to the class j whose posterior probability is the maximum.

Coming on to the right hand side of this equation, $P(\omega_j)$ is same for all classes in our data set because each class has same number of training examples and we do not know anything about the real world prior distribution.Also, denominator is ignored as it is same for all classes for a particular value of x, our decision is solely dependent on $p(x|\omega_j)$.

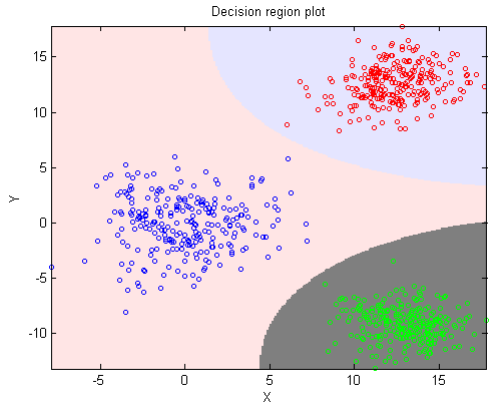
For this assignment, we have considered gaussian distribution function for deciding $p(x|\omega_j)$. This function is dependent on mean and covariance of the data and is decided based on that. This function is given by

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

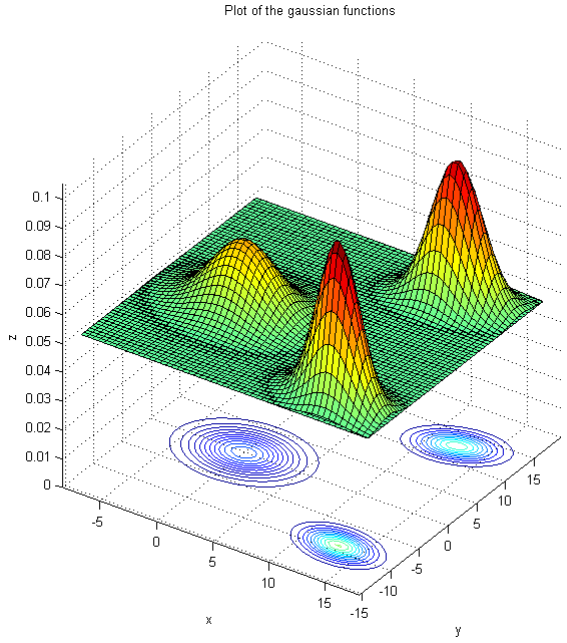
In order to deal with calculations on very small values of probabilities, we take the log of likelihood function and then calculate the probabilities.



(a) Contour map



(b) Decision region plot



(c) Gaussian plot

Figure 2: Bayes classifier using Gaussian model for Linearly separable classes

The confusion matrix for both validation and test data are shown below:

	class1	class2	class3
class1	150	0	0
class2	0	150	0
class3	0	0	150

Table 1: Validation data

	class1	class2	class3
class1	100	0	0
class2	0	100	0
class3	0	0	100

Table 2: Test data

Inferences

- One of the three classes has a circular contour which suggests independence of features in that class i.e. diagonal covariance matrix with all variances same
- 3 gaussians are enough to classify the data (one for each class)
- the means of the three classes are well separated from each other
- the covariance matrices of the three classes are different implies a non-linear decision boundary

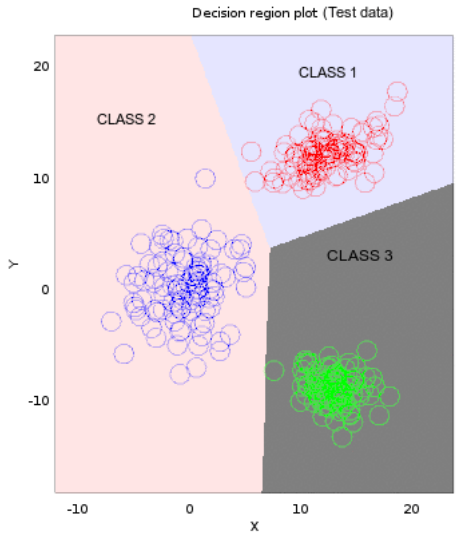
2.1.2 Perceptron

The perceptron is a type of artificial neural networks. It can be seen as the simplest kind of feedforward neural network: a linear classifier. We experimented with both pattern mode of learning and batch mode of learning.

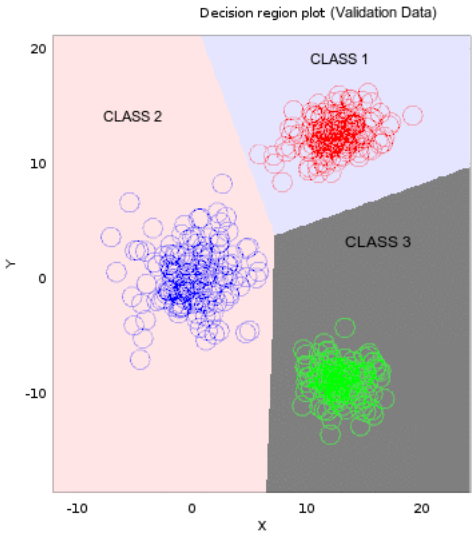
Pattern mode

- Initialize the weight vector w$< w_0, w_1 ..... >$ to 0.
- Repeat
- Alter the weights for every misclassified point in training data.
  w(m+1) = w(m) + η*y*x(m)
- till all the points are properly classified.

We trained the perceptron on the training data provided to us with $\eta = 0.1$.The decision region plots for the validation and test data is given below.



(a) Decision region plot for Test Data



(b) Decision region plot for Validation data

Figure 3: Classification using Perceptron

The confusion matrix for the test data and validation data is given below:

	A	B	C
A	150	0	0
B	0	150	0
C	0	0	150

(a) Validation Data

	A	B	C
A	100	0	0
B	0	100	0
C	0	0	100

(b) Test data

Figure 4: Confusion matrix

The accuracy of the learned model is 100%.

Batch Mode Learning:

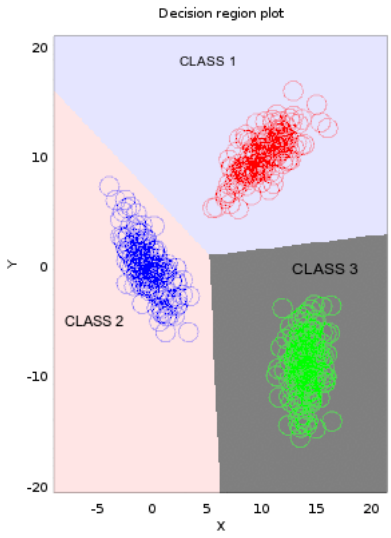
In this variant of perceptron learning we scan all of the training examples once and then update the weight at the end one epoch.Cumulative changes of all misclassified examples is done at the end.

Algorithm:

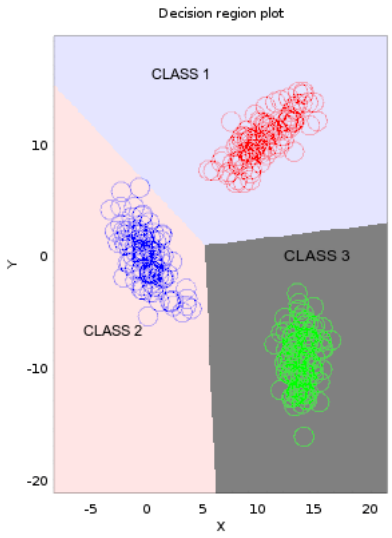
- Initialize the wight vector w to 0.
- Repeat for each epoch
- $\triangle w(l) = 0$
- Repeat for each training data sample
- If the sample is misclassified
- $\triangle w(l) = \triangle w(l) + x*y$

- Endif
- w(l+1) = w(l) + η * $\triangle$w(l)
- End epoch

The perceptron weights were learned with $\eta = 0.1$.The decision plots for the test data and the validation data after learning perceptron in batch mode are:



(a) Validation Data



(b) Test data

Figure 5: Decision region plots

Confusion matrix for the test and train data is as follows:

	A	B	C
A	100	0	0
B	0	100	0
C	0	0	100

(a) Test data

	A	B	C
A	150	0	0
B	0	150	0
C	0	0	150

(b) Validation Data

Figure 6: Confusion matrix

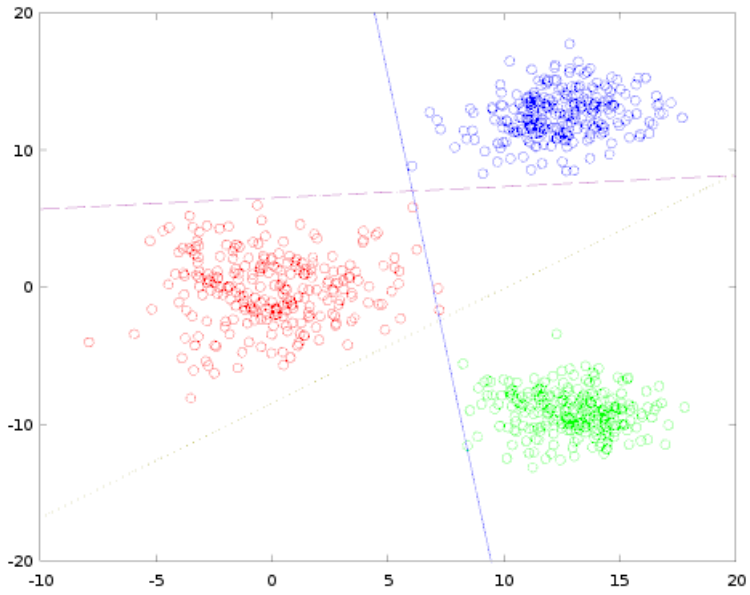
The accuracy of the learned model is 100%.

Learning Rate	**Epocs Class A**	**Epocs Class B**	**Epocs Class C**	**Accuracy Testdata**
0.1	500	8	5	100
0.2	500	3	6	99.67
0.3	500	18	8	100
0.4	500	7	16	100
0.5	500	15	19	100
0.6	500	16	17	100
0.7	500	9	9	100
0.8	500	8	8	100
0.9	500	9	8	99.67

Table 3: Learning rate analysis with respect to number of epochs for pattern mode

Learning Rate	**Epocs Class A**	**Epocs Class B**	**Epocs Class C**	**Accuracy Testdata**
0.1	500	34	31	99.67
0.2	500	34	31	100
0.3	500	8	5	100
0.4	500	8	5	99.67
0.5	500	34	31	99.67
0.6	500	34	31	99.67
0.7	500	34	31	99.67
0.8	500	34	31	99.67
0.9	500	34	31	100

Table 4: Learning rate analysis with respect to number of epochs for batch mode



(a) Separating line obtained by the perceptron learn on training data

Inferences

- Batch mode takes more number of epochs to converge
- In pattern mode, we can observe that the perceptron converges faster with increase in learning rate
- The perceptron that is learned for the class 1 Vs rest is not converging even at 500 epochs.This is because there is one data point in training set of Class 1 that lies on the boundary.Every time we alter the weights near the separating line,atleast one sample is misclassified .

2.1.3 MLFFNN

Multilayer feedforward neural network consists of multiple hidden layers which are connected in feedforward way. There are three main layers - input layers, hidden layers and output layers. The number of hidden layers decides the complexity of the decision surface.

We used gradient descent method to train linearly separable data. The data is trained with one hidden layer. The number of hidden nodes are chosen by cross validation. The plot for the mean square error versus number of hidden nodes is shown in Figure 7.

Inferences

- Minimum MSE is obtained for 3 hidden nodes
- Fixed the number of hidden nodes as 3

In order to choose the learning rate, we plotted the MSE versus number of epocs for different learning rates. The Figure 8 depicts the plots for learning rate 0.1 and 0.9. For these plots we used hyperbolic tangent activation function. The plot for logistic sigmoid activation function is shown in Figure 9.

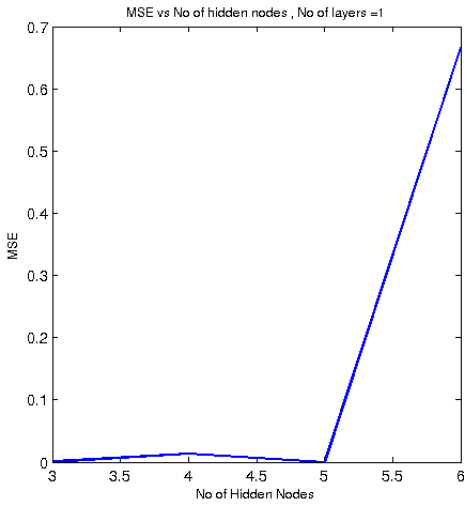
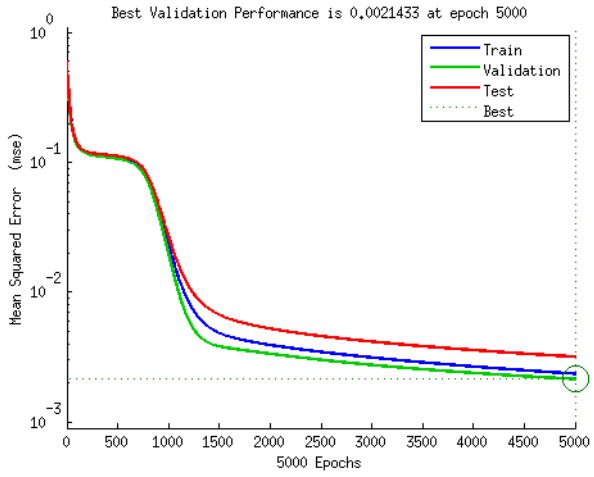
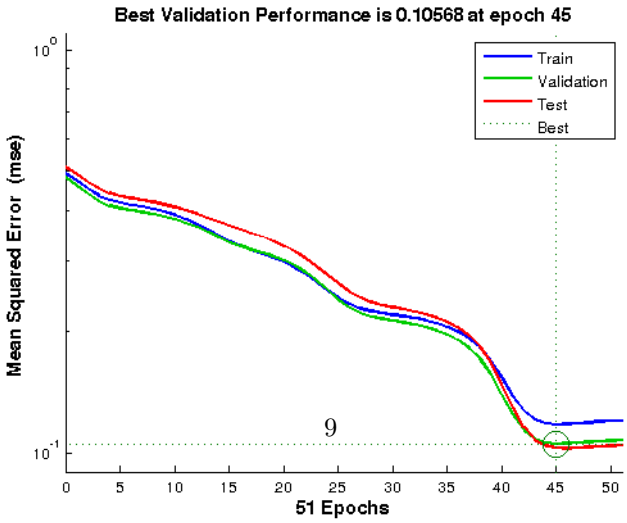


Figure 7: Linearly separable classes



(a) Learning Rate =0.1



(b) Learning Rate =0.9

Figure 8: Learning Rate analysis for hyperbolic tangent activation function

Inferences

- No oscillations observed for learning rate=0.1
- At learning rate =0.9, we can see oscillations
- Minimum MSE achieved at learning rate =0.1

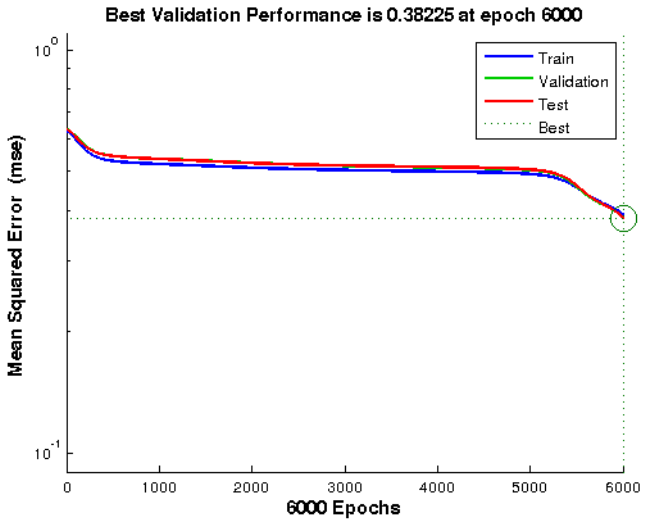


Figure 9: At Learning Rate =0.1 for Logistic activation function

Inferences

- MSE is more compared to hyperbolic tangent activation results
- Hyperbolic tangent function is faster than sigmoid function

Finally we chose the model with one hidden layer and 3 hidden nodes, hyperbolic tangent activation function and learning rate as 0.1. The network view is shown in Figure 10.

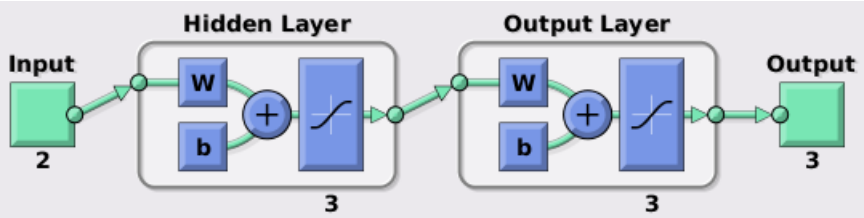


Figure 10: Neural Network View

The decision plot obtained for the given data using the mentioned model parameters is shown in Figure 11.

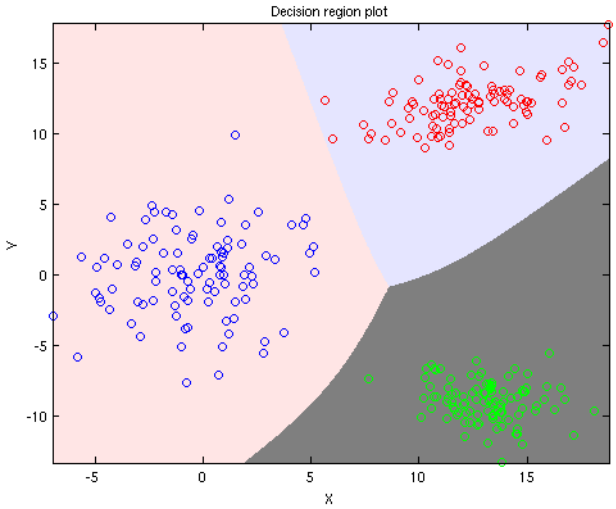
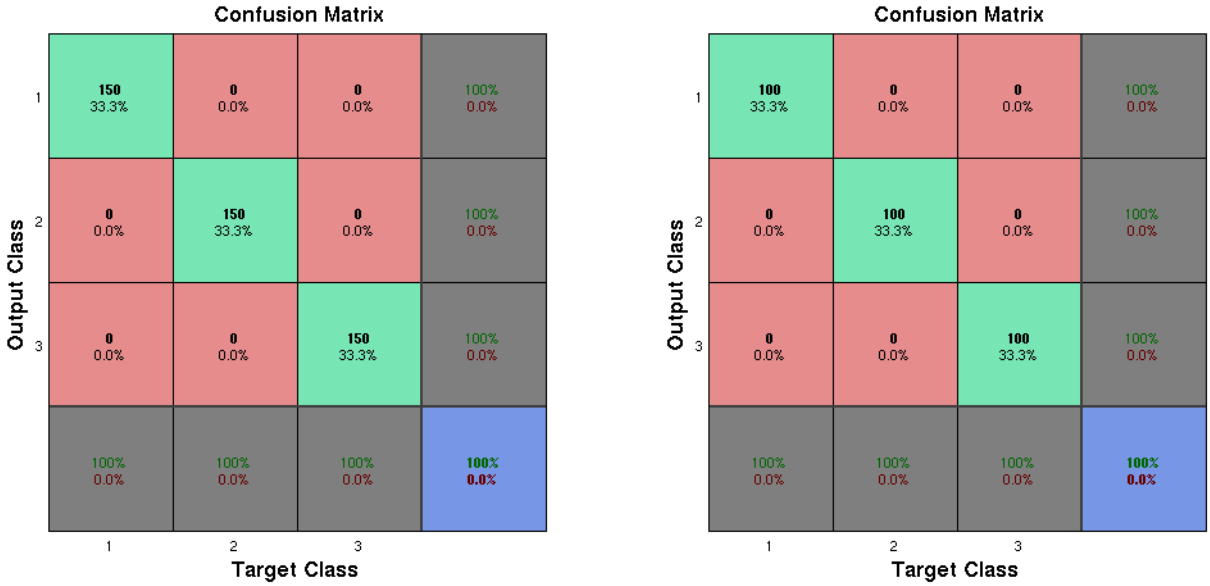


Figure 11: Decision region plot

The model gave 100% accuracy. The corresponding confusion matrix for validation data and test data are shown in Figure 12.



(a) Validation data

(b) Test data

Figure 12: Confusion Matrix

Inferences

- 100% accuracy obtained for both validation data and test data.

2.1.4 Comparison of performance of different models

Accuracy Comparison across different models

Method	Validation Accuracy (%)	Test Accuracy (%)
Bayes Classifier	100	100
Perceptron	100	100
MLFFN 1 hidden layer	100	100

Inferences

- All models are working well for linearly separable data

2.2 Non-linearly separable classes

The given data includes 2 linearly separable classes. Visualisation of the training data is as in Figure 13

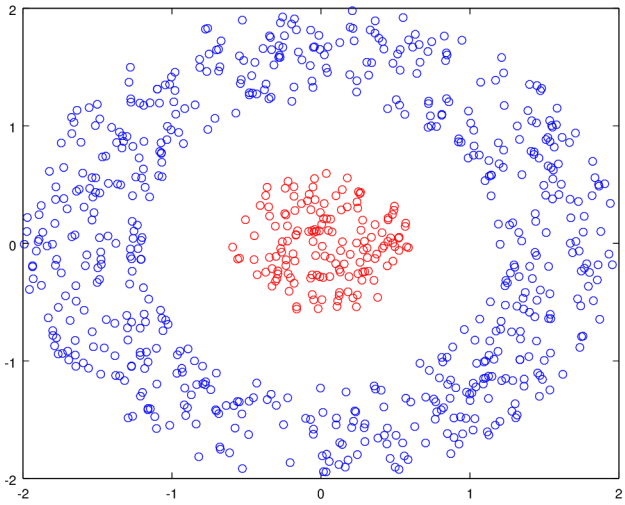
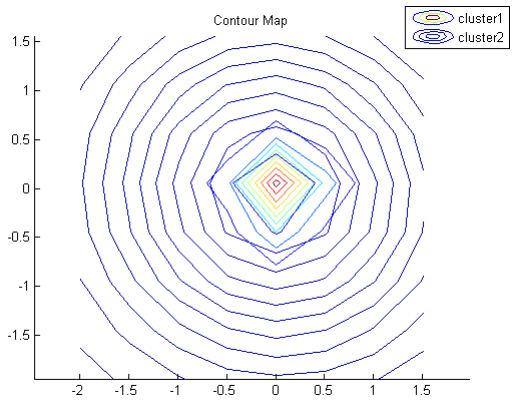
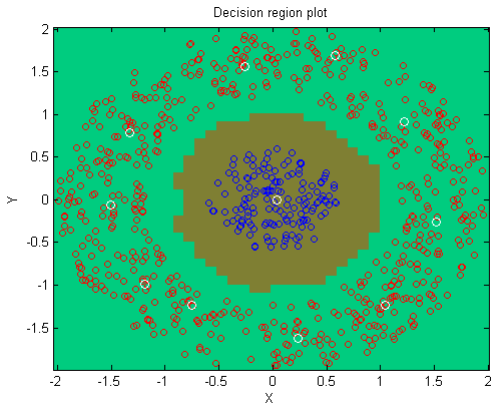


Figure 13: Non-linearly separable classes

2.2.1 Bayes classifier using GMM and Gaussian



(a) Contour map of GMM with 1 gaussian for each class



(b) Decision region plot of GMM with 10 gaussians for class 1 and 1 for class 2

Figure 14: Bayes classifier using GMM for Non-linearly separable classes

The confusion matrices for validation and testing data are as shown below:

	class1	class2
class1	90	0
class2	0	360

Table 5: Validation data

	class1	class2
class1	60	0
class2	0	240

Table 6: Test data

Inferences

- GMM using 10 clusters (as highlighted in the Figure 14) gives 100 percent accurracy.
- A single gaussian also gives 100% accuracy in this case. This is because of the contour of gaussian 1 is concentric to that of gaussian 2. However, this might not give good accuracy on all non linearly separable datasets.

2.2.2 MLFFNN

The data is trained using gradient descent method with one hidden layer. The number of hidden nodes are decided using cross validation The Figure 18 shows the plot of mean square error versus number of hidden nodes.

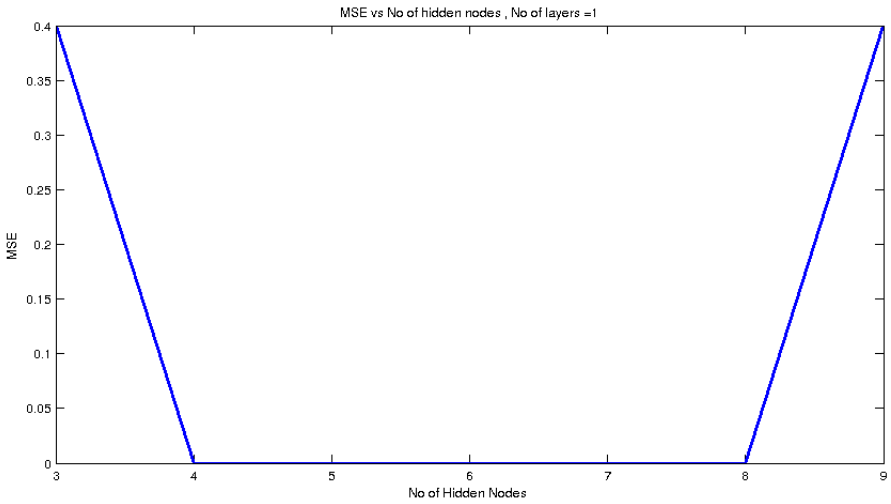


Figure 15: MSE vs No of Hidden nodes

Inferences

- Minimum mean square error obtained for 4 hidden layers

Then the number of epochs versus MSE is plotted for learning rate 0.1 and 0.9. The plots are shown in Figures 16 and 17.

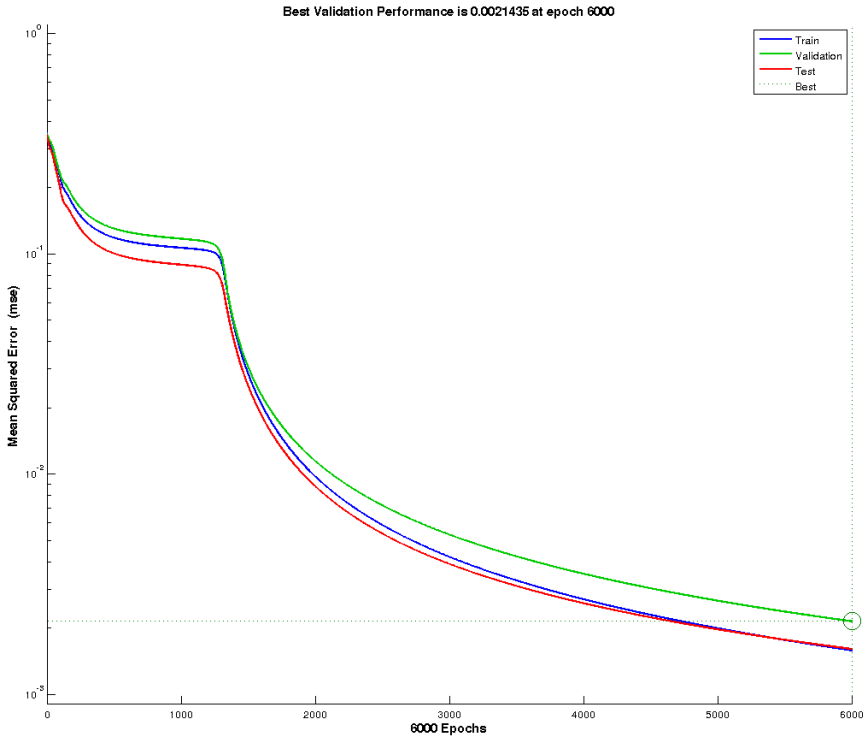


Figure 16: No of epochs vs MSE at learning rate =0.1

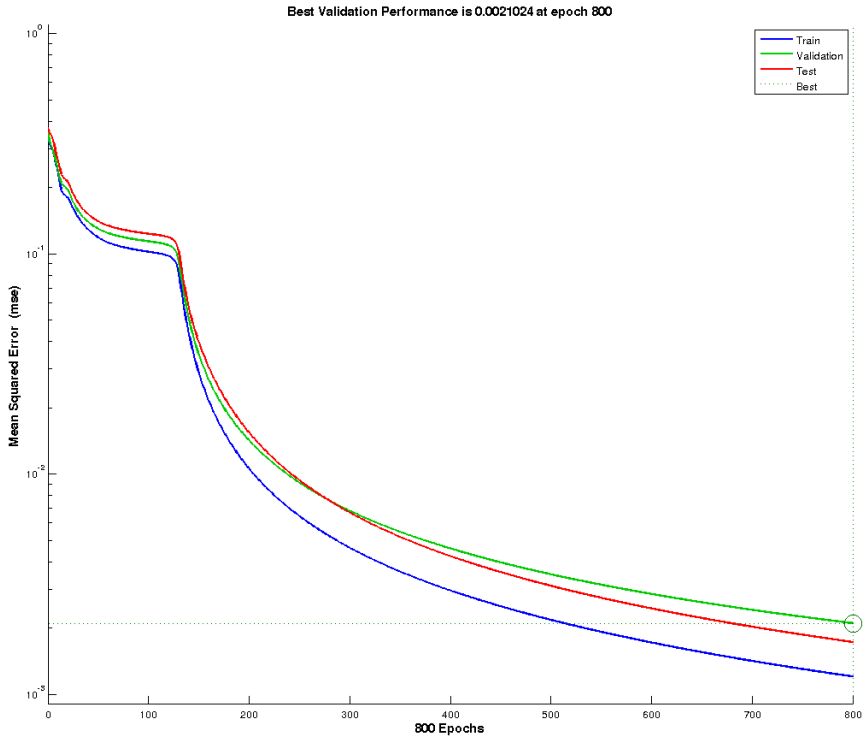


Figure 17: No of epochs vs MSE at learning rate =0.9

Inferences

- No oscillations are found in both the cases.
- Fixed learning rate as 0.1

We used hyperbolic tangent function as activation function in both hidden layer and output layer. The view of the network is shown in Figure 19.

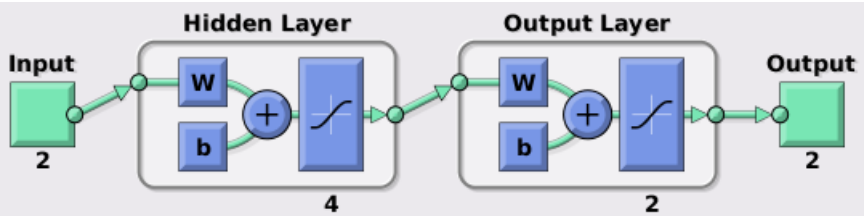


Figure 18: View of Neural network

The decision plot obtained for the given data using the mentioned model parameters is shown in Figure 19.

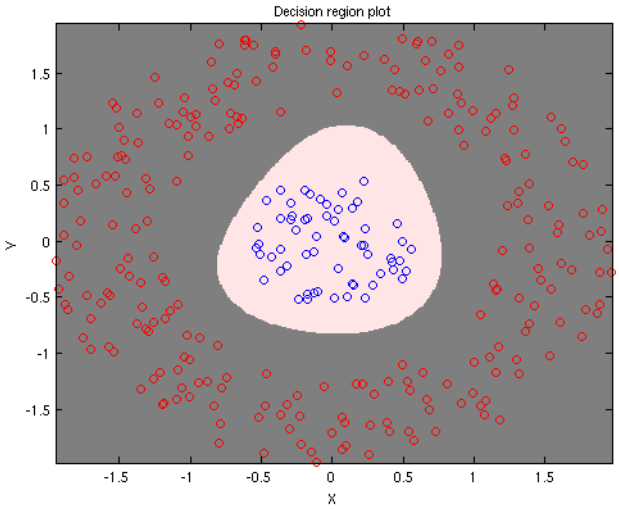
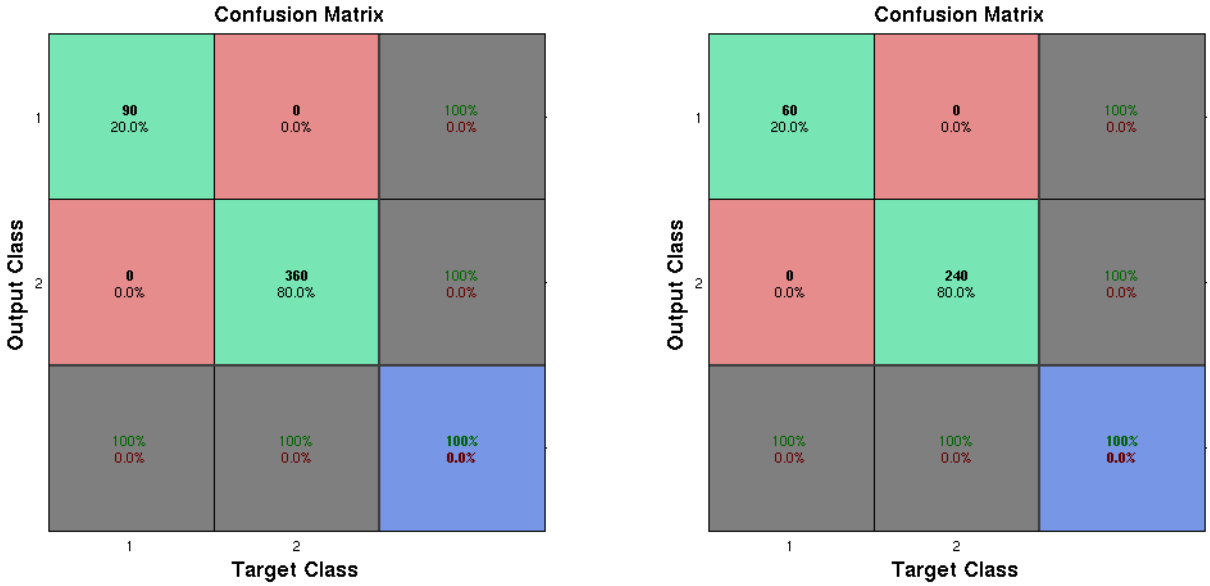


Figure 19: Decision region plot

The model gave 100% accuracy. The corresponding confusion matrix for validation data and test data are shown in Figure 20.



(a) Validation data

(b) Test data

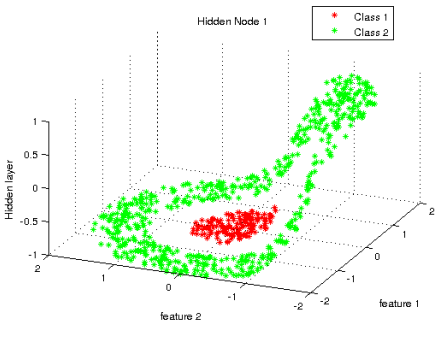
Figure 20: Confusion Matrix

Inferences

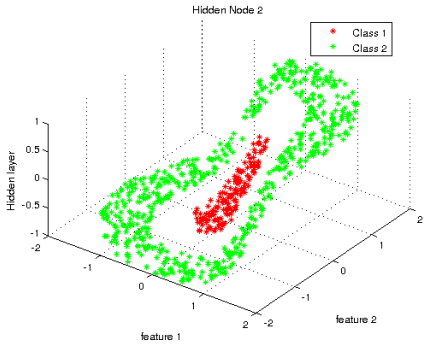
- 100% accuracy obtained for both validation data and test data.

2.2.3 Plots of Outputs of each of the hidden nodes and output nodes

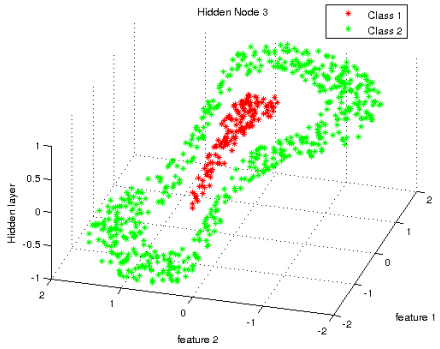
Using 1 hidden layer for no of epochs = 1



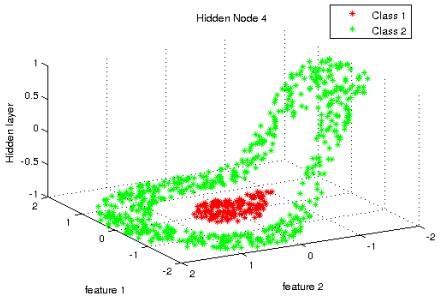
(a) Hidden node 1



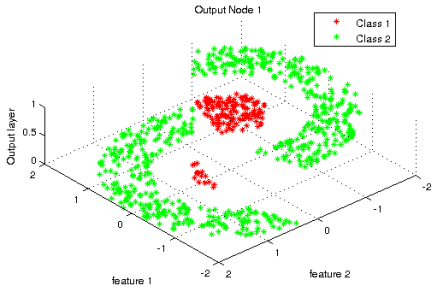
(b) Hidden node 2



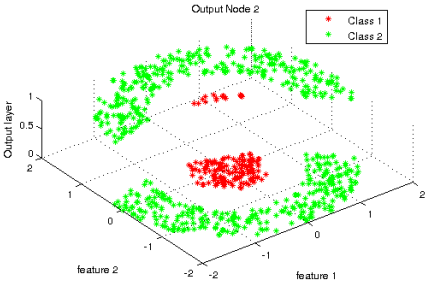
(c) Hidden node 3



(d) Hidden node 4



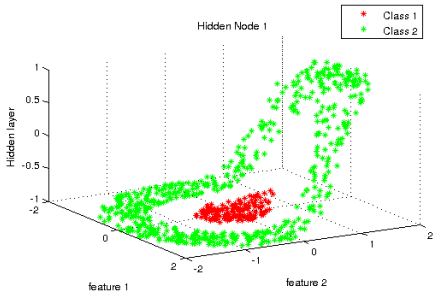
(e) Output node 1



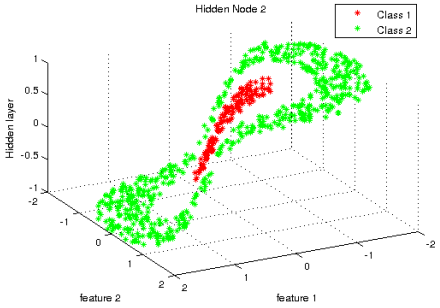
(f) Output node 2

Figure 21: No of Epochs =1

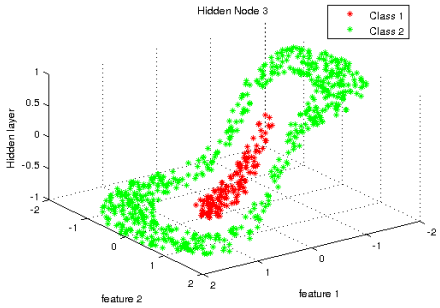
Using 1 hidden layer for no of epochs = 2



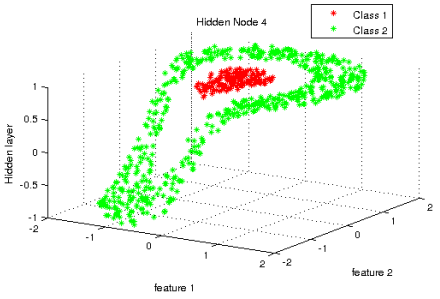
(a) Hidden node 1



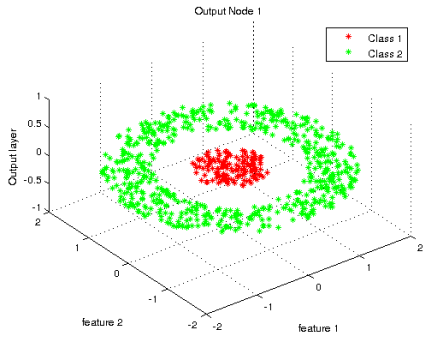
(b) Hidden node 2



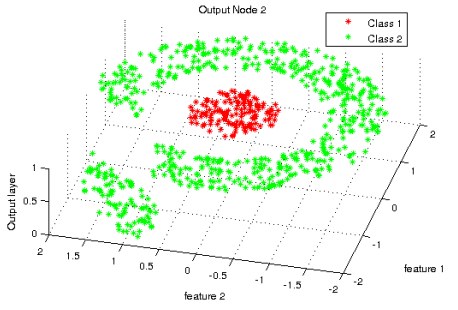
(c) Hidden node 3



(d) Hidden node 4



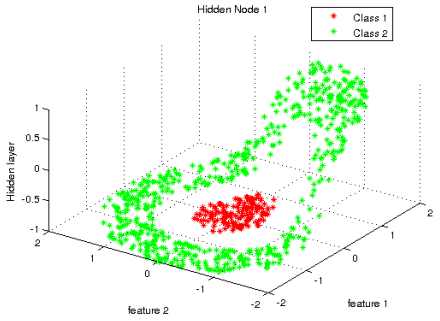
(e) Output node 1



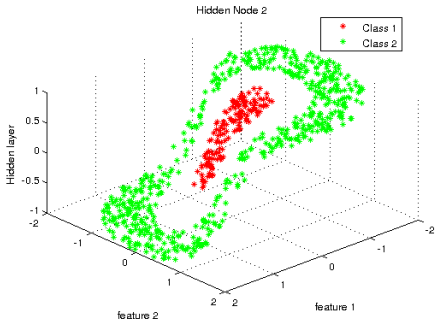
(f) Output node 2

Figure 22: No of Epochs =2

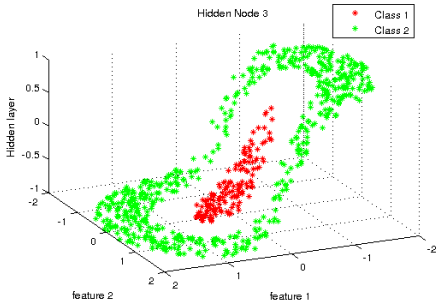
Using 1 hidden layer for no of epochs = 10



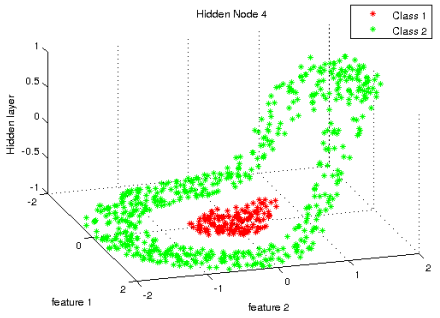
(a) Hidden node 1



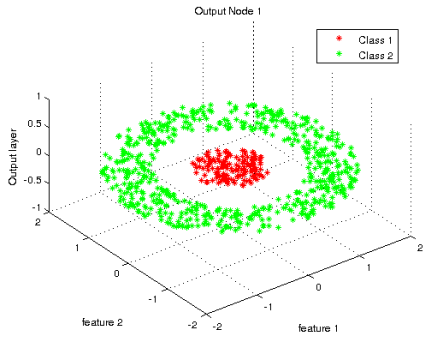
(b) Hidden node 2



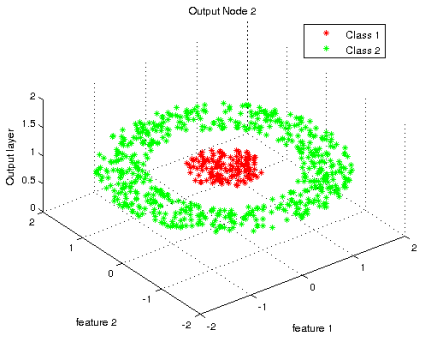
(c) Hidden node 3



(d) Hidden node 4



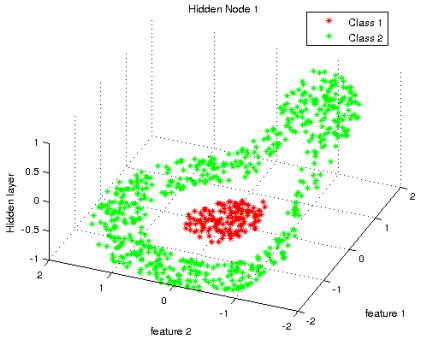
(e) Output node 1



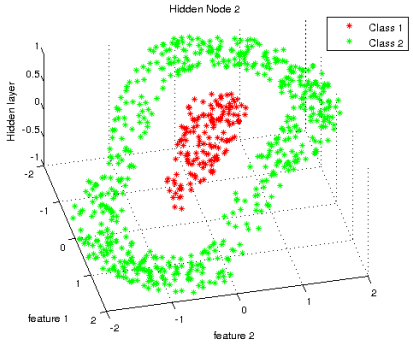
(f) Output node 2

Figure 23: No of Epochs =10

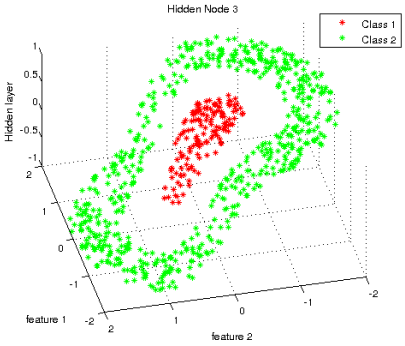
Using 1 hidden layer for no of epochs = 50



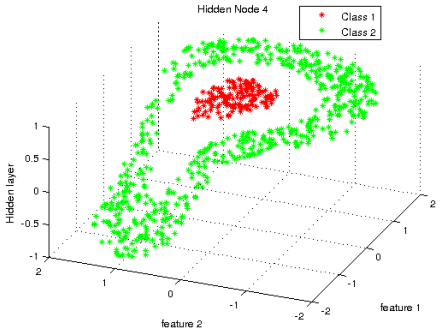
(a) Hidden node 1



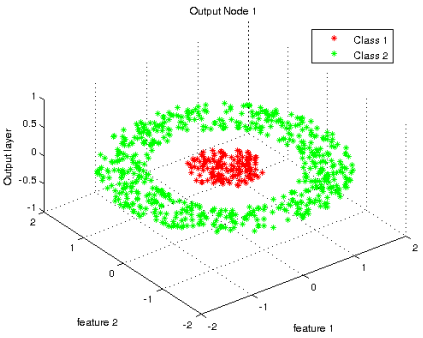
(b) Hidden node 2



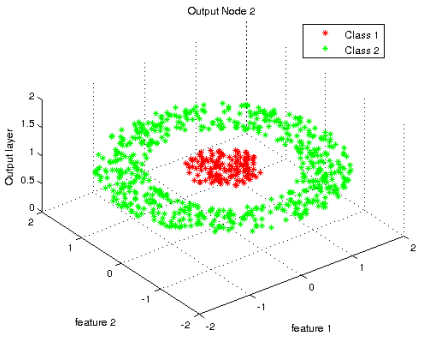
(c) Hidden node 3



(d) Hidden node 4



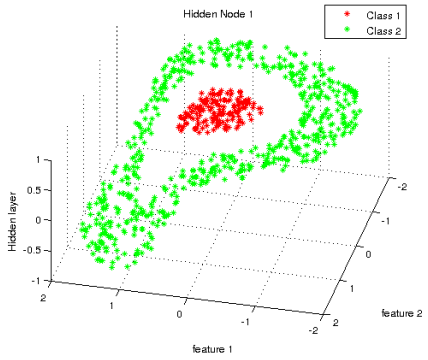
(e) Output node 1



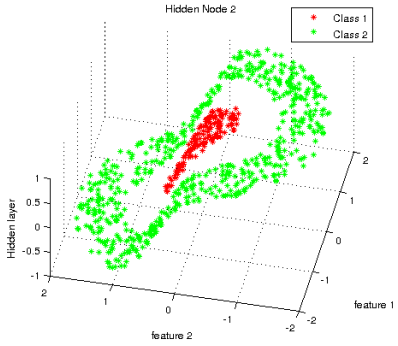
(f) Output node 2

Figure 24: No of Epochs =50

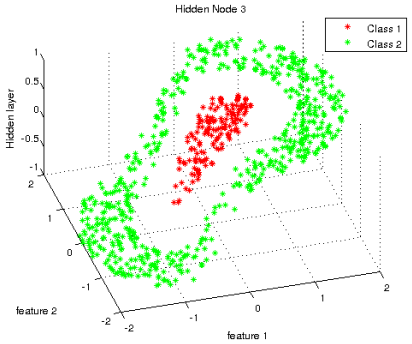
Using 1 hidden layer for no of epochs = 100



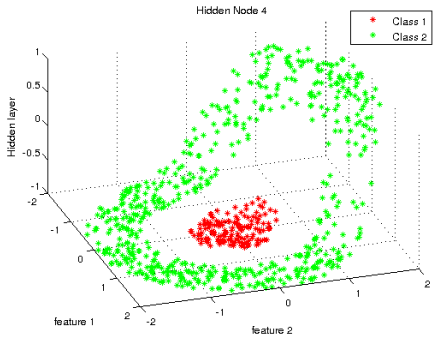
(a) Hidden node 1



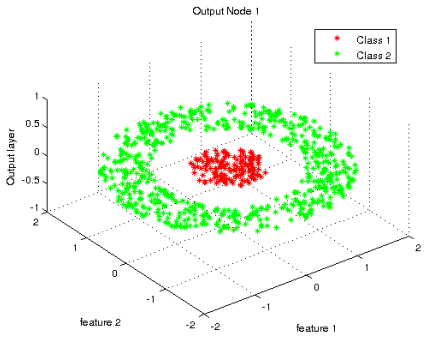
(b) Hidden node 2



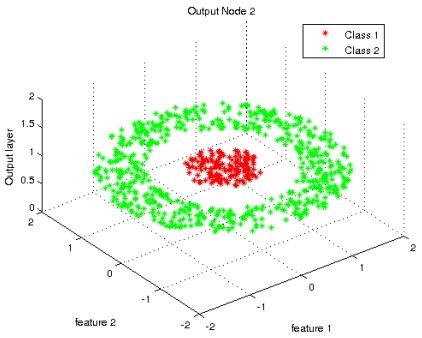
(c) Hidden node 3



(d) Hidden node 4



(e) Output node 1



(f) Output node 2

Figure 25: No of Epochs =100

Inferences

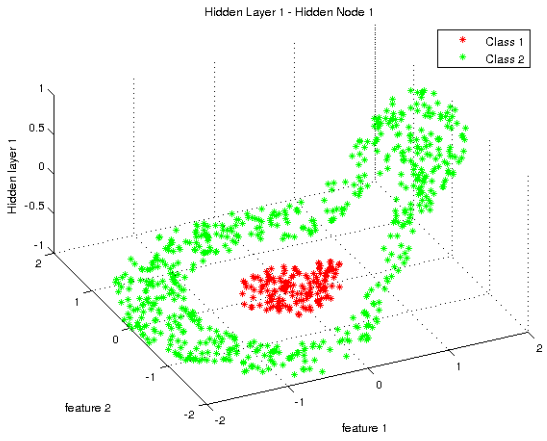
- Non linear tangent activation function results an S-shaped surface for each hidden node plots.
- From the output nodes of each epochs we can observe that as the number of epochs increases, the classification

accuracy is increasing

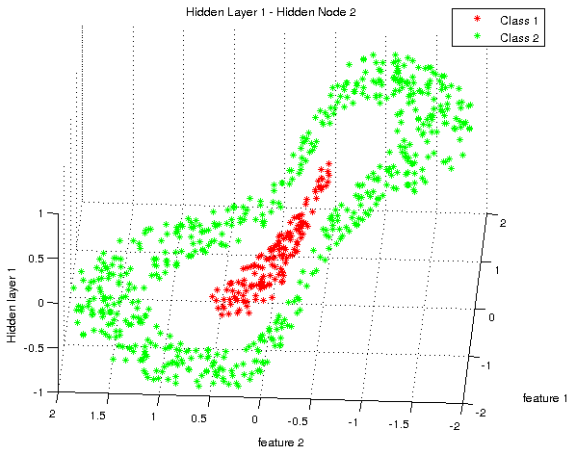
- At 100 epochs all data points are classified into one class(majority). This is due to the class imbalance in the given data.
- With more number of epochs(500), model classifies correctly

Using 2 hidden layer for no of epochs = 1

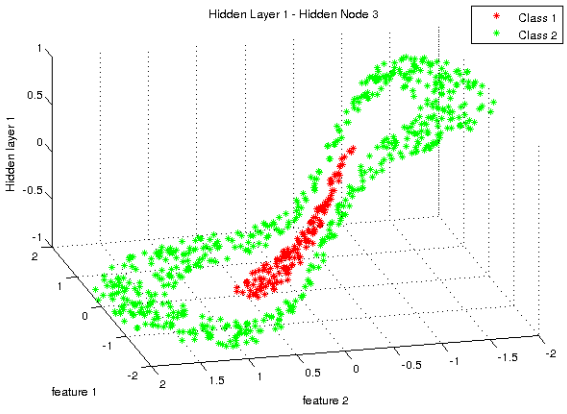
Our best model uses 4 hidden nodes in the first hidden layer and 3 hidden nodes in the second hidden layer.



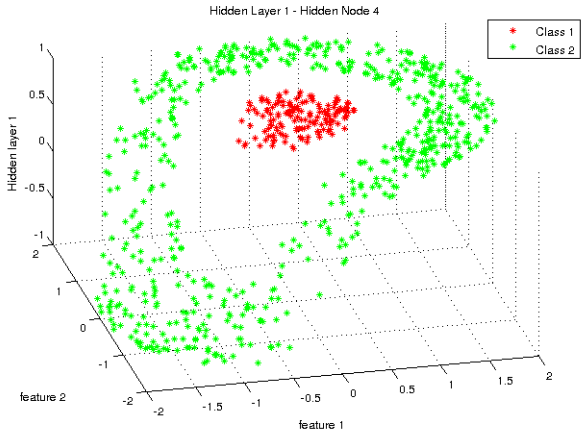
(a) Hidden layer 1, Hidden node 1



(b) Hidden layer 1, Hidden node 2

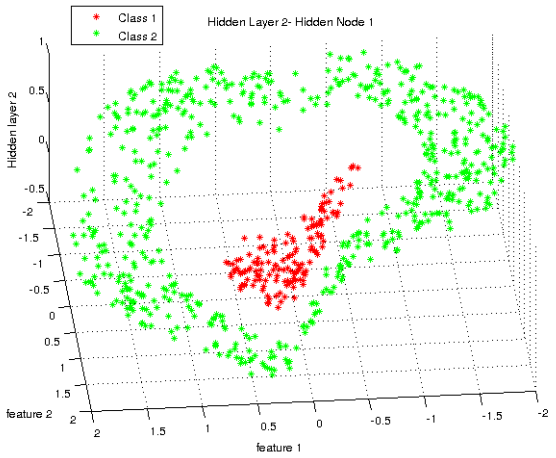


(c) Hidden layer 1, Hidden node 3

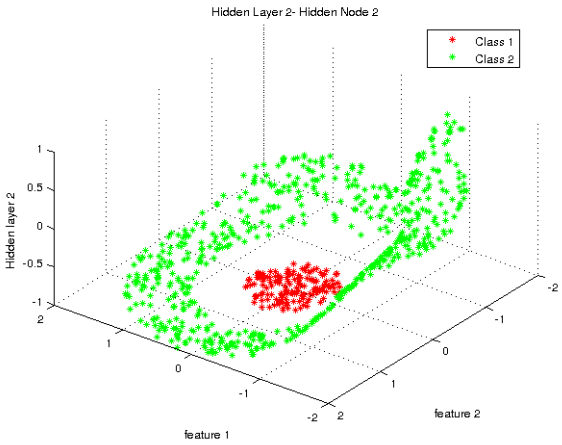


(d) Hidden layer 1, Hidden node 4

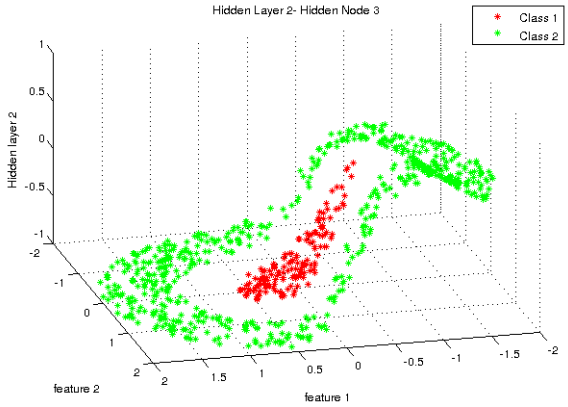
Figure 26: Hidden Nodes for No of Epochs =1



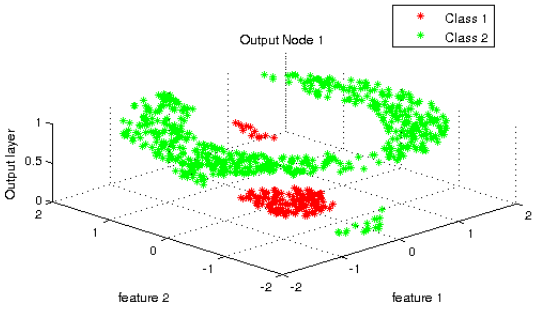
(a) Hidden layer 2, Hidden node 1



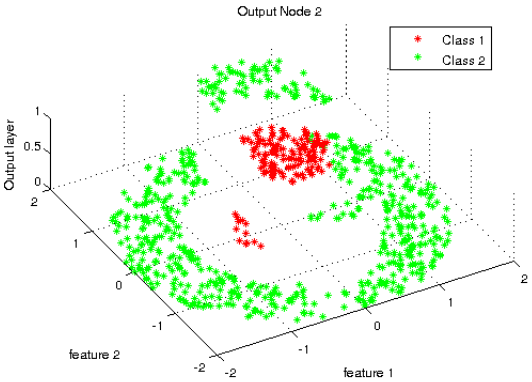
(b) Hidden layer 2, Hidden node 2



(c) Hidden layer 2, Hidden node 3



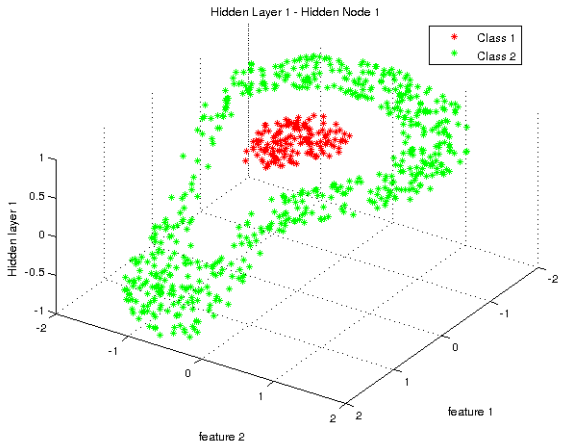
(d) Output node 1



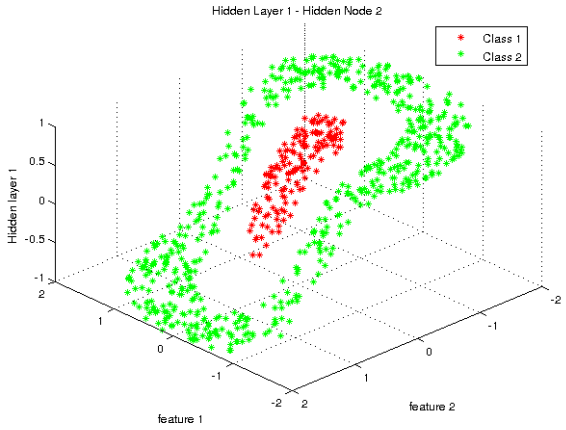
(e) Output node 2

Figure 27: Hidden and Output nodes for No of Epochs =1

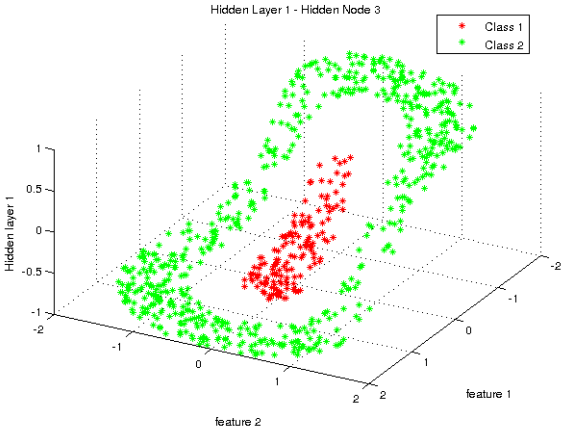
Using 2 hidden layer for no of epochs = 2



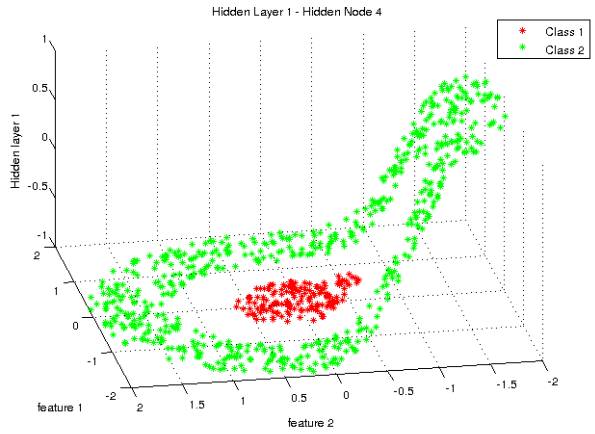
(a) Hidden layer 1, Hidden node 1



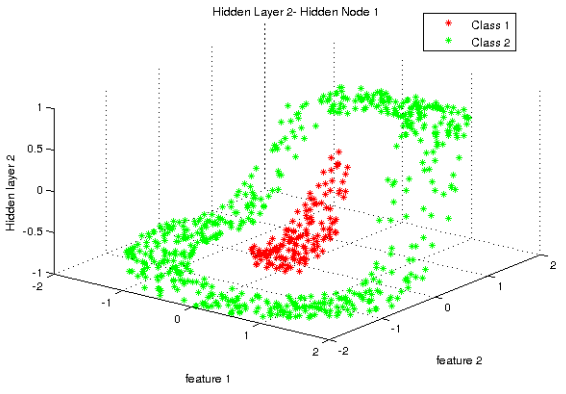
(b) Hidden layer 1, Hidden node 2



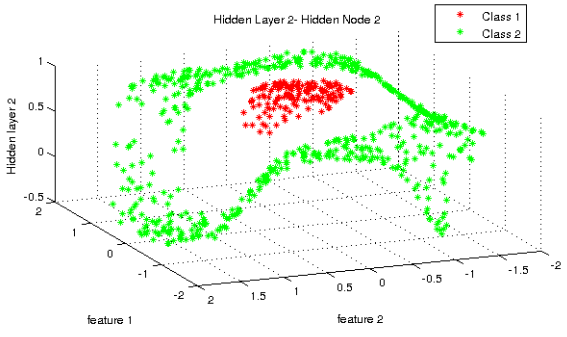
(c) Hidden layer 1, Hidden node 3



(d) Hidden layer 1, Hidden node 4

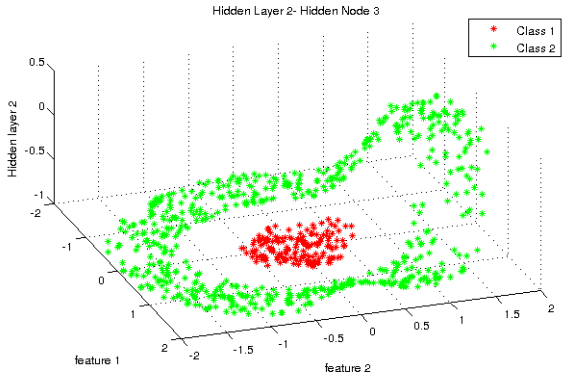


(e) Hidden layer 2, Hidden node 1

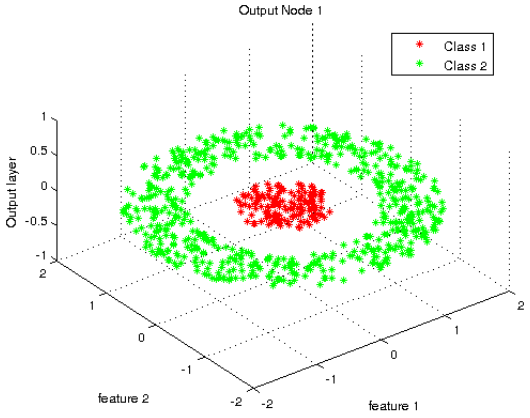


(f) Hidden layer 2, Hidden node 2

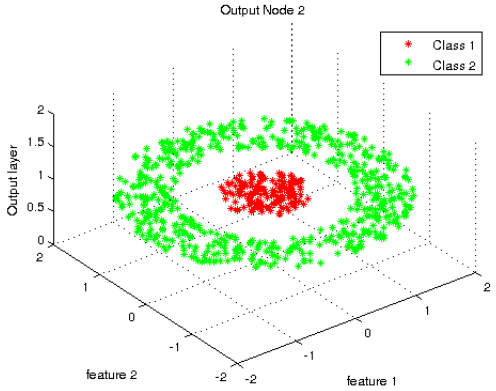
Figure 28: Hidden Nodes for No of Epochs =2



(a) Hidden layer 2, Hidden node 3



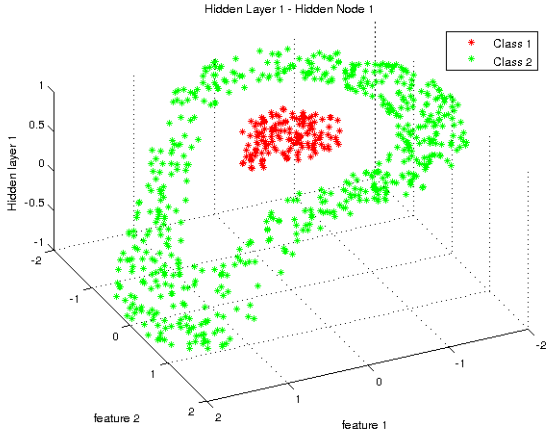
(b) Output node 1



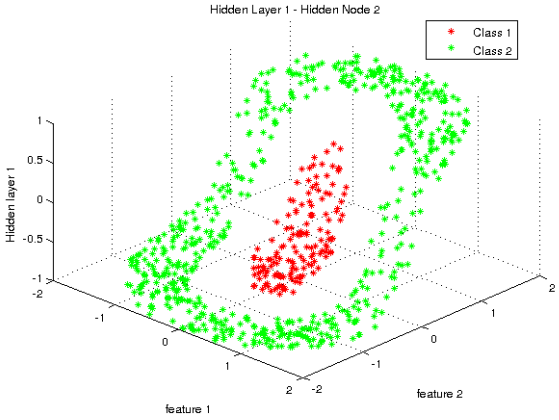
(c) Output node 2

Figure 29: Output nodes for No of Epochs =2

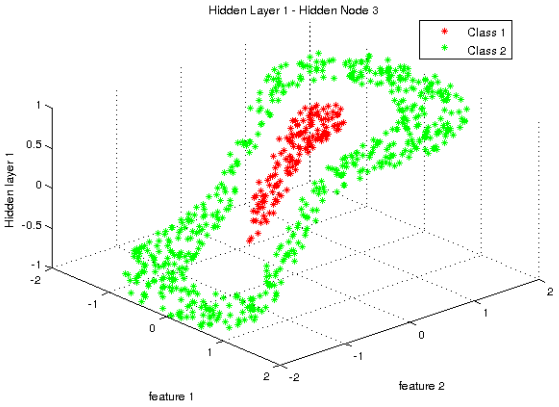
Using 2 hidden layer for no of epochs = 10



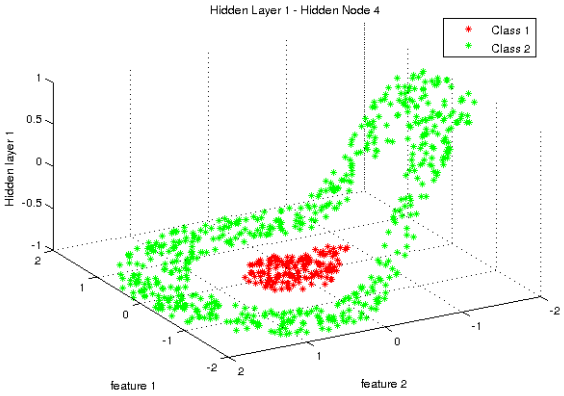
(a) Hidden layer 1, Hidden node 1



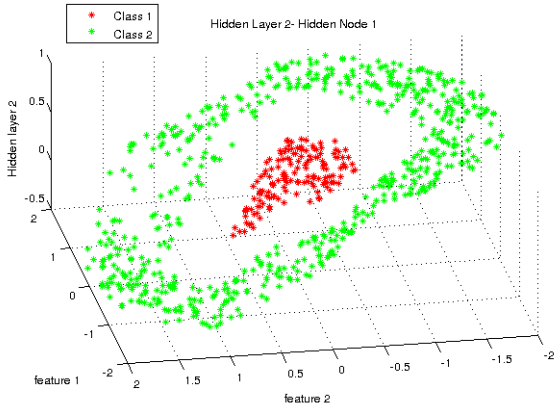
(b) Hidden layer 1, Hidden node 2



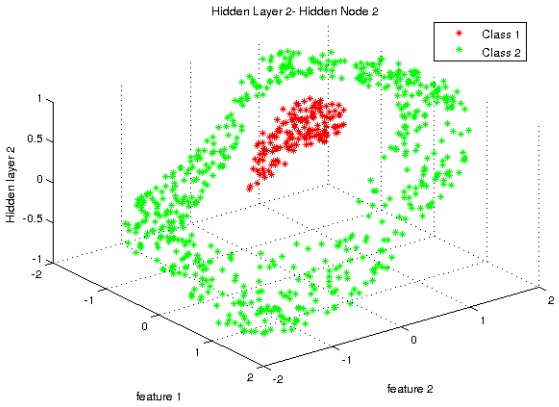
(c) Hidden layer 1, Hidden node 3



(d) Hidden layer 1, Hidden node 4

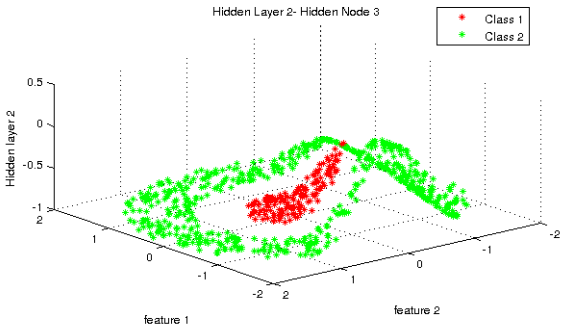


(e) Hidden layer 2, Hidden node 1

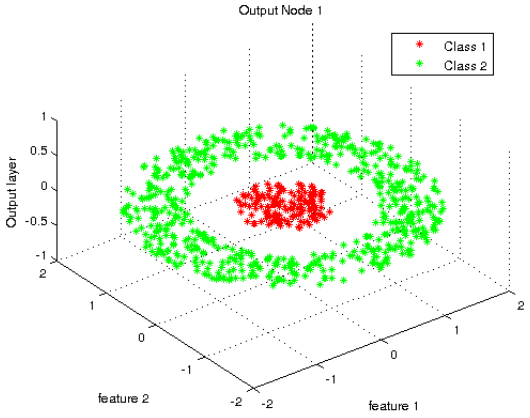


(f) Hidden layer 2, Hidden node 2

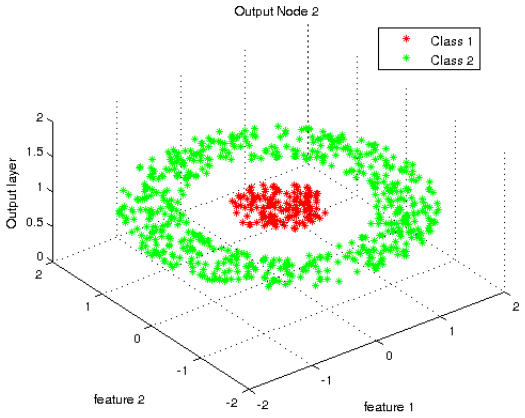
Figure 30: Hidden Nodes for No of Epochs =10



(a) Hidden layer 2, Hidden node 3



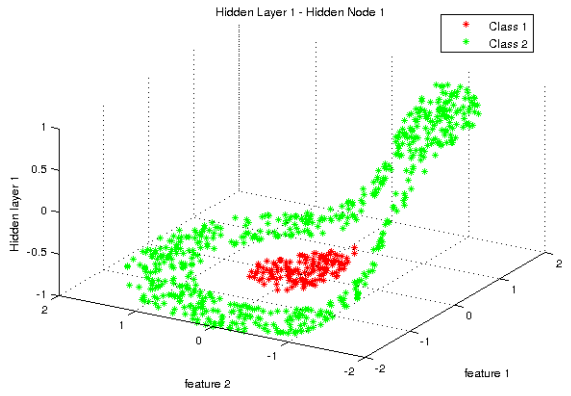
(b) Output node 1



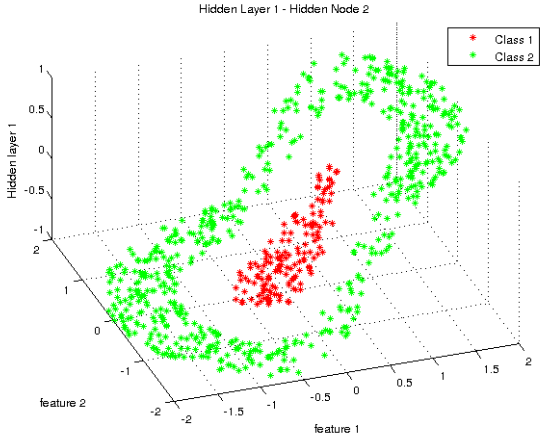
(c) Output node 2

Figure 31: Output nodes for No of Epochs =10

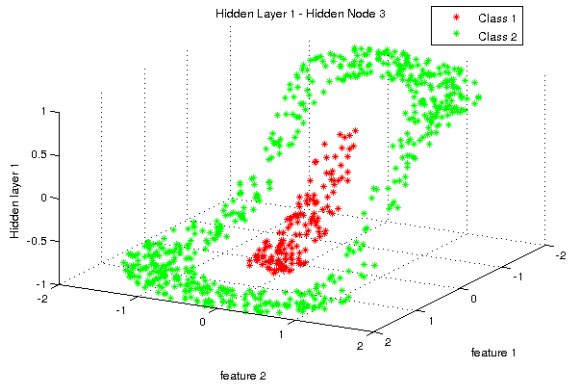
Using 2 hidden layer for no of epochs = 50



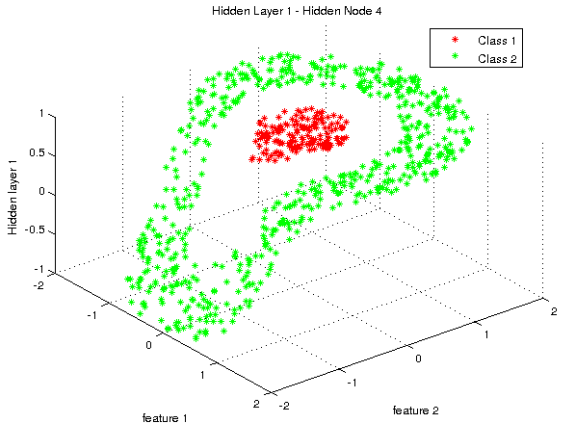
(a) Hidden layer 1, Hidden node 1



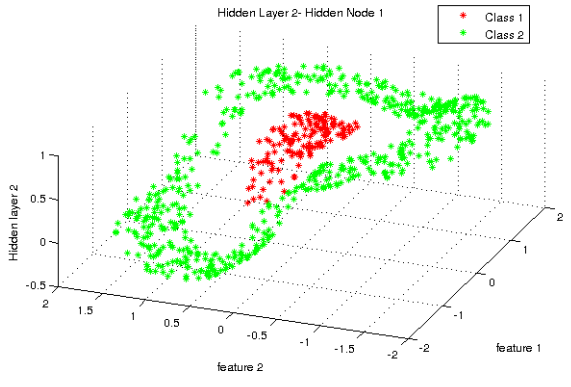
(b) Hidden layer 1, Hidden node 2



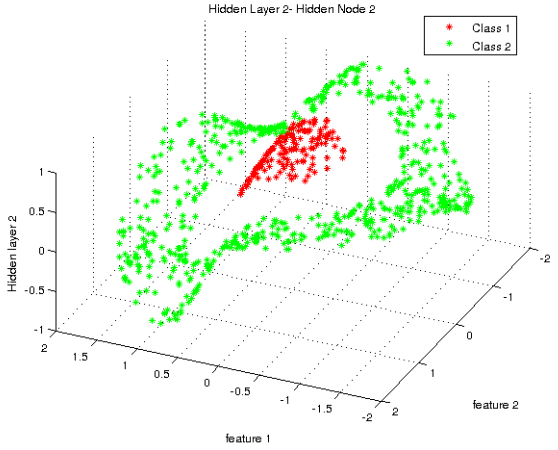
(c) Hidden layer 1, Hidden node 3



(d) Hidden layer 1, Hidden node 4

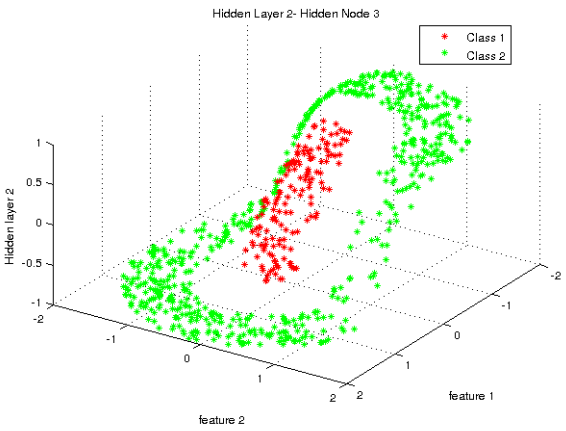


(e) Hidden layer 2, Hidden node 1

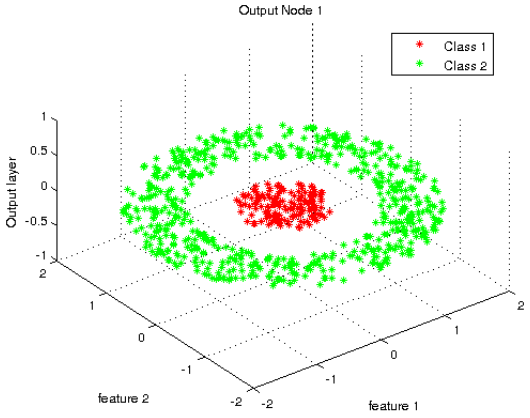


(f) Hidden layer 2, Hidden node 2

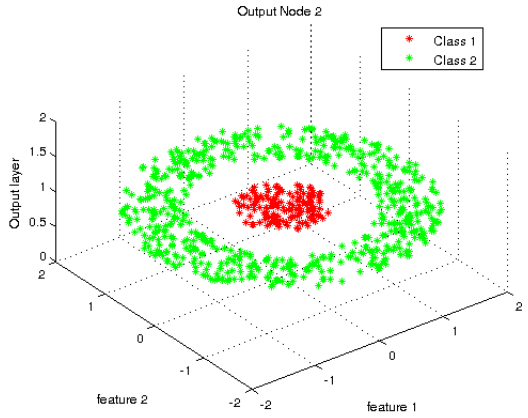
Figure 32: Hidden Nodes for No of Epochs =50



(a) Hidden layer 2, Hidden node 3



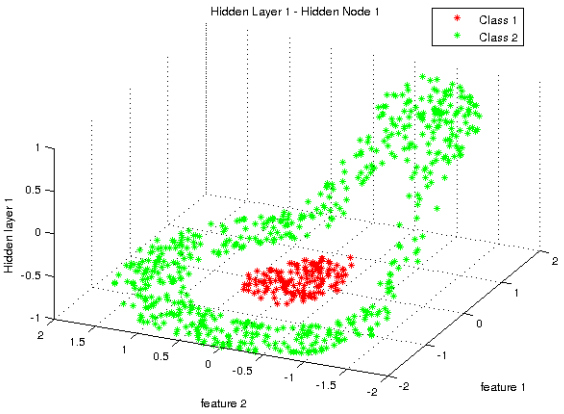
(b) Output node 1



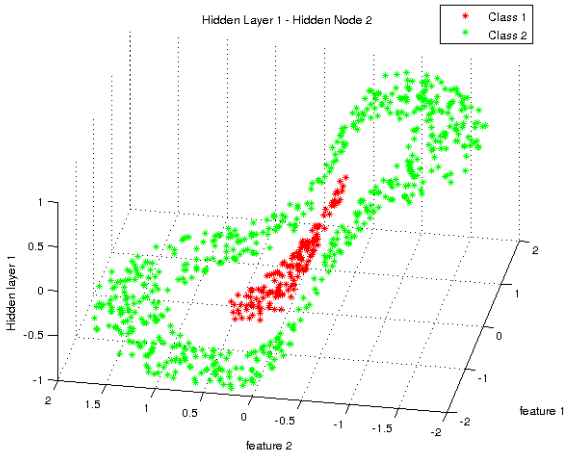
(c) Output node 2

Figure 33: Output nodes for No of Epochs =50

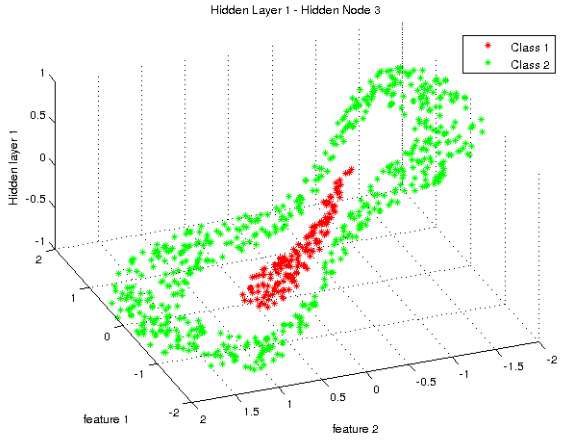
Using 2 hidden layer for no of epochs = 100



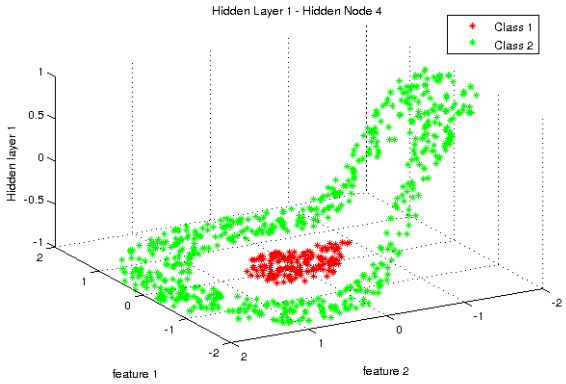
(a) Hidden layer 1, Hidden node 1



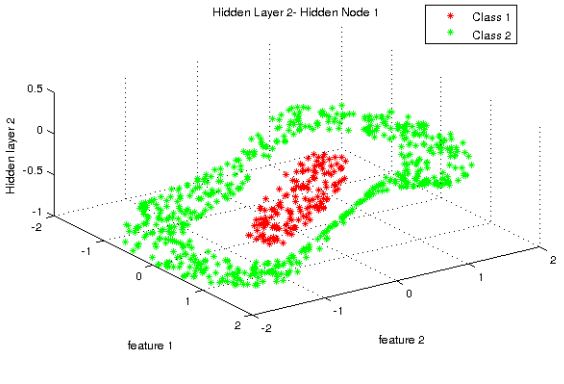
(b) Hidden layer 1, Hidden node 2



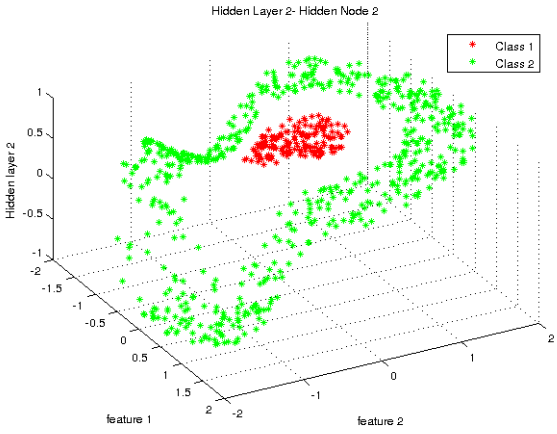
(c) Hidden layer 1, Hidden node 3



(d) Hidden layer 1, Hidden node 4

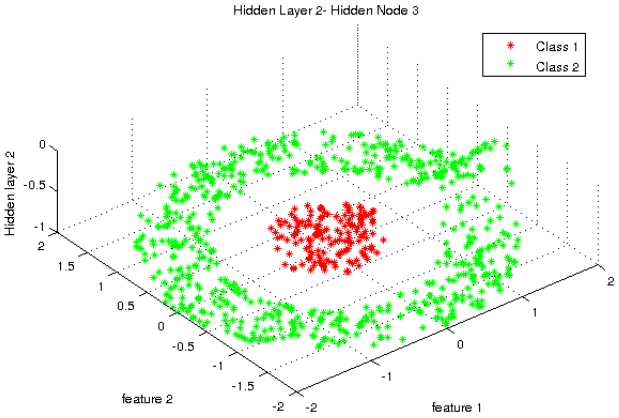


(e) Hidden layer 2, Hidden node 1

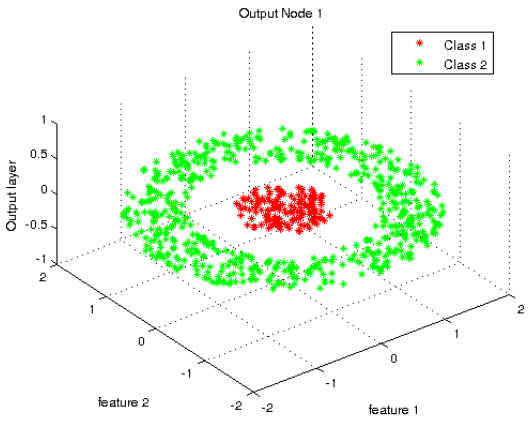


(f) Hidden layer 2, Hidden node 2

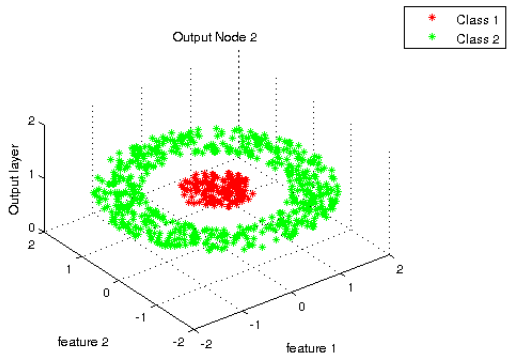
Figure 34: Hidden Nodes for No of Epochs =100



(a) Hidden layer 2, Hidden node 3



(b) Output node 1



(c) Output node 2

Figure 35: Output nodes for No of Epochs =100

Inferences

- For number of epochs=2, all data points get classified into one class

2.2.4 Comparison of performance of different models

Accuracy Comparison across different models

Method	Validation Accuracy (%)	Test Accuracy (%)
Bayes Classifier	100	100
MLFFN 1 hidden layer	100	100

2.3 Overlapping classes

The given data includes 3 linearly separable classes. Visualisation of the training data is as in Figure 36

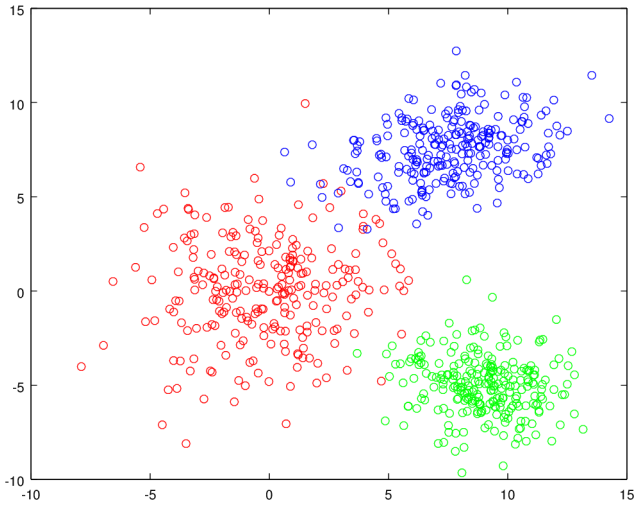
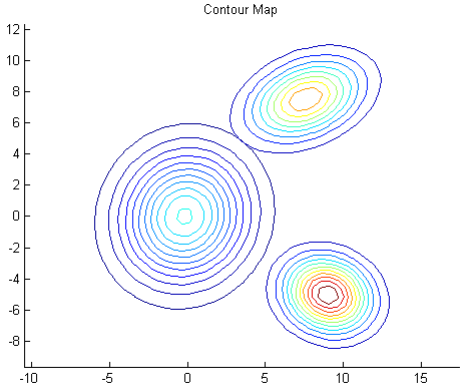
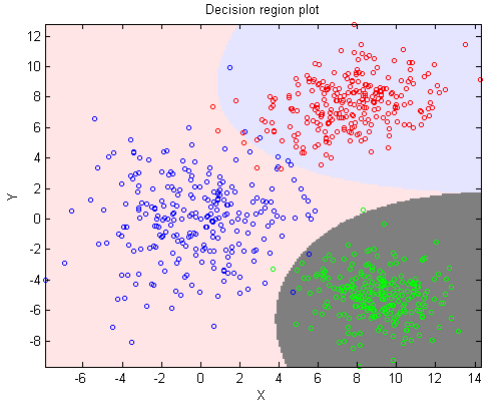


Figure 36: Overlapping classes

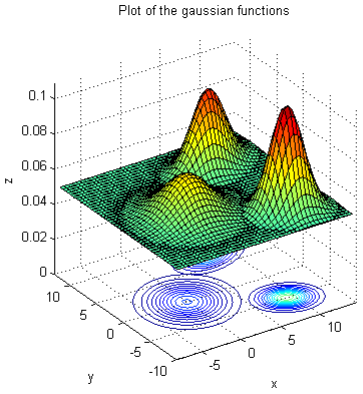
2.3.1 Bayes classifier using Gaussian model



(a) Contour map



(b) Decision region plot



(c) Gaussian plot

Figure 37: Bayes classifier using Gaussian model for Overlapping classes

The confusion matrices for validation and testing data are as follows:

	class1	class2	class3
class1	146	3	1
class2	1	149	0
class3	1	0	149

Table 7: Validation data

	class1	class2	class3
class1	97	1	2
class2	2	98	0
class3	0	0	100

Table 8: Test data

Inferences

- 3 gaussians are used to model the given classes
- Accuracy obtained on validation data is 98.67%
- Accuracy obtained on test data is 98.33%

2.3.2 MLFFNN

For overlapped data, we used two hidden layers. The number of hidden nodes in each hidden layer are found using cross validation. The Figure 38 is plotted to decide the number of hidden nodes in first hidden layer. For second hidden layer we obtained the number of hidden nodes as 3.

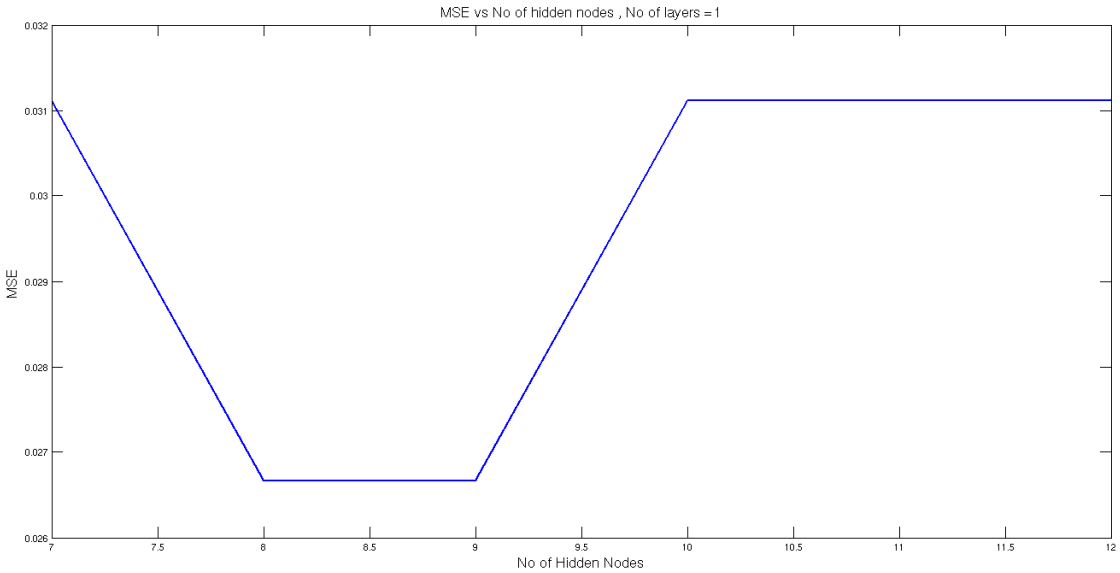


Figure 38: MSE vs Number of hidden nodes

Inferences

- Minimum MSE obtained for 8 hidden nodes in hidden layer 1 and 3 hidden nodes in hidden layer 2

We fixed the learning rate as 0.4 for this experiment(using cross validation). Also, we used generalized delta rule for weight update by using *traingdm*. The momentum factor we used is 0.8. The network view of our model is shown in Figure 39.

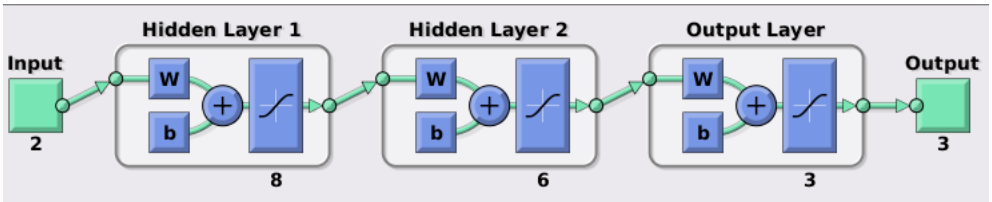


Figure 39: View of network

The decision plot obtained for the given data using the mentioned model parameters is shown in Figure 40.

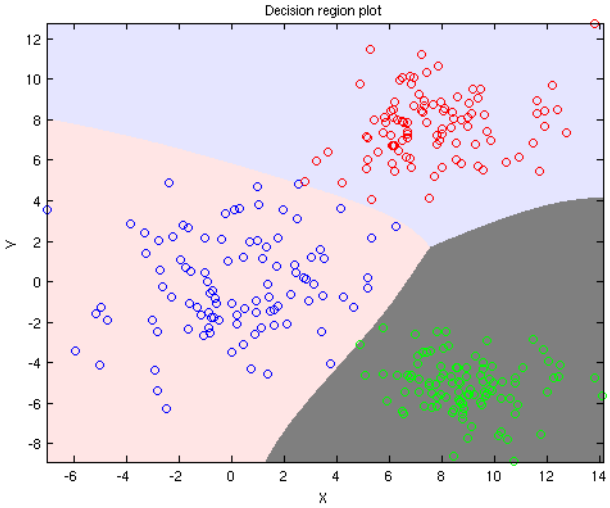
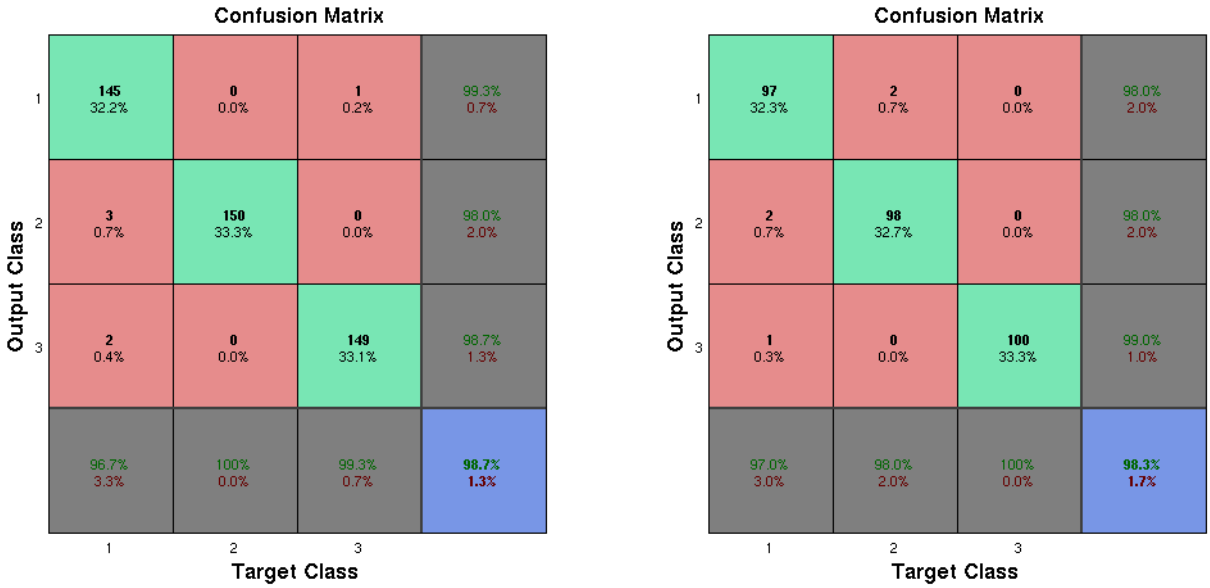


Figure 40: Decision region plot

The model gave 98% accuracy. The corresponding confusion matrix for validation data and test data are shown in Figure 41.



(a) Validation data

(b) Test data

Figure 41: Confusion Matrix

Inferences

- 98.7% accuracy obtained for validation data
- 98.3% accuracy obtained for test data

2.3.3 Comparison of performance of different models

Accuracy Comparison across different models

Method	Validation Accuracy (%)	Test Accuracy (%)
Bayes Classifier	98.6	98.3
MLFFN 1 hidden layer	98.7	98.3

Inferences

- Both are performing similar

2.4 Image data

2.4.1 Bayes classifier using GMM

We used GMM on the image dataset with the following number of clusters.

```
class 1 : 15
class 2 : 15
class 3 : 25
class 4 : 25
class 5 : 25
```

We obtained 62% accuracy on the validation and test data. Confusion matrices are as shown below:

	1	2	3	4	5
1	20	0	0	0	0
2	6	18	1	0	3
3	6	0	8	0	1
4	8	5	0	2	1
5	4	0	1	1	13

Table 9: Validation Data

	1	2	3	4	5
1	32	7	0	0	1
2	16	35	1	1	4
3	1	0	25	0	3
4	12	10	0	4	5
5	11	2	0	0	25

Table 10: Test data

Inferences

The classes 2,4,5 are frequently confused with those in classes 1.

2.4.2 MLFFNN with 2 hidden layers

We used Levenberg-Marquardt optimization method for training which was giving more performance than gradient descent method. The activation function used is hyperbolic tangent function. The learning rate is chosen as 0.4 using cross validation. Two hidden layers are used. The plot for number of hidden nodes in hidden layer 1 versus MSE is shown in Figure 43.

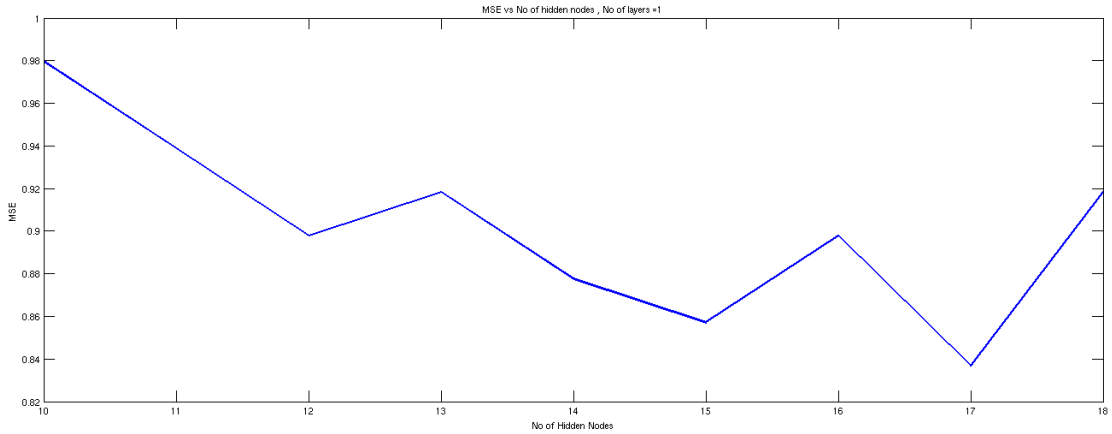


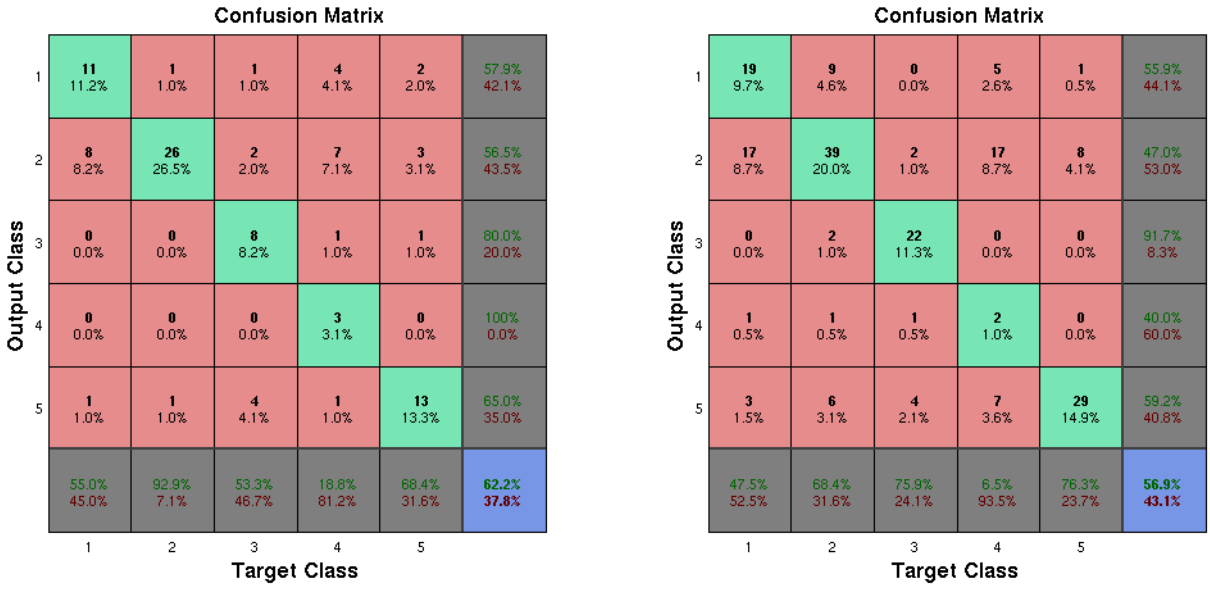
Figure 42: MSE vs No of hidden nodes in hidden layer 1

Inferences

- The minimum error obtained for 17 hidden nodes.

By cross validation we chose the number of hidden nodes in the hidden layer 2 as 3.

Then we trained the model using these parameters and the confusion matrix for validation data and test data are shown in Figure 43.



(a) Validation data

(b) Test data

Figure 43: Confusion Matrix

Inferences

- Obtained 62.2% accuracy for validation data and 56.9% accuracy for test data
- The performance is poor compared to other datasets, since this is a real time data.

2.4.3 Comparison of performance of different models

Accuracy Comparison across different models

Method	Validation Accuracy (%)	Test Accuracy (%)
Bayes Classifier	62	62
MLFFN 2 hidden layer	62.2	56.9

Inferences

- Bayesian is performing better in test data
- We are not getting better accuracy because this is real time data

3 Regression tasks

3.1 MLFFNN on Univariate Dataset

The underlying function of univariate dataset is $cos^2(2\pi x)$. A gaussian noise with zero mean is added. For this experiment we generated 5000 samples for training data, 1000 samples for validation data and 500 samples for test data. We plotted the values of mean squared error(MSE) on training data, validation data and test data across varying number of hidden nodes. The plot is shown in Figure 44.

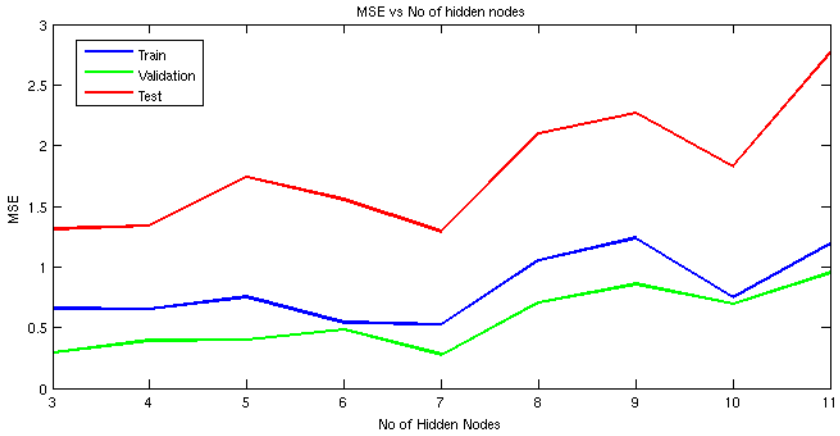
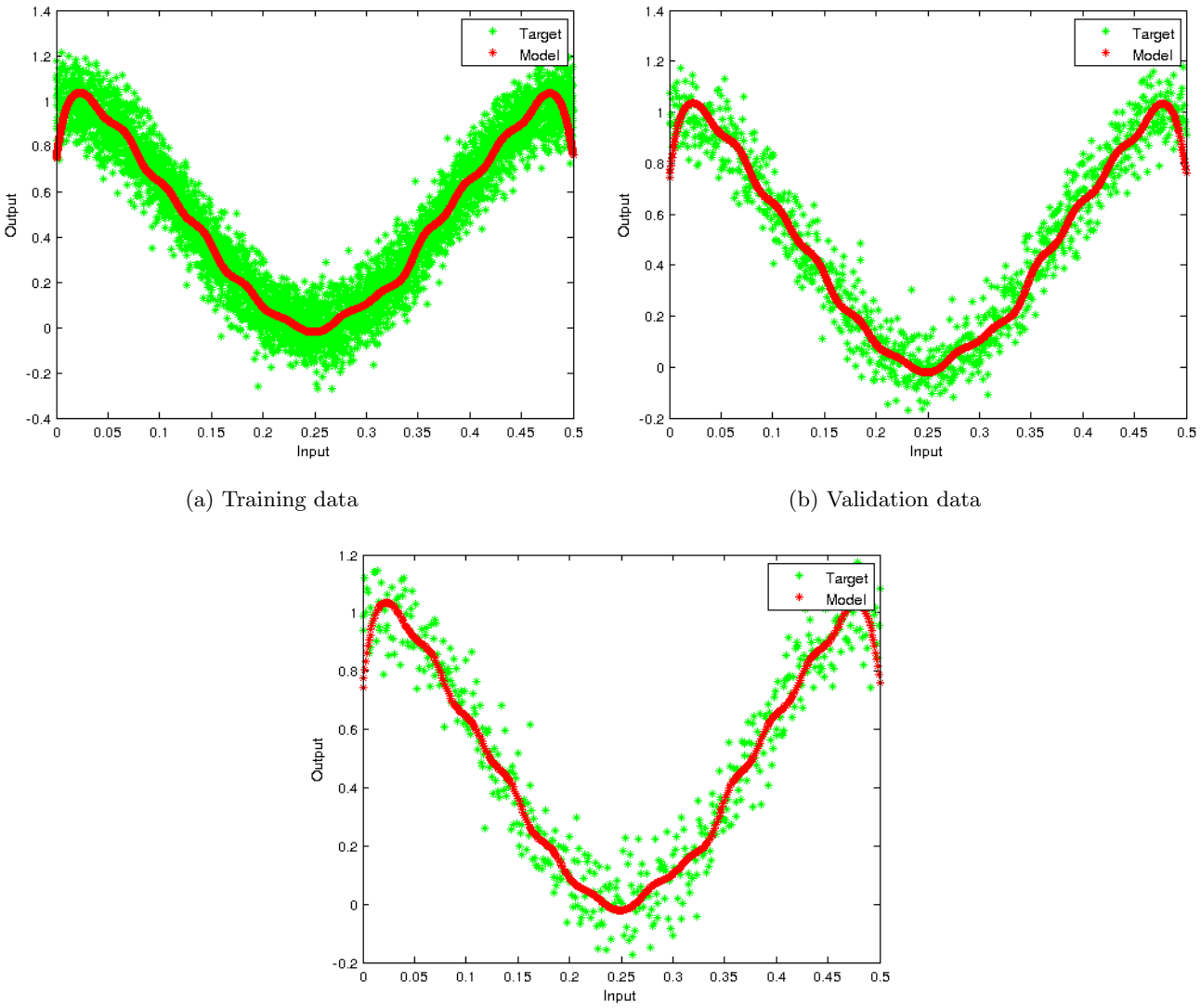


Figure 44: MSE versus No of hidden nodes

Inferences

- The minimum validation error obtained for No of hidden nodes=12

The plot across model output and target output for three sets data are shown in Figure 45.



(a) Training data

(b) Validation data

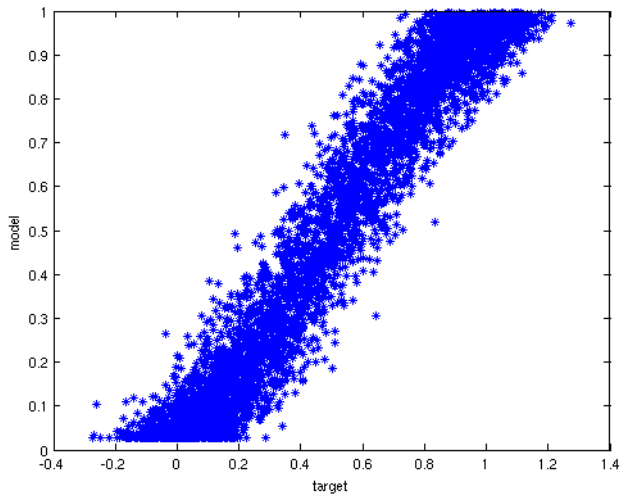
(c) Test data

Figure 45: Model output and target output for training data, validation data and test dat

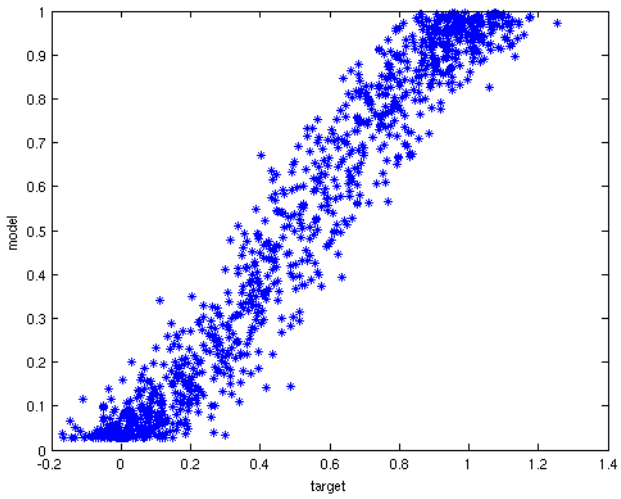
Inferences

- The approximated function is somewhat fitting the underlying function

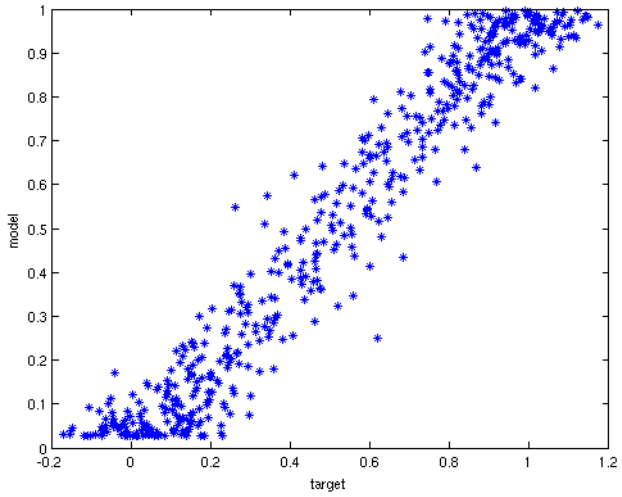
The scatter plots with target output on x axis and model output on y axis for training data, validation data and test data are shown in Figure 46.



(a) Training data



(b) Validation data



(c) Test data

Figure 46: Scatter plot with target output as x axis and model output as y axis

Result

Error	MSE
Training	0.0086
Validation	0.0081
Testing	0.0085

Inferences

- The scatter plots for all the three datasets are close to y=x line.
- The error is similar to regression tasks.

3.2 MLFFNN on Bivariate Dataset

We experimented the bivariate datasets for various training data size. The plots obtained for training data size 100 and 2000 are shown in this report. We trained the data using gradient descent method and Levenberg-Marquardt optimization method. The number of hidden nodes we chose using cross validation and we experimented for one hidden layer and two hidden layer. We used hyperbolic tangent function as activation function in hidden layers and linear activation function in output layer. The learning rate fixed through cross validation and it is fixed as 0.1.

The plot of the values of MSE on training data, validation data and test data for different hidden nodes are shown in Figure 47.

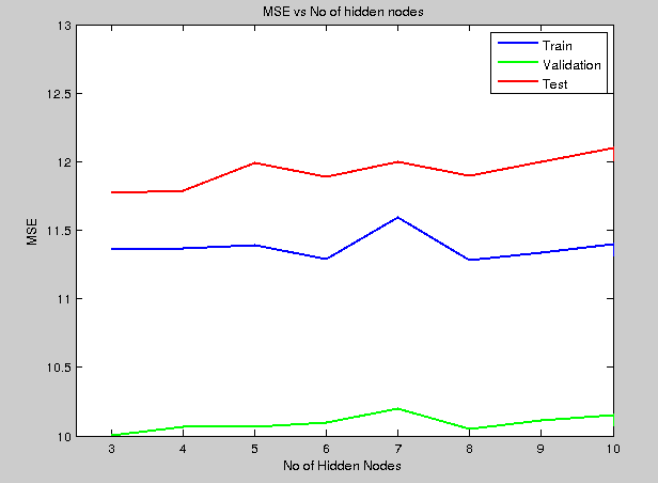


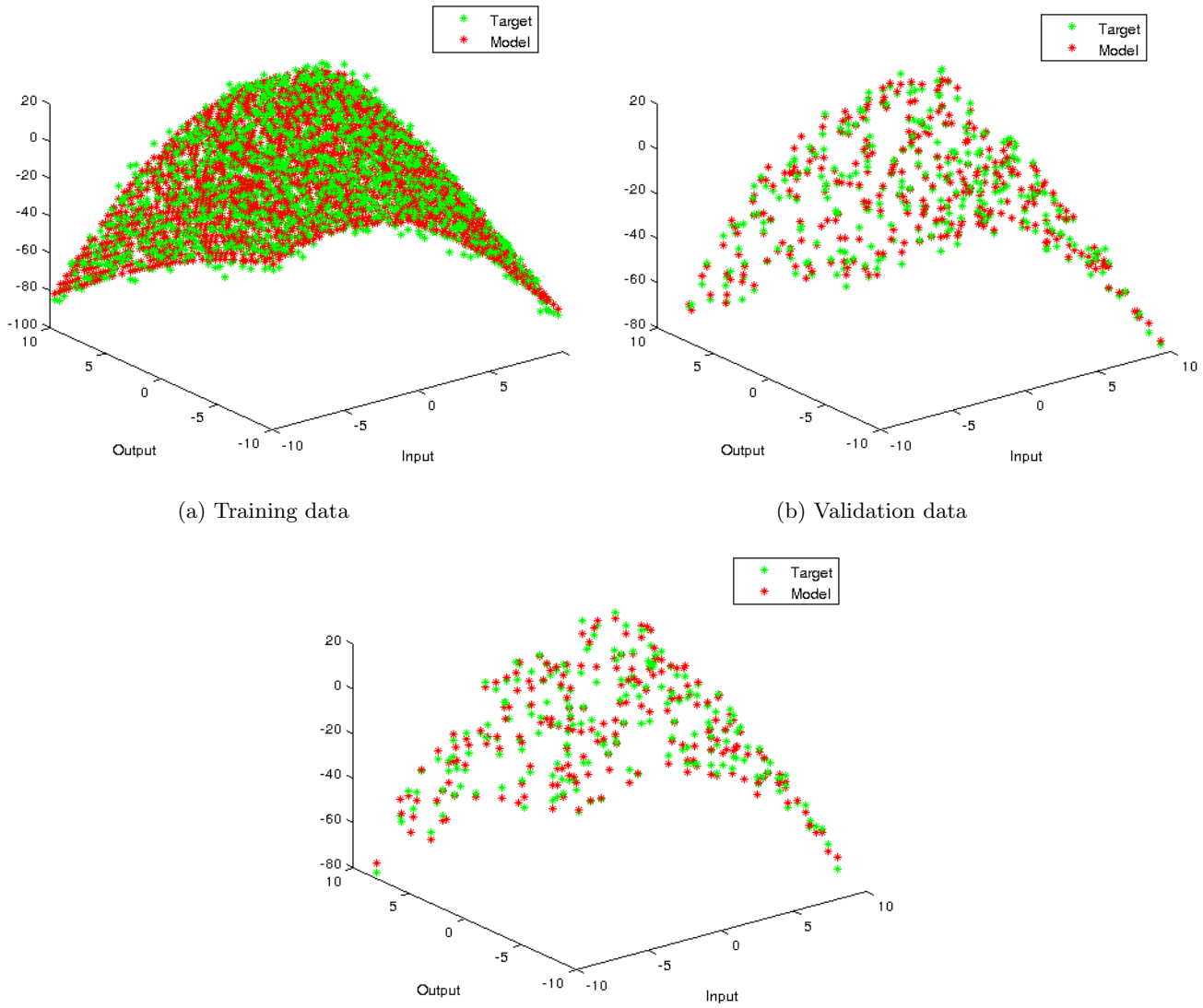
Figure 47: MSE vs No of hidden nodes

Inferences

- The best performance is at no of hidden nodes =3

So we chose number of hidden nodes in first hidden layer as 3 and second hidden layer as 2.

The plot across model output and target output for three sets data with training data size as 2000 and one hidden layer are shown in Figure 48.



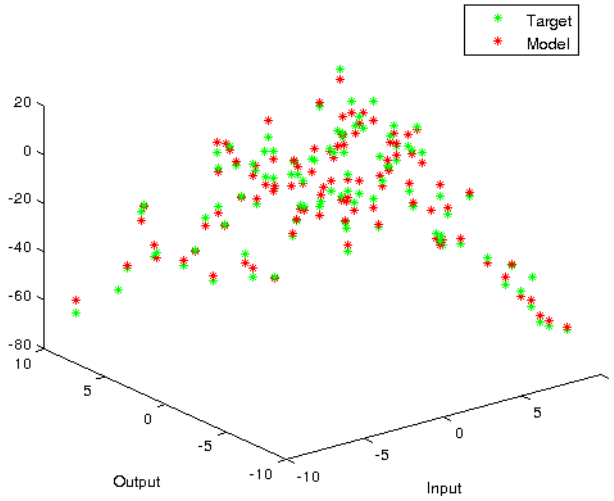
(a) Training data

(b) Validation data

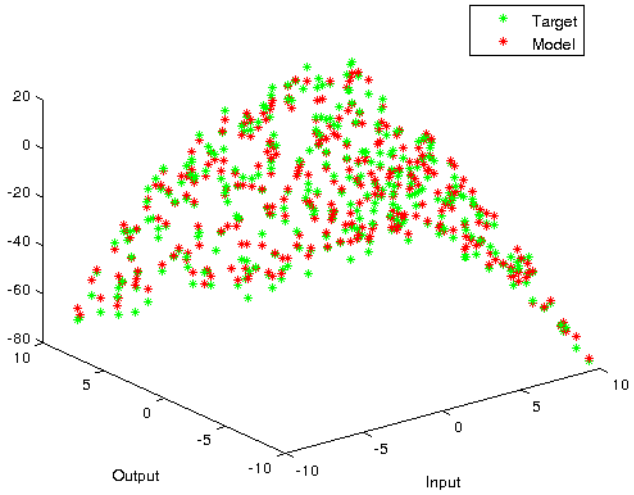
(c) Test data

Figure 48: Model output and target output for training data, validation data and test data with training data size=2000

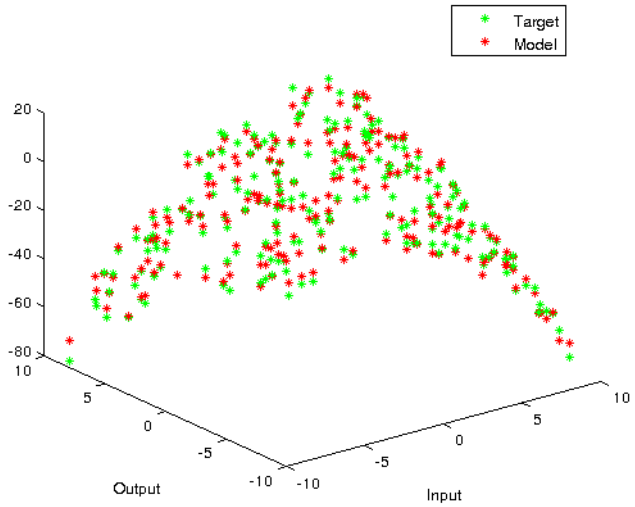
The plot across model output and target output for three sets data with training data size as 100 and one hidden layer are shown in Figure 49.



(a) Training data



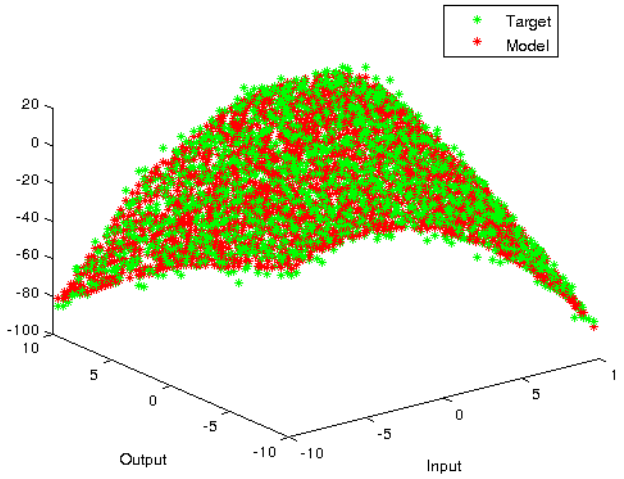
(b) Validation data



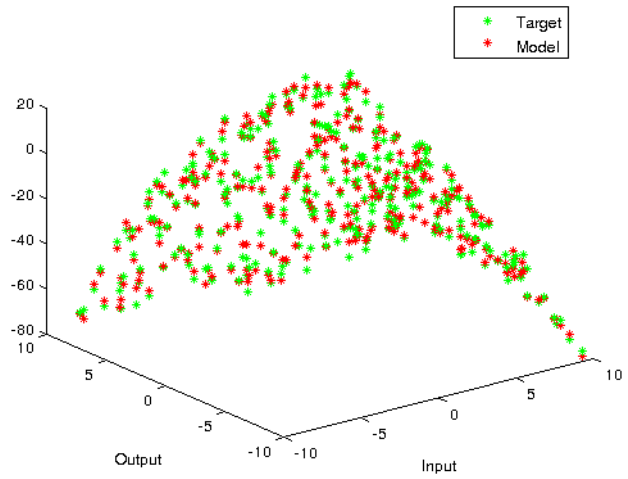
(c) Test data

Figure 49: Model output and target output for training data, validation data and test data with training data size=100

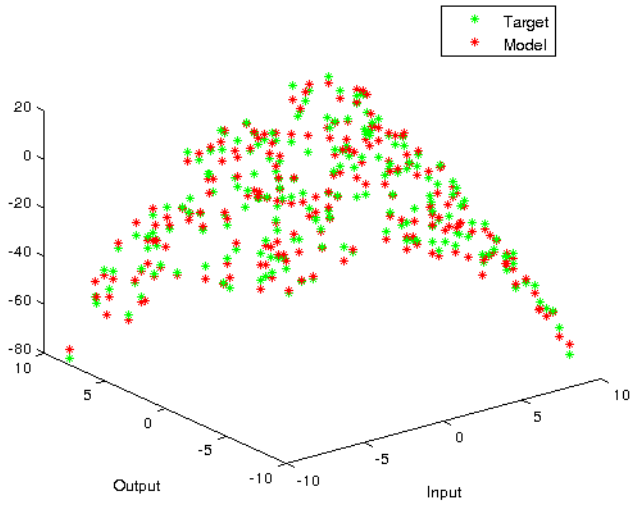
The plot across model output and target output for three sets data with training data size as 2000 and two hidden layer are shown in Figure 50.



(a) Training data



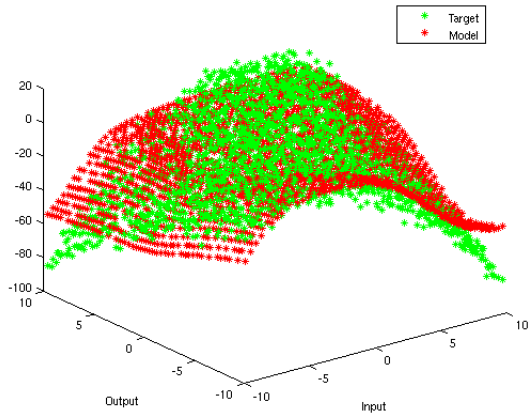
(b) Validation data



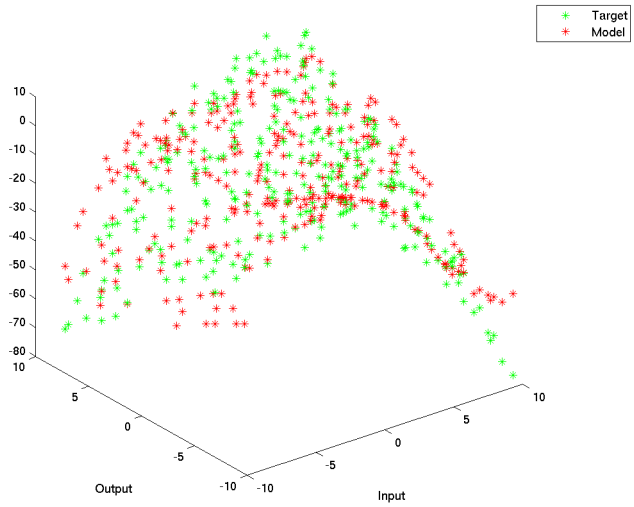
(c) Test data

Figure 50: Model output and target output for training data, validation data and test data with training data size=2000 obtained using two hidden layer

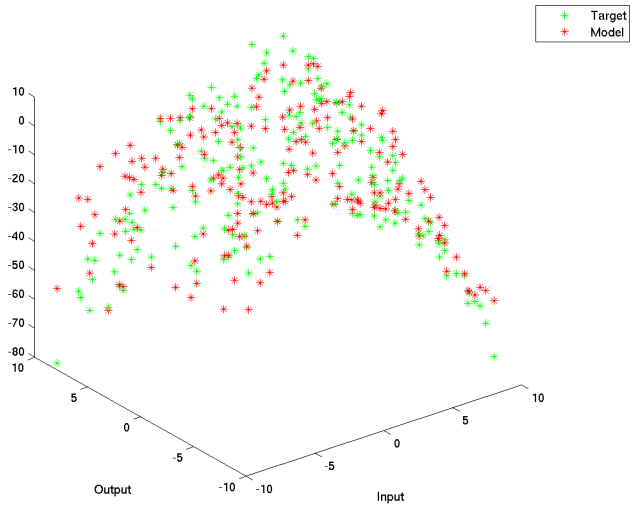
The Figures 48, 49 and 50 are obtained using Levenberg-Marquardt optimization method. The plots for the same using gradient descent method for training data size 2000 is shown in Figure 51.



(a) Training data



(b) Validation data



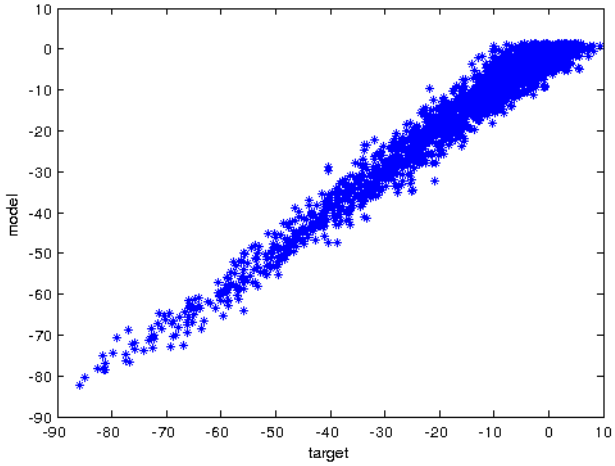
(c) Test data

Figure 51: Model output and target output for training data, validation data and test data with training data size=2000 obtained using gradient descent method

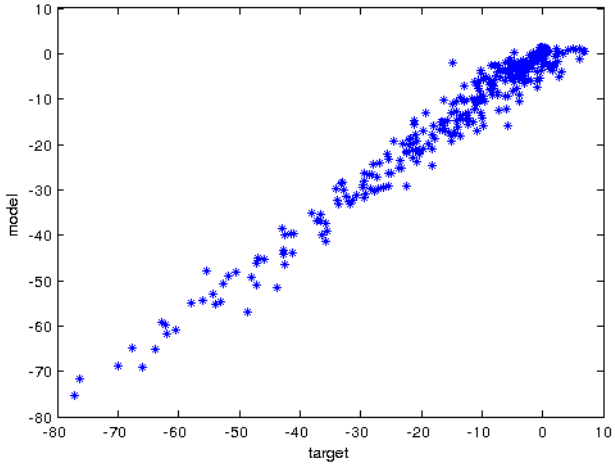
Inferences

- Levenberg-Marquardt optimization method performs well than gradient descent method

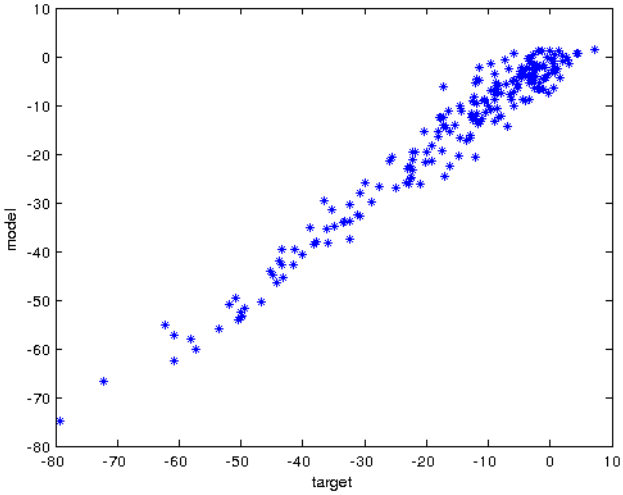
The scatter plot with target output on x-axis and model output on y-axis for three sets data with training data size as 2000 and one hidden layer are shown in Figure 52.



(a) Training data



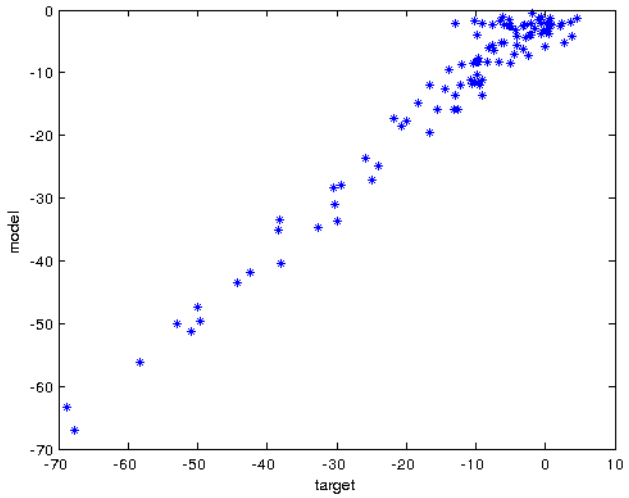
(b) Validation data



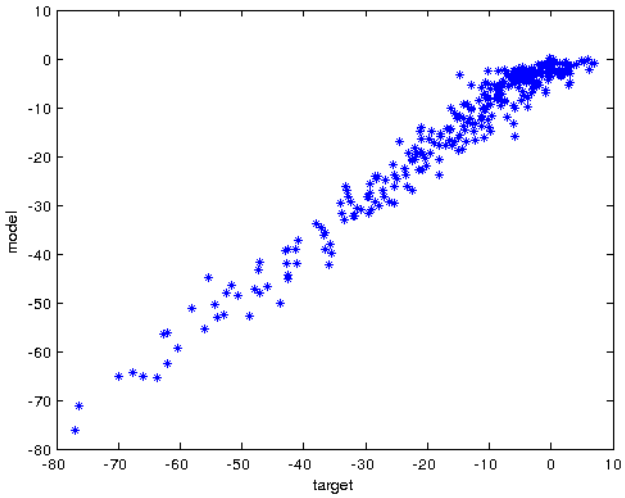
(c) Test data

Figure 52: Scatter plot with target output on x-axis and model output on y-axis with training data size=2000

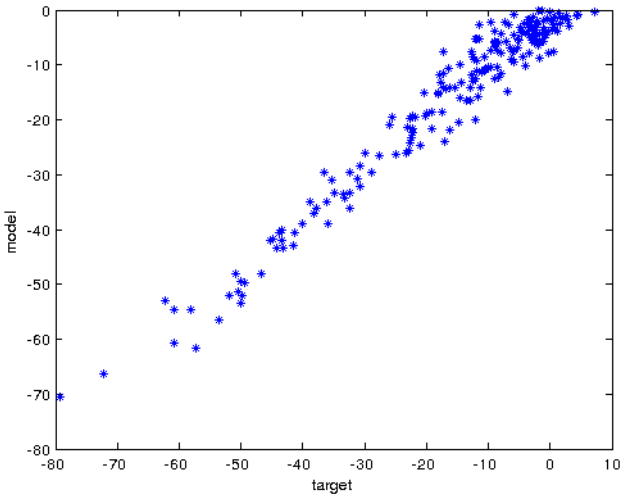
The scatter plot with target output on x-axis and model output on y-axis for three sets data with training data size as 100 and one hidden layer are shown in Figure 53.



(a) Training data



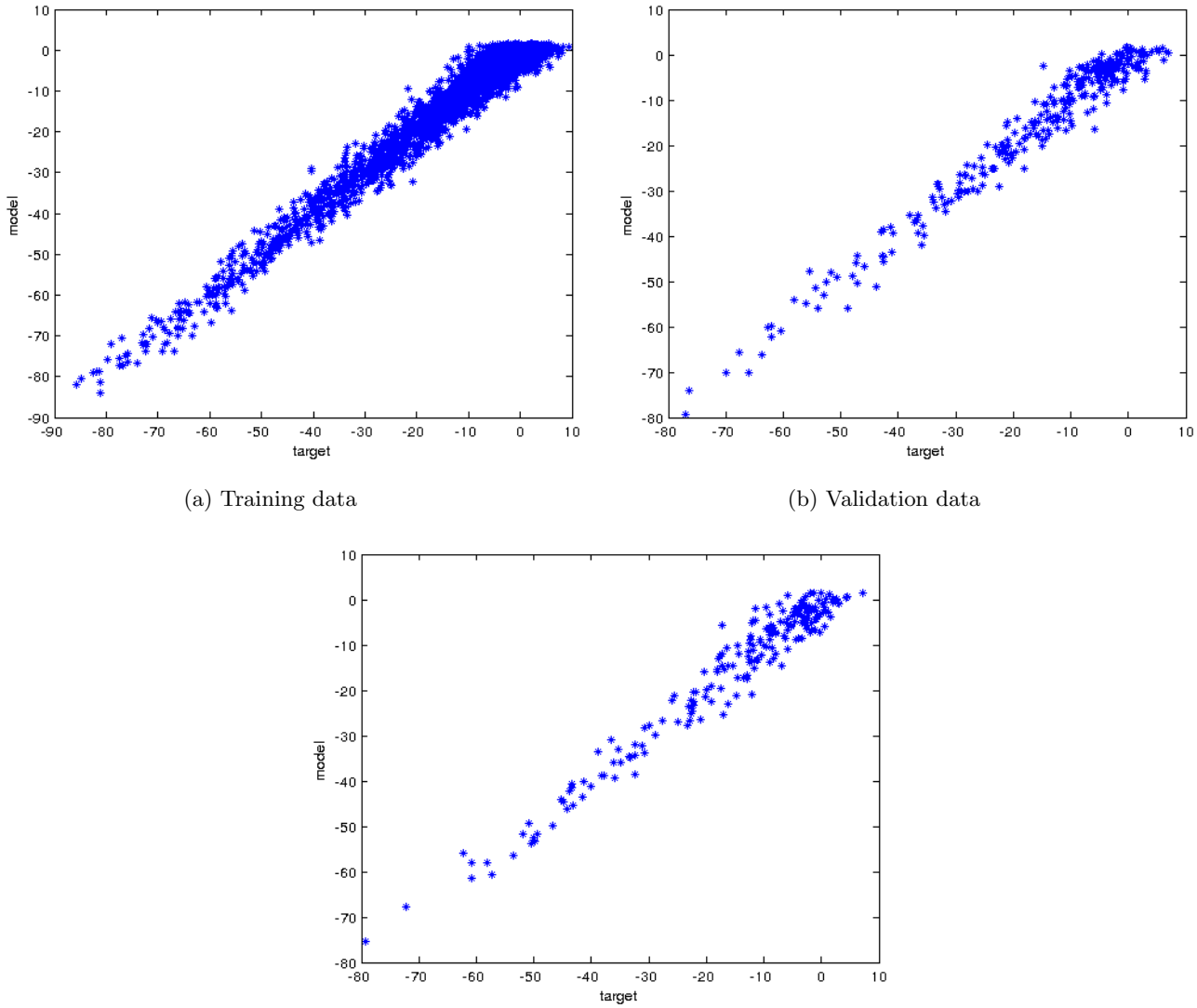
(b) Validation data



(c) Test data

Figure 53: scatter plot with target output on x-axis and model output on y-axis for training data, validation data and test data with training data size=100

The scatter plot with target output on x-axis and model output on y-axis for three sets data with training data size as 2000 and two hidden layer are shown in Figure 54.



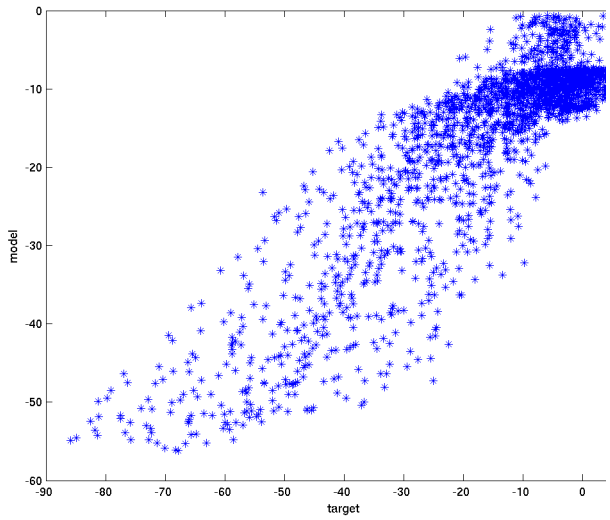
(a) Training data

(b) Validation data

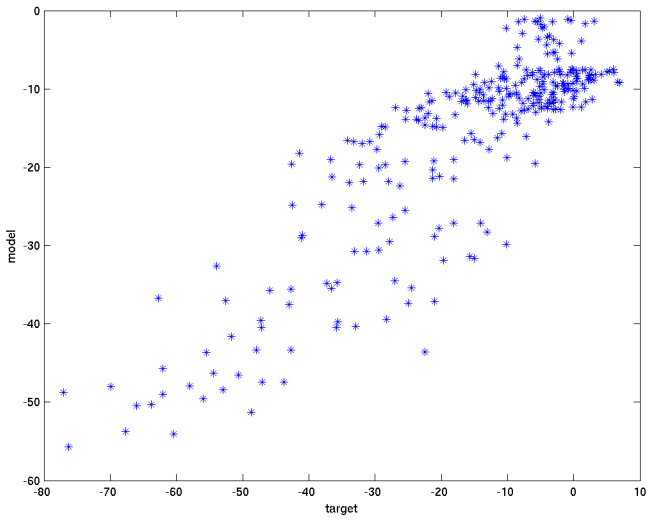
(c) Test data

Figure 54: scatter plot with target output on x-axis and model output on y-axis for training data, validation data and test data with training data size=2000 obtained using two hidden layer

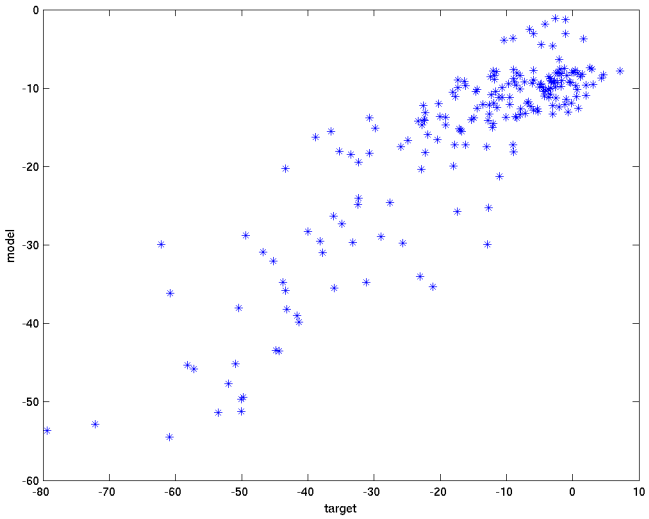
The Figures 52, 53 and 54 are obtained using Levenberg-Marquardt optimization method. The plots for the same using gradient descent method for training data size 2000 is shown in Figure 55.



(a) Training data



(b) Validation data



(c) Test data

Figure 55: scatter plot with target output on x-axis and model output on y-axis for training data, validation data and test data with training data size=2000 obtained using gradient descent method

Result

Error	MSE for N=2000 One hidden layer	MSE for N=2000 Two hidden layer	MSE for N=2000 Using Gradient descent	MSE for N=100 One hidden layer
Training	11.46	11.22	81.13	11.33
Validation	10.06	10.06	78.60	11.74
Testing	11.82	12.05	75.47	12.69

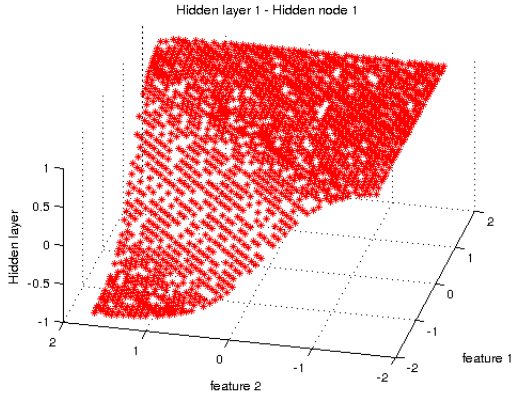
Inferences

- The points in the scatter plots are across 45 degree inclination of the axes indicating that the modeled data follow the target data very well
- The plots obtained for gradient decent method based training doesn't show much inclination as other method do.
- Error is less compared to regression models
- MSE obtained for the model using gradient descent method is too high compared to Levenberg-Marquardt method

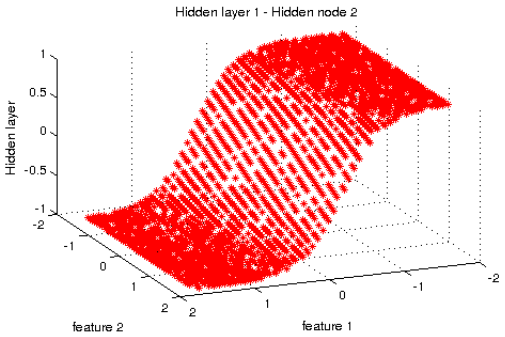
3.2.1 Plots of Outputs of each of the hidden nodes and output nodes

The best model obtained for one hidden layer contains 3 hidden nodes. The plot of outputs of each of the hidden nodes and output nodes for various number of epochs are shown below. These plot are generated for training data set size 2000.

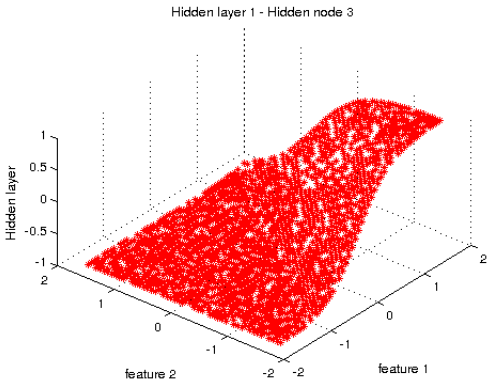
Using 1 hidden layer for no of epocs =1



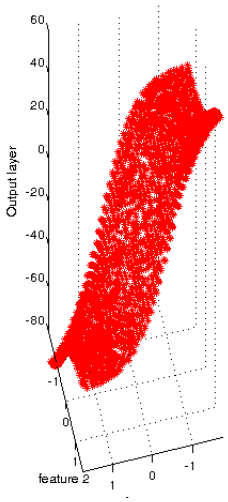
(a) First hidden node



(b) Second hidden node



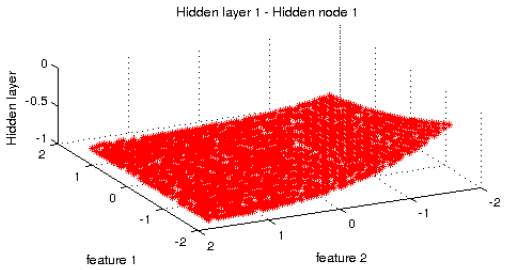
(c) Third hidden node



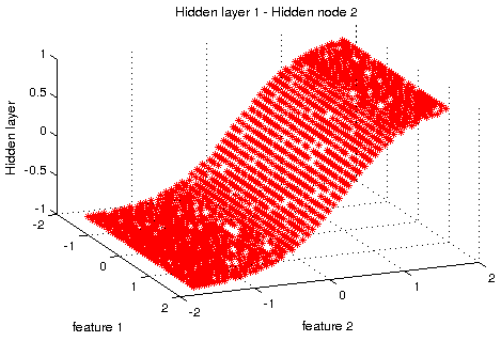
(d) Output nodes

Figure 56: Plot for hidden nodes and output node values for no of epochs =1

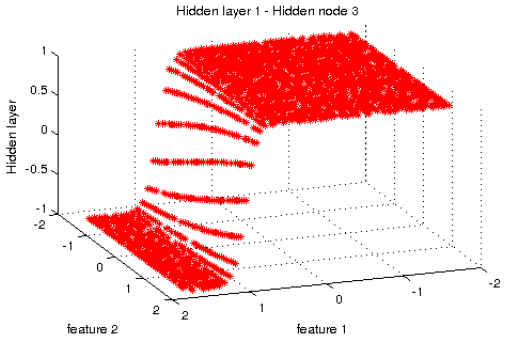
Using 1 hidden layer for no of epocs =2



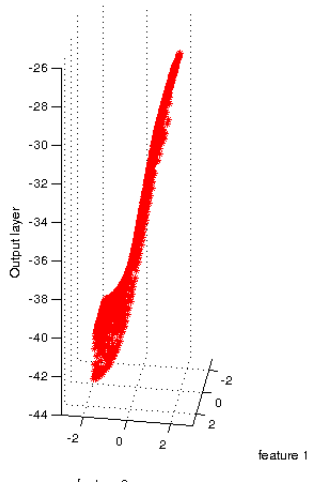
(a) First hidden node



(b) Second hidden node



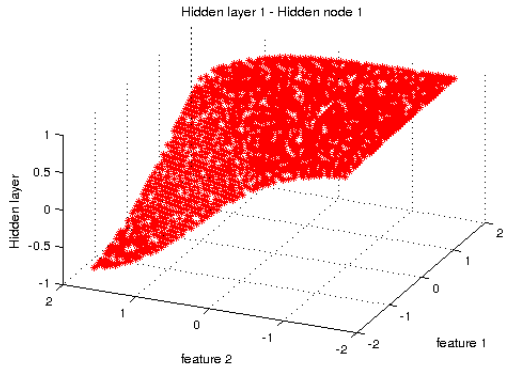
(c) Third hidden node



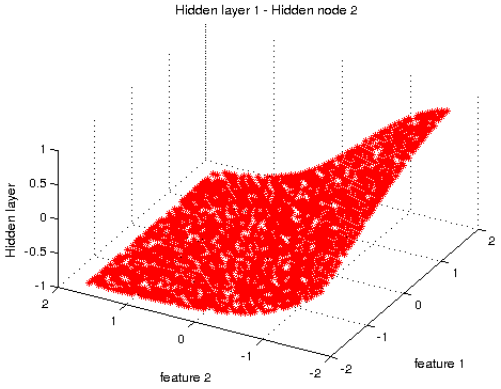
(d) Output nodes

Figure 57: Plot for hidden nodes and output node values for no of epochs =2

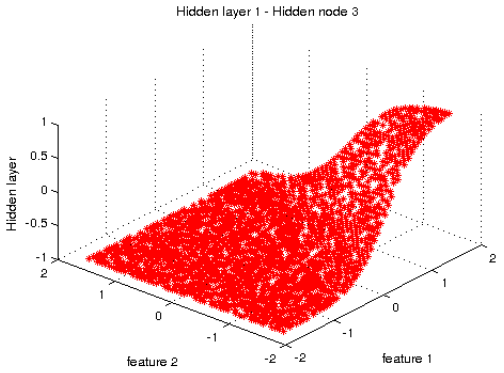
Using 1 hidden layer for no of epocs =10



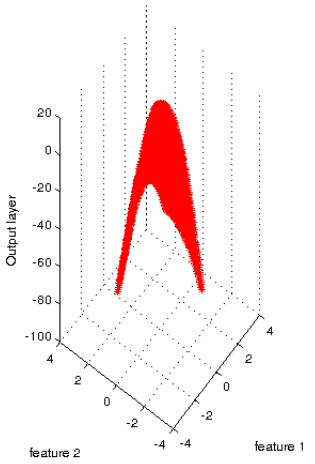
(a) First hidden node



(b) Second hidden node



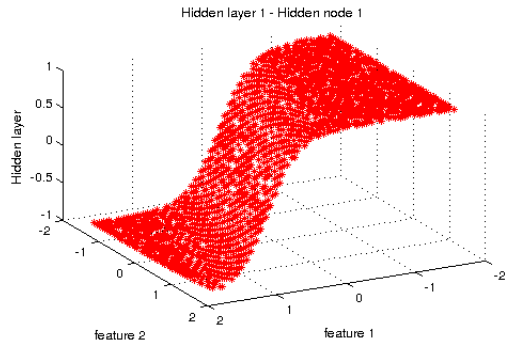
(c) Third hidden node



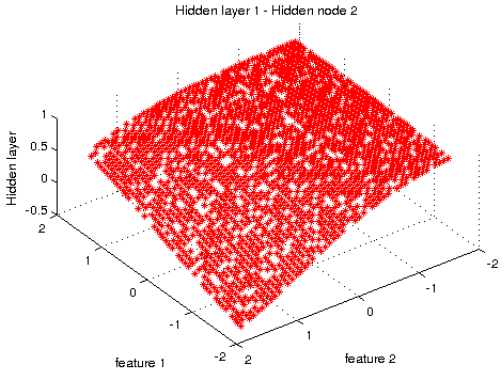
(d) Output nodes

Figure 58: Plot for hidden nodes and output node values for no of epochs =10

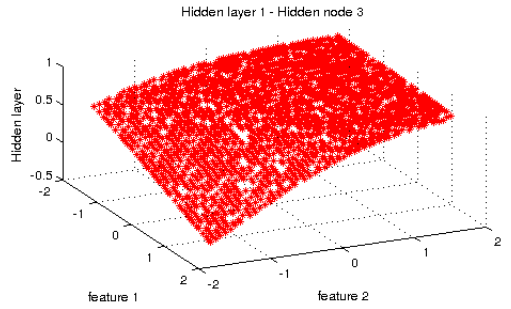
Using 1 hidden layer for no of epocs =50



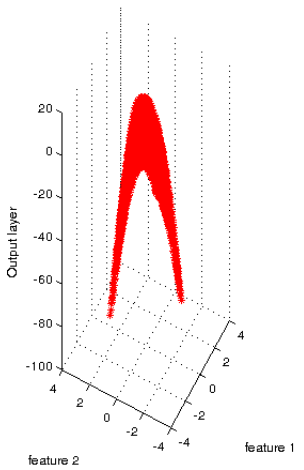
(a) First hidden node



(b) Second hidden node



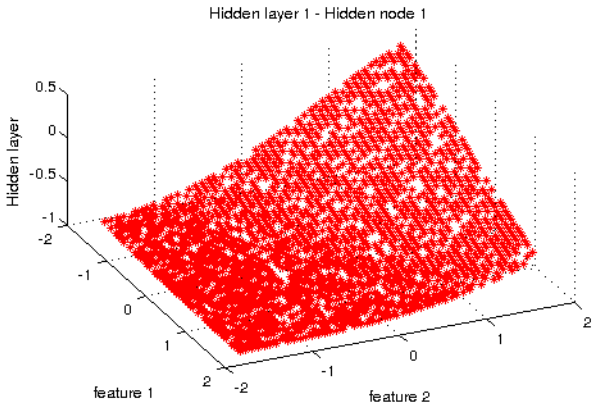
(c) Third hidden node



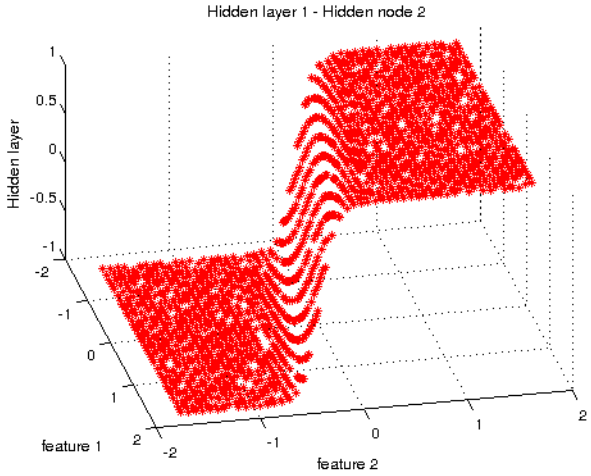
(d) Output nodes

Figure 59: Plot for hidden nodes and output node values for no of epochs =50

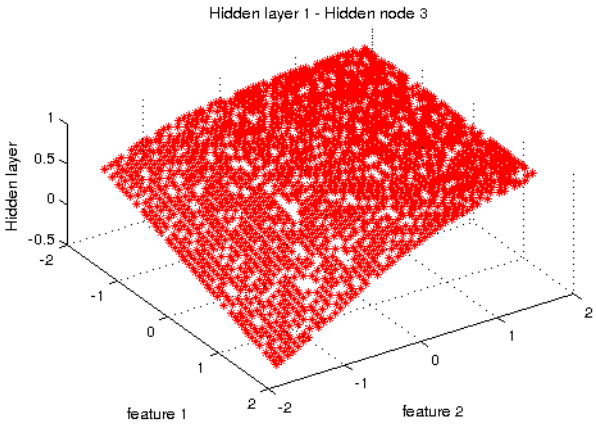
Using 1 hidden layer for no of epocs =100



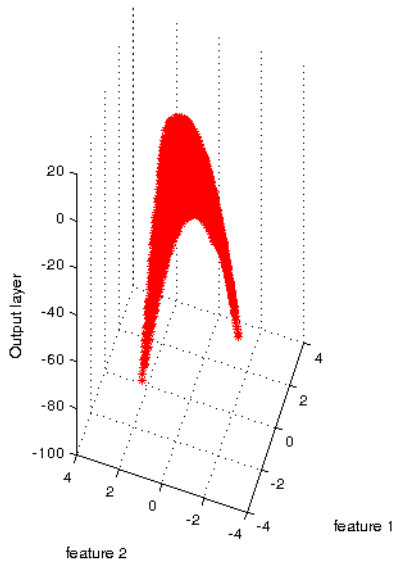
(a) First hidden node



(b) Second hidden node



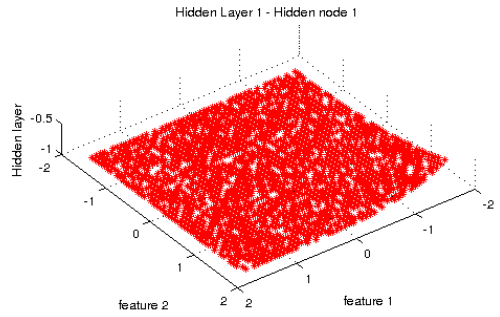
(c) Third hidden node



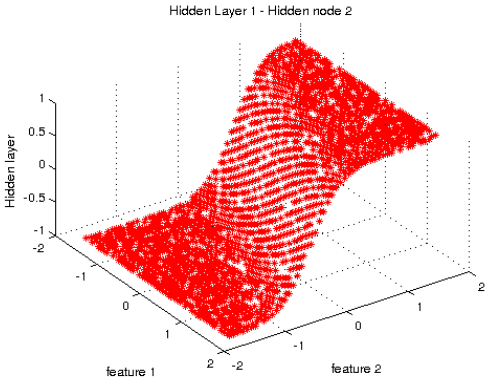
(d) Output nodes

Figure 60: Plot for hidden nodes and output node values for no of epochs =100

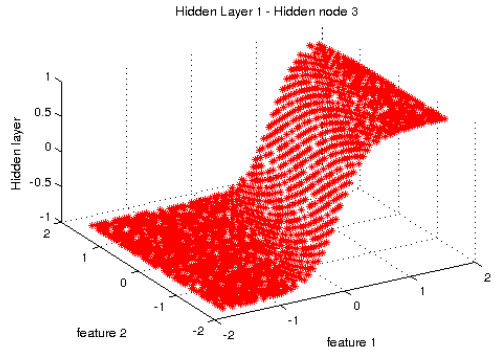
Using 2 hidden layer for no of epocs =1



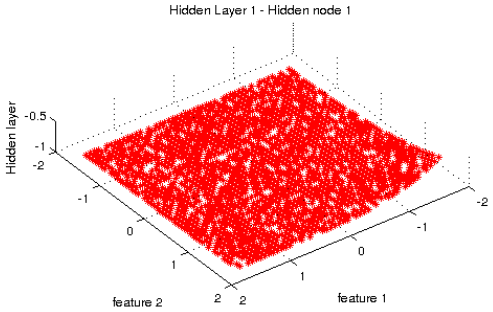
(a) Hidden layer 1 ,First hidden node



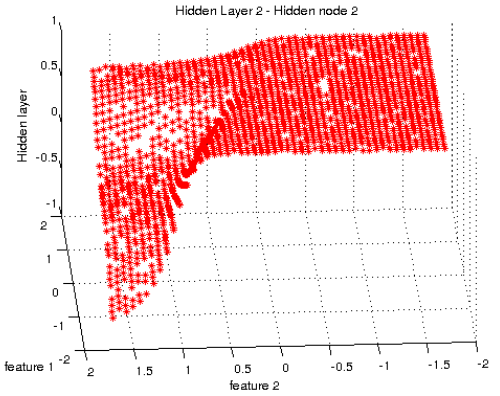
(b) Hidden layer 1, Second hidden node



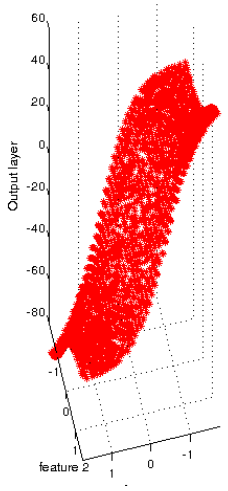
(c) Hidden layer 1, Third hidden node



(d) Hidden layer 2, First hidden node



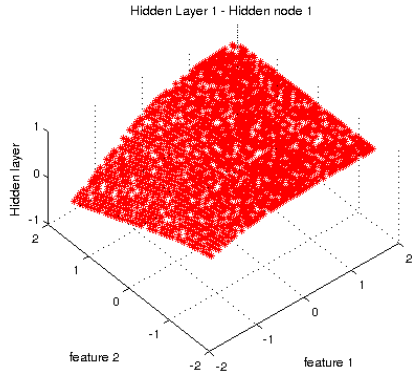
(e) Hidden layer 2, Second hidden node



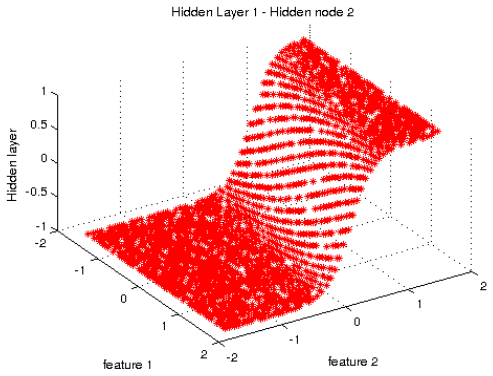
(f) Output nodes

Figure 61: Plot for hidden nodes and output node values for no of epochs =1

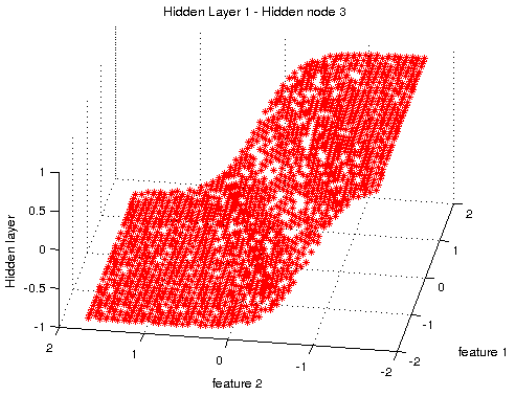
Using 2 hidden layer for no of epocs =2



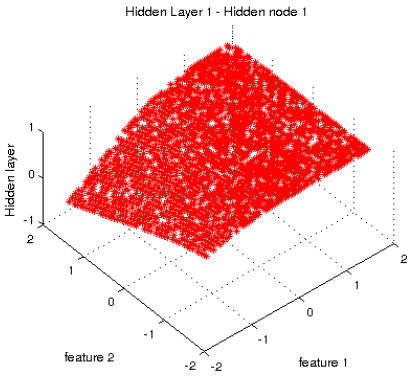
(a) Hidden layer 1 ,First hidden node



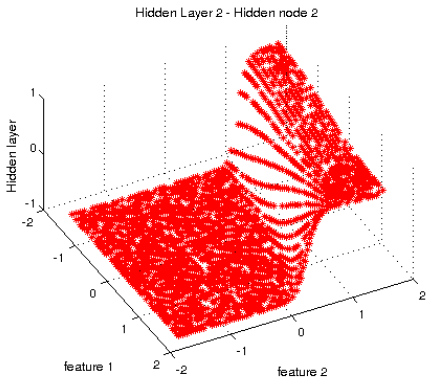
(b) Hidden layer 1, Second hidden node



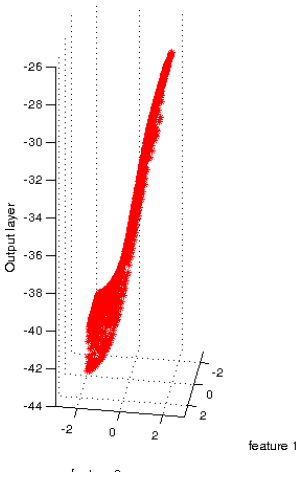
(c) Hidden layer 1, Third hidden node



(d) Hidden layer 2, First hidden node



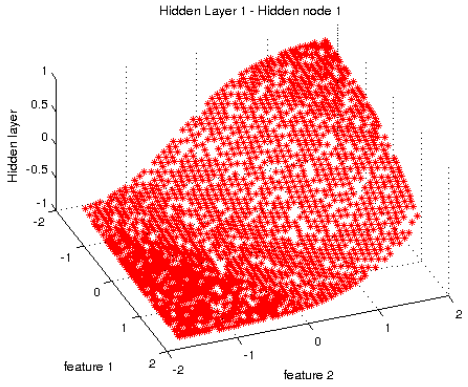
(e) Hidden layer 2, Second hidden node



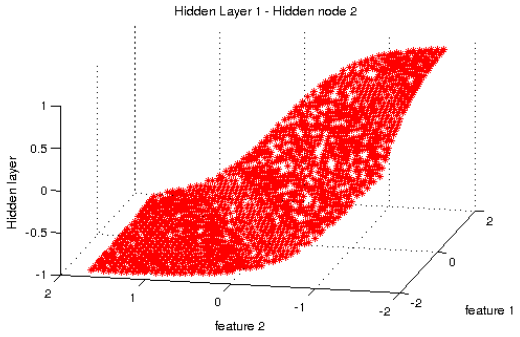
(f) Output nodes

Figure 62: Plot for hidden nodes and output node values for no of epochs =2

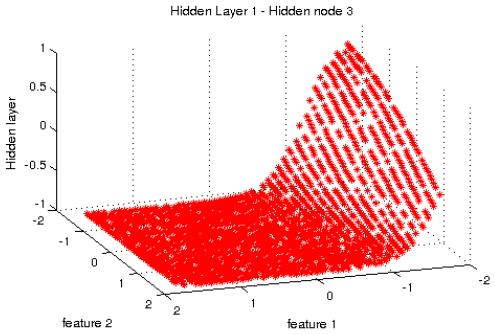
Using 2 hidden layer for no of epocs =10



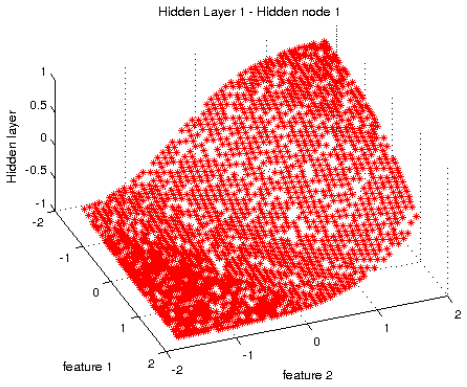
(a) Hidden layer 1 ,First hidden node



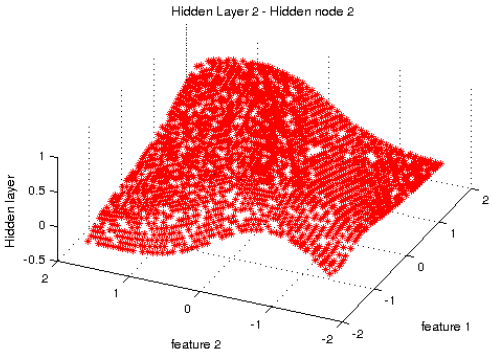
(b) Hidden layer 1, Second hidden node



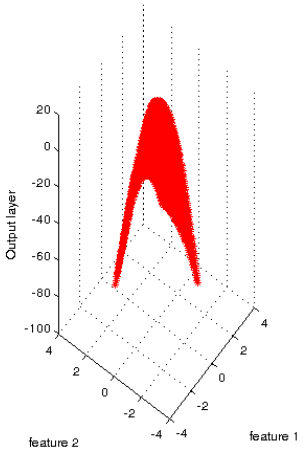
(c) Hidden layer 1, Third hidden node



(d) Hidden layer 2, First hidden node



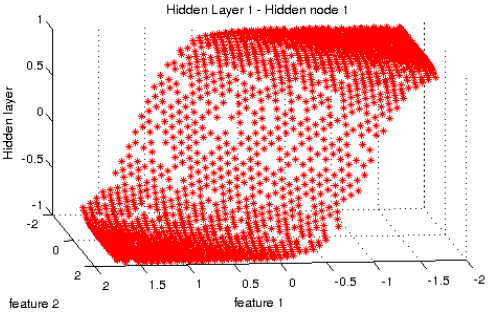
(e) Hidden layer 2, Second hidden node



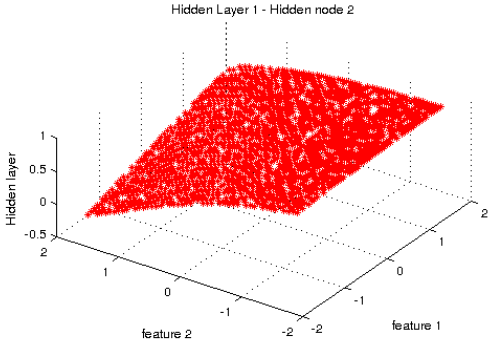
(f) Output nodes

Figure 63: Plot for hidden nodes and output node values for no of epochs =10

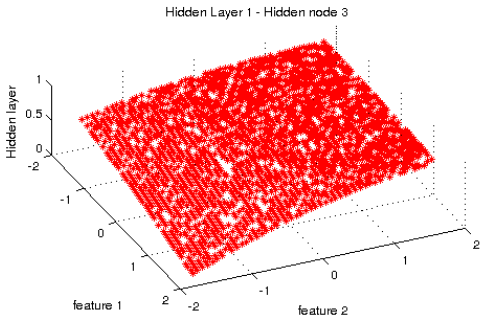
Using 2 hidden layer for no of epocs =50



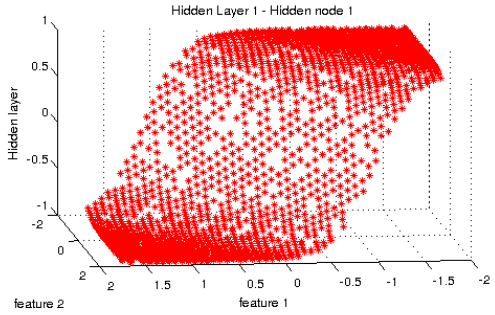
(a) Hidden layer 1 ,First hidden node



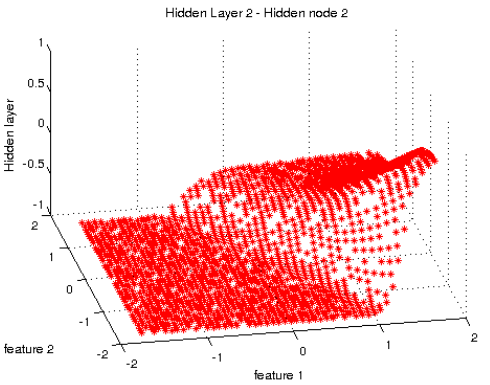
(b) Hidden layer 1, Second hidden node



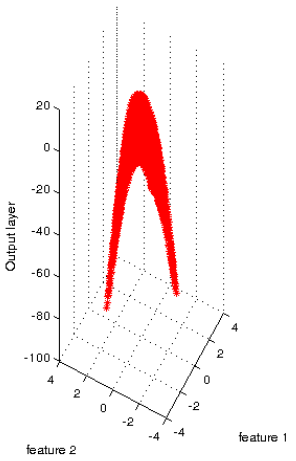
(c) Hidden layer 1, Third hidden node



(d) Hidden layer 2, First hidden node



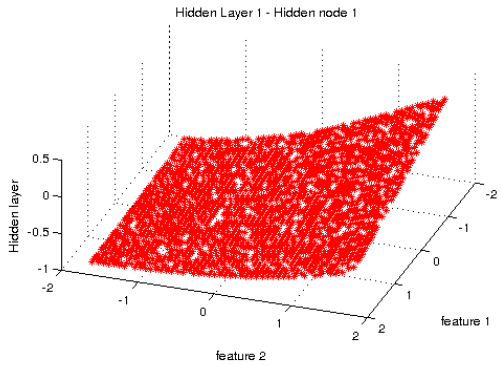
(e) Hidden layer 2, Second hidden node



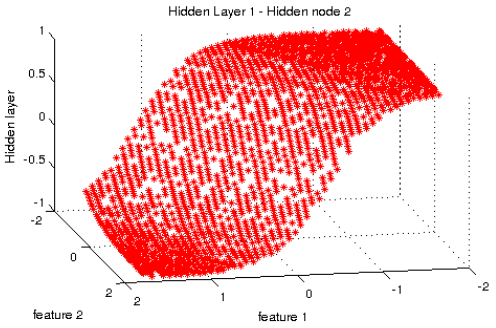
(f) Output nodes

Figure 64: Plot for hidden nodes and output node values for no of epochs =50

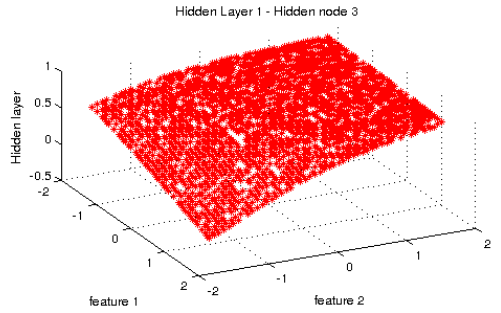
Using 2 hidden layer for no of epocs =100



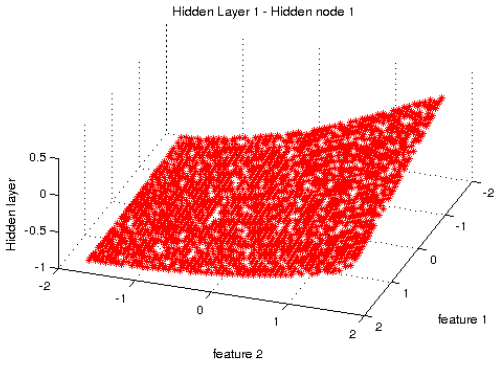
(a) Hidden layer 1 ,First hidden node



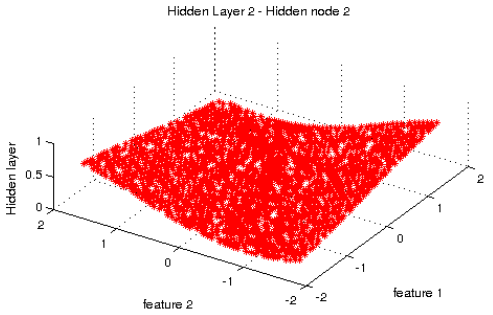
(b) Hidden layer 1, Second hidden node



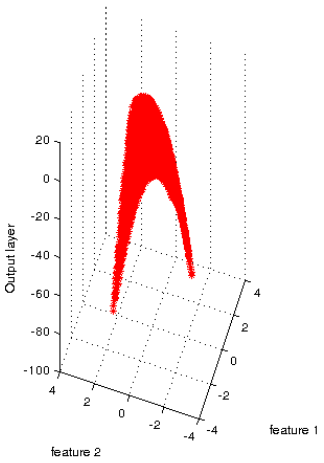
(c) Hidden layer 1, Third hidden node



(d) Hidden layer 2, First hidden node



(e) Hidden layer 2, Second hidden node



(f) Output nodes

Figure 65: Plot for hidden nodes and output node values for no of epochs =100

Inferences

- As the number of epochs increases, the output surface is becoming close to the underlying surface

3.3 Generalised RBF model

In Generalised RBF, we can use a subset of the data points itself for the RBF neurons. Selection of these RBF neurons is crucial because we can get better performance when these data points are located as diverse as possible. Also, the spread parameter should be such that interpolation can happen smoothly.

We have used K-means to find the cluster centroids for RBF neurons instead of random selection of data points. The spread is $1/2\sqrt{(M)}$ times the maximum inter-centroid distance.

3.3.1 Univariate Dataset

MSE variation with model complexities and regularisation parameter

We plotted MSE with varoius model complexities for dataset size 10 without regularization. The plot is shown below.

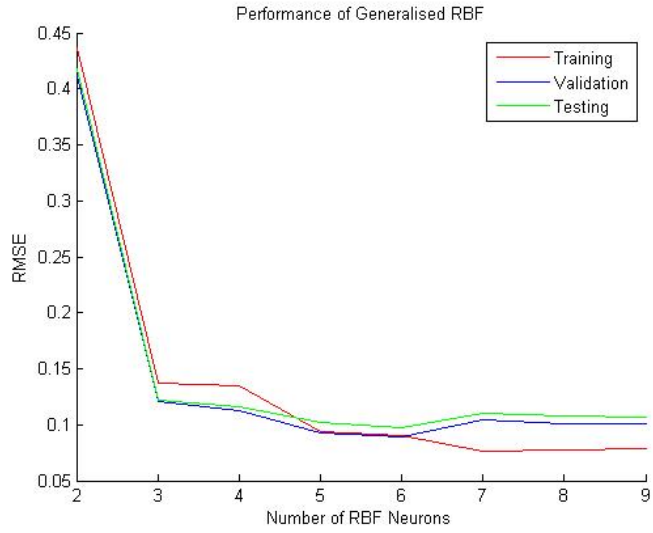


Figure 66: Performance of RBF for different numbers of RBF neurons without regularisation on N=10

Inferences

- Minimum error obtained for 6 neurons in the hidden layer
- We could see that overfitting has occurred at this stage and for further increase in number of RBF neurons, the training error goes down while the validation and training error are increasing

Then we plotted with regularization for $\lambda = 0.1$.

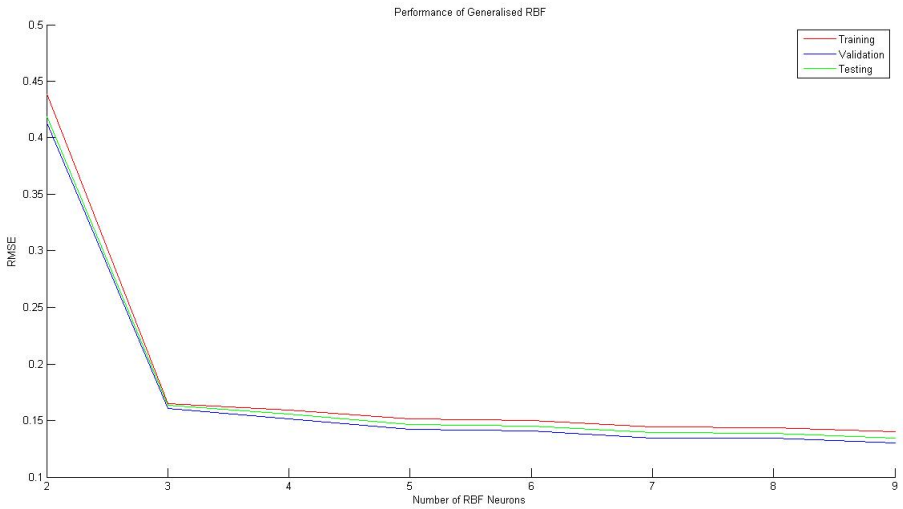


Figure 67: Performance of RBF for different numbers of RBF neurons with regularisation 0.1 on N=10

Inferences

- We can see that the addition of a small regularisation term has helped to overcome the over fitting effect.

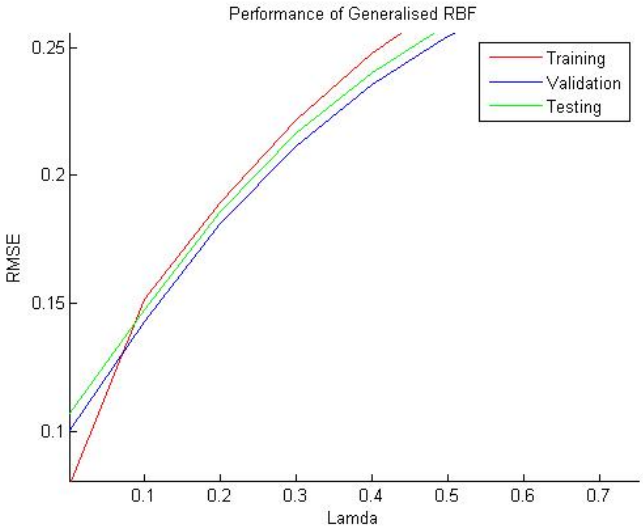


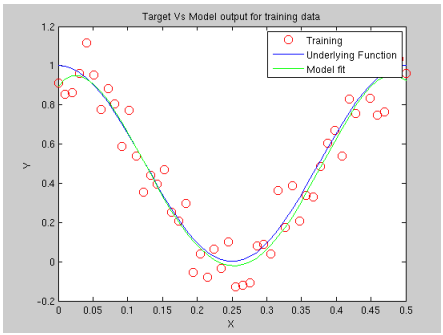
Figure 68: Performance of RBF for different values of regularisation on N=10 with 9 RBF Neurons

Inferences

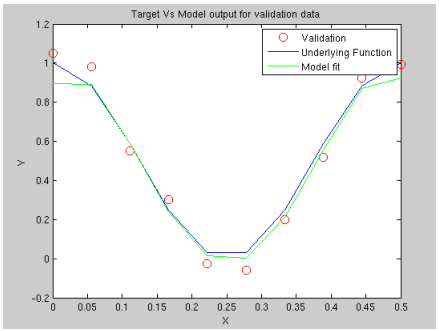
- When regularisation term is zero, the error in test and validation data are more than in training data

- When regularisation is near 0.1, the error in test data and validation data are almost same as in training data
- When regularisation is crossing 0.1, the error is getting increased continuously.

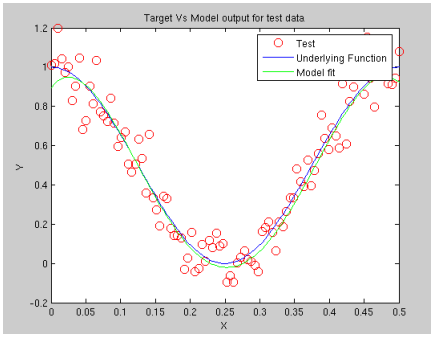
Plots of Model output Vs Target output



(a) Training data

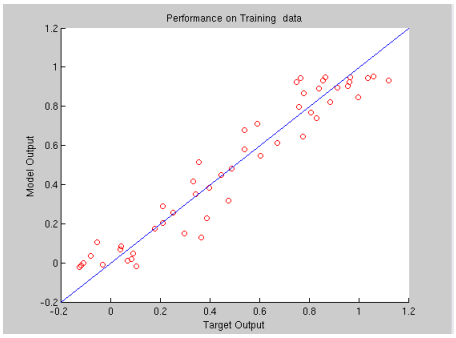


(b) Validation data

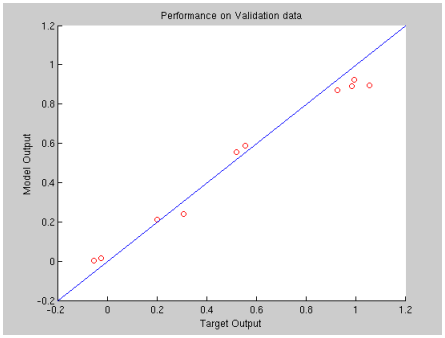


(c) Test data

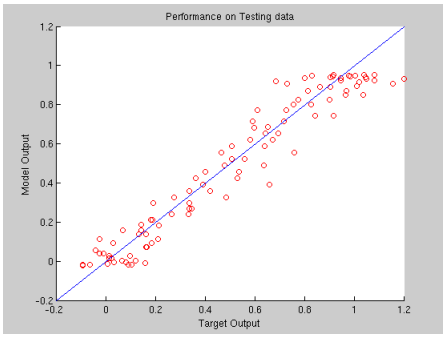
Figure 69: Model output Vs Target output for N=100 , 4 RBF Neurons



(a) Training data



(b) Validation data



(c) Test data

Figure 70: scatter plot with target output on x-axis and model output on y-axis for training data, validation data and test data with training data size=100

Inferences

We can see that most of the scatter points are aound the x=y line showing good accuracy

3.3.2 Bivariate Dataset

MSE variation with model complexities and regularisation parameter

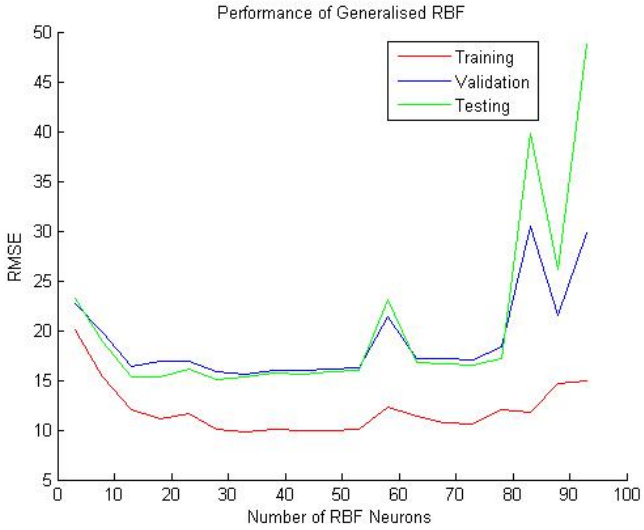


Figure 71: Performance of RBF for different numbers of RBF neurons without regularisation on N=100

Inferences

- We can that as the model complexity, i.e. the number of RBF neurons increase in number, the RMSE on training data keeps decreasing while that on the test and validation error goes up after M=80

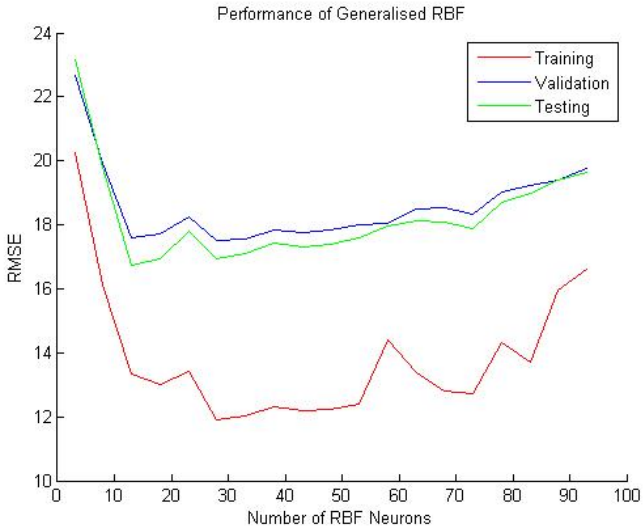


Figure 72: Performance of RBF for different numbers of RBF neurons with regularisation 0.9 on N=100

Inferences

- Adding a regularisation term of 0.9 on the training sample size N=100 gives better performance of the model on test data even for higher complexities

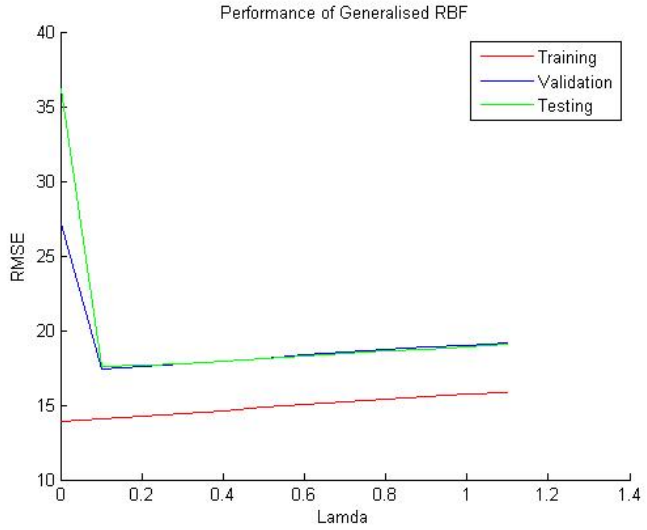


Figure 73: Performance of RBF for different values of regularisation on N=100 with 99 RBF neurons

Inferences

- Regularisation value of 0.1 has the most effect on reducing overfitting with M=99 RBF neurons for training sample of N=100.
- With regularisation, the variance of the model has come down.

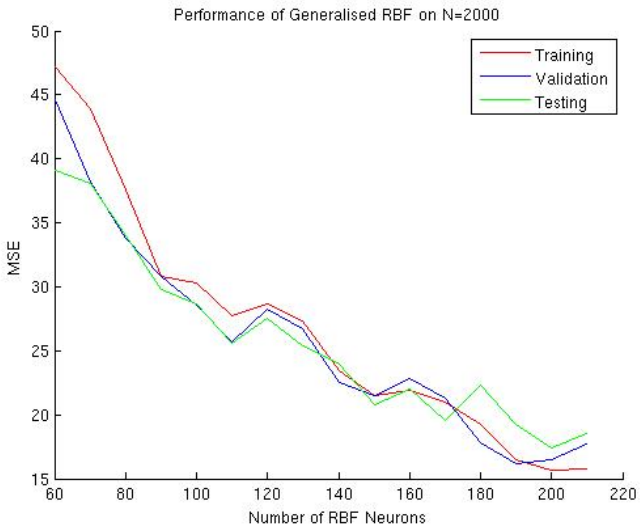
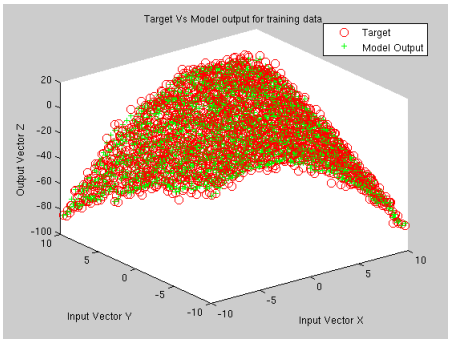


Figure 74: Performance of RBF Vs Number of RBF Neurons on N=2000

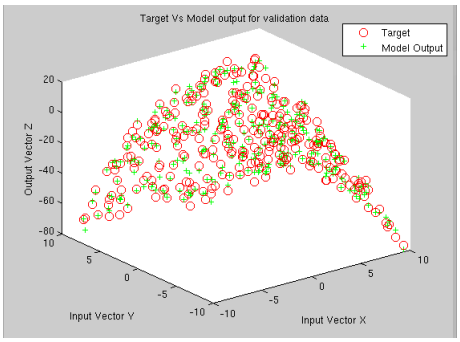
Inferences

- Best model performance of 16.9 MSE is achieved by choosing M=190 based on validation data performance

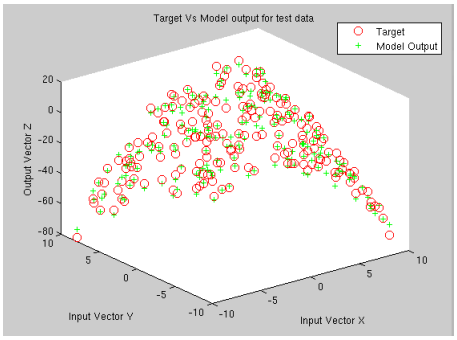
Plots of Model output Vs Target output



(a) Training data



(b) Validation data

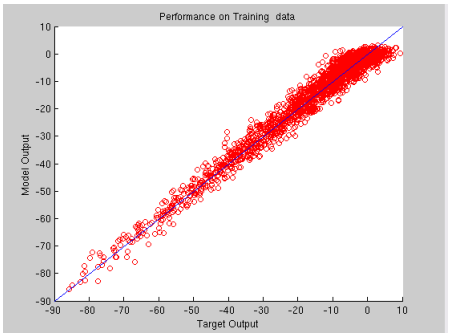


(c) Test data

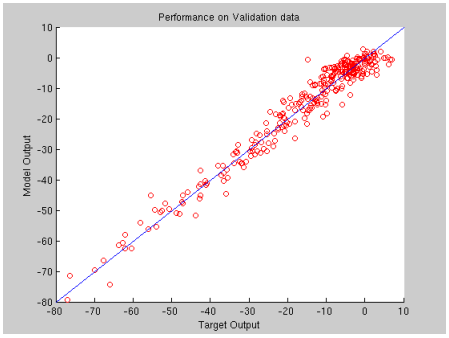
Figure 75: Model output Vs Target output for N=2000 , RBF Neurons= 190

Inferences

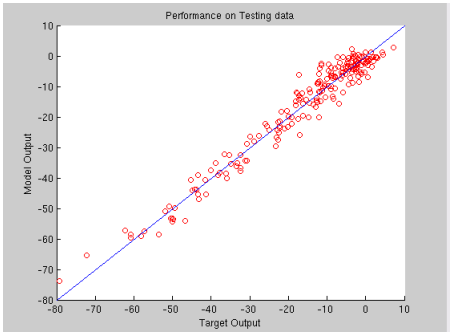
- Best accuracy of MSE 16.9 is obtained with 190 RBF neurons.
- This is performing better than gradient descent MLFNN on small size training sets because we are using regularisation in RBFNN.
- The newrb module in matlab could give the same accuracy with 69 neurons itself. Selection of RBF neurons in newrb module is based on picking the data point with maximum error in each iteration.



(a) Training data



(b) Validation data



(c) Test data

Figure 76: scatter plot with target output on x-axis and model output on y-axis for training data, validation data and test data with training data size=2000

- Most of the scatter plots are close to the x=y line.