Kernel Method for Pattern Analysis

CS6011

IIT Madras

Assignment 1

By:
Group 6:
Arun Baby (CS15S016)
Vishal Subbiah (MM12B035)
Nikhil Thomas Stephen (CS15M032)

Contents

1 Classification **2**

1.1 Models . . . 2

1.1.1 Multilayer Feedforward Neural Network . . . 2

1.2 Datasets . . . 2

1.2.1 Linearly separable data . . . 2

1.2.2 Nonlinearly separable data . . . 2

1.2.3 Overlapping data . . . 2

1.2.4 Linearly separable data . . . 2

1.2.5 Image data . . . 2

1.3 Experimentations and Plots . . . 3

1.3.1 Linearly separable classes . . . 3

1.3.2 Non-linearly separable classes . . . 5

1.3.3 Overlapping classes . . . 21

1.3.4 Image data . . . 23

1.4 Observations and Inferences . . . 24

2 Regression **25**

2.1 Models . . . 25

2.1.1 Polynomial Curve Fitting . . . 25

2.1.2 Regularization . . . 26

2.1.3 Linear model for regression . . . 26

2.1.4 Multilayer Feedforward Neural Network . . . 27

2.1.5 Radial Basis Function Neural Network (RBFNN) . . . 28

2.2 Datasets . . . 28

2.2.1 Dataset 1: 1-dimensional (Univariate) input data . . . 28

2.2.2 Dataset 2: 2-dimensional (Bivariate) input data . . . 28

2.3 Experimentations and Plots . . . 29

2.3.1 Polynomial curve fitting for dataset 1 . . . 29

2.3.2 Regression using Gaussian basis functions for dataset 2 . . . 35

2.3.3 MLFFNN with 1 hidden layer for dataset 1 . . . 41

2.3.4 MLFFNN with 2 hidden layers for Dataset 2 . . . 43

2.3.5 Generalized RBF model for Datasets 1 . . . 50

2.3.6 Generalized RBF model for Datasets 2 . . . 52

2.4 Observations and Inferences . . . 52

1 Classification

Classification is a general process related to categorization, the process in which ideas and objects are recognized, differentiated and understood.

1.1 Models

1.1.1 Multilayer Feedforward Neural Network

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 are using cross entropy function for computing the error.

1.2 Datasets

1.2.1 Linearly separable data

The dataset is a 2 dimensional linearly separable data. It has 3 classes of 500 samples each. We used 50% for training, 30% for validation and 20% for testing.

1.2.2 Nonlinearly separable data

The dataset is a 2 dimensional nonlinearly separable data. It has 3 classes of 300, 600 and 900 samples each. We used 50% for training, 30% for validation and 20% for testing.

1.2.3 Overlapping data

The dataset is a 2 dimensional overlapping separable data. It has 4 classes of 500 samples each. We used 50% for training, 30% for validation and 20% for testing.

1.2.4 Linearly separable data

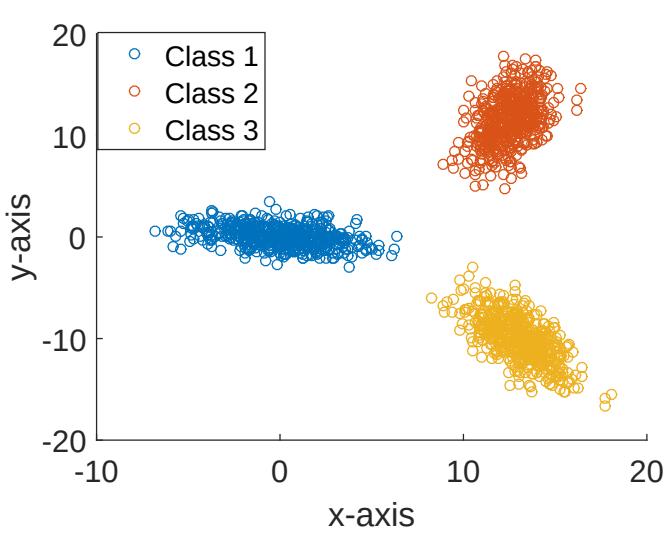
The dataset is a 2 dimensional linearly separable data. It has 3 classes of 500 samples each. We used 50% for training, 30% for validation and 20% for testing.

1.2.5 Image data

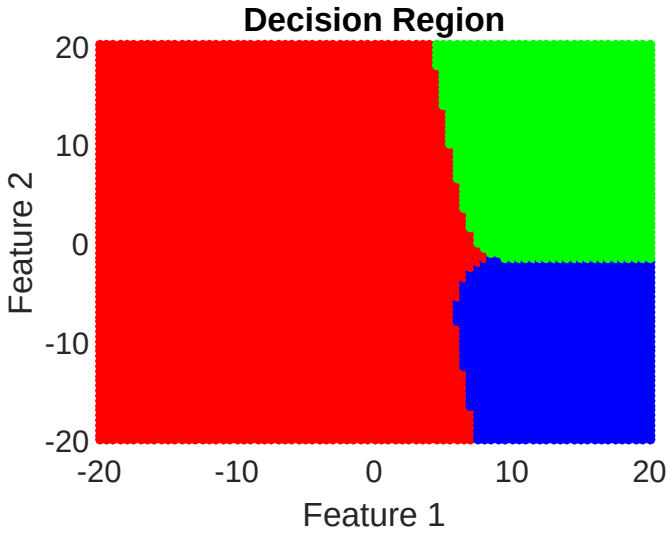
The dataset consist of images of thread mills, mattresses, binoculars, ladders, hot tubs. We used 50% for training, 30% for validation and 20% for testing.

1.3 Experimentations and Plots

1.3.1 Linearly separable classes



(a) Scatter Plot



(b) Decision Regions

Figure 1

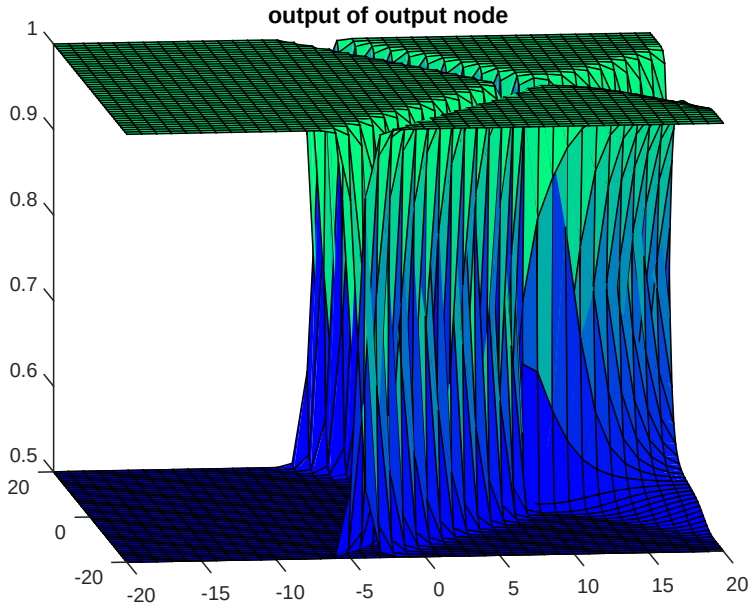


Figure 2: Output of all output nodes

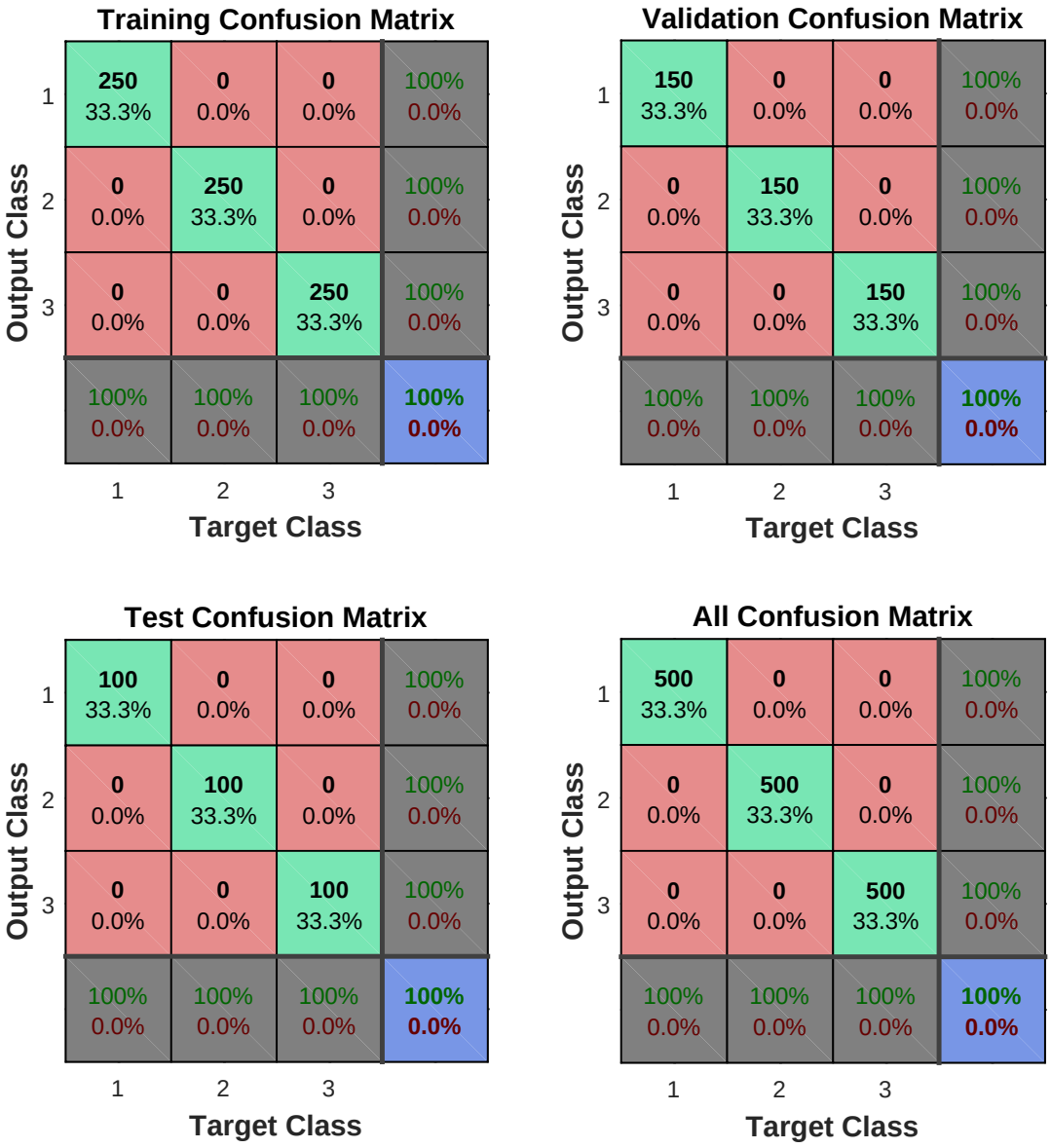
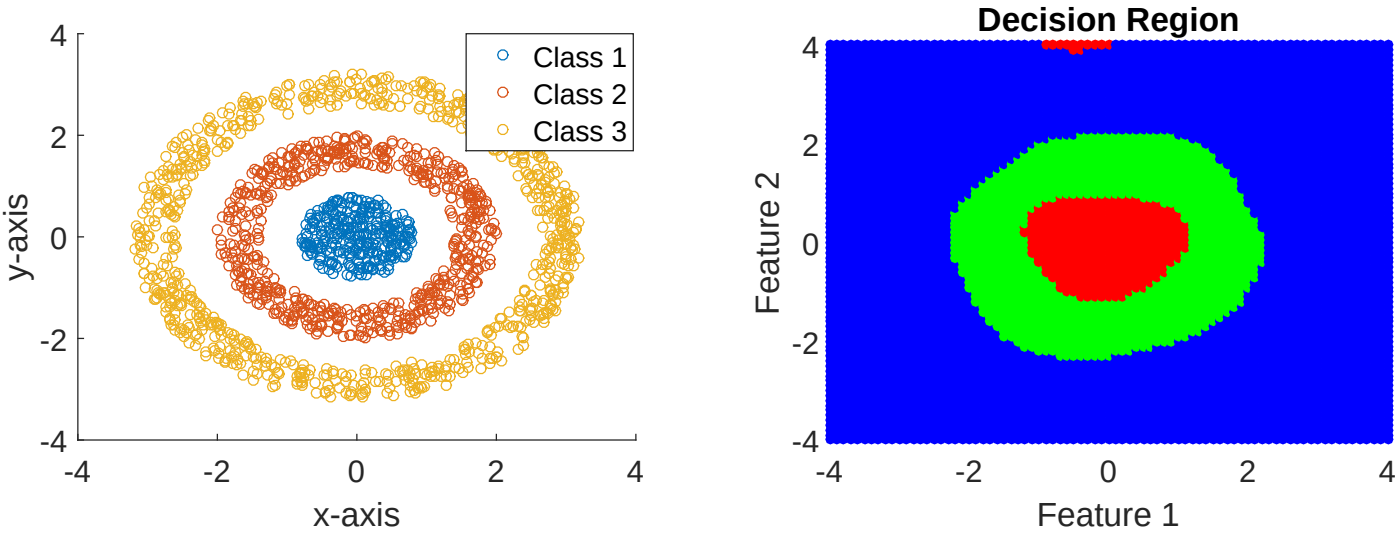


Figure 3: Confusion matrix with 2 hidden layers of 3 nodes each

1.3.2 Non-linearly separable classes



(a) Scatter Plot

(b) Decision Regions

Figure 4

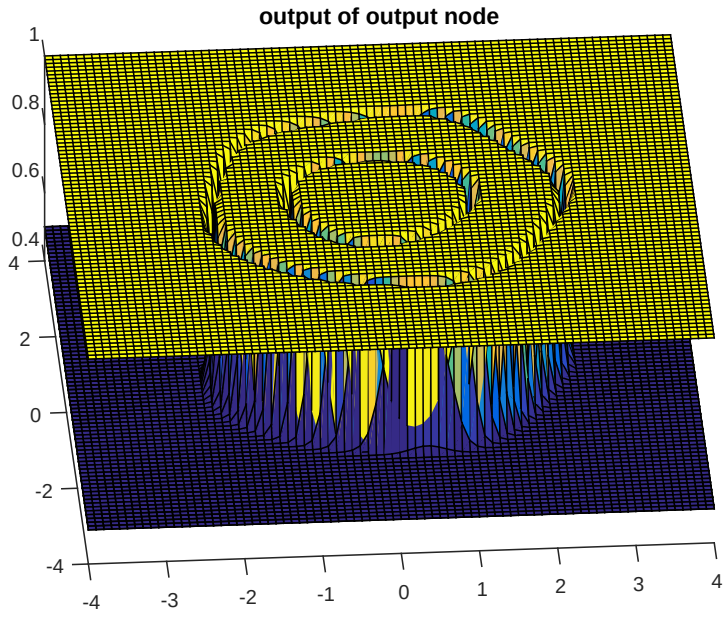
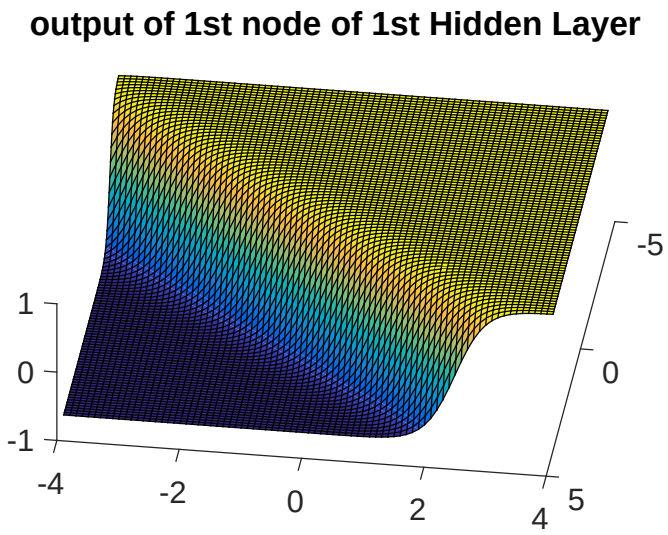
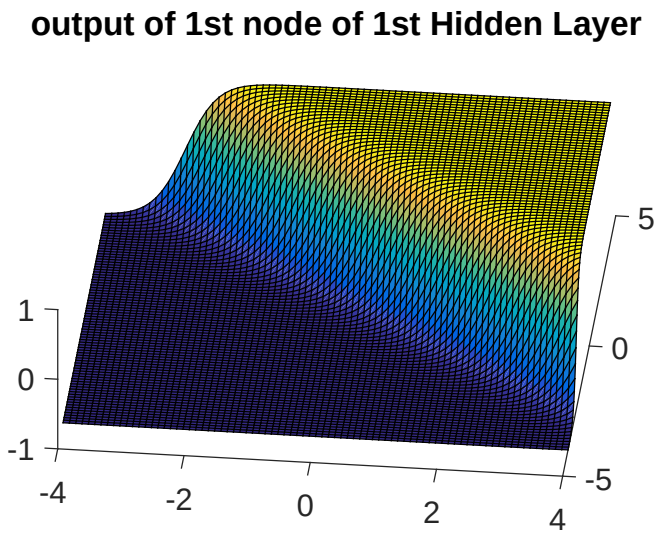


Figure 5: Output of all output nodes

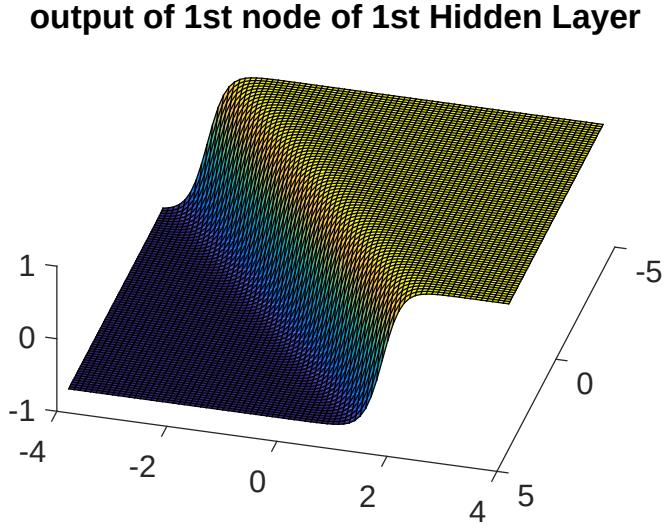
Figure 6: Confusion matrix with 2 hidden layers of 4 nodes each



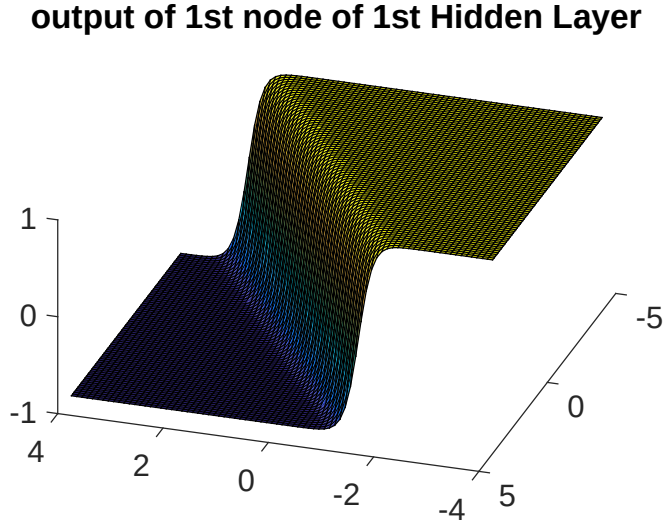
(a) Epoch 1



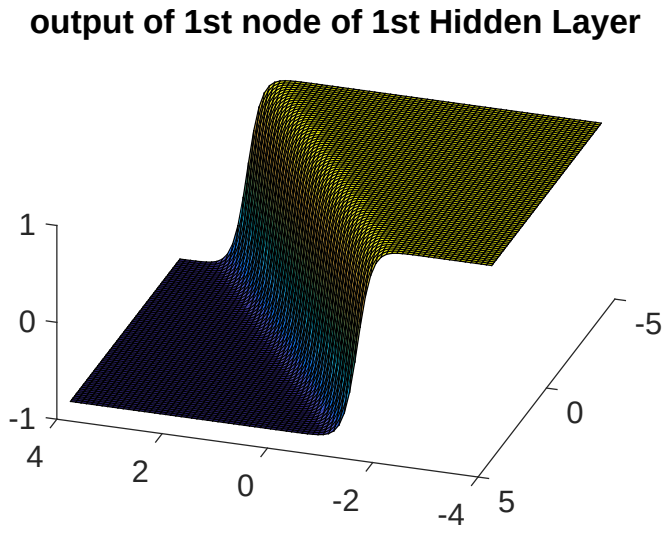
(b) Epoch 2



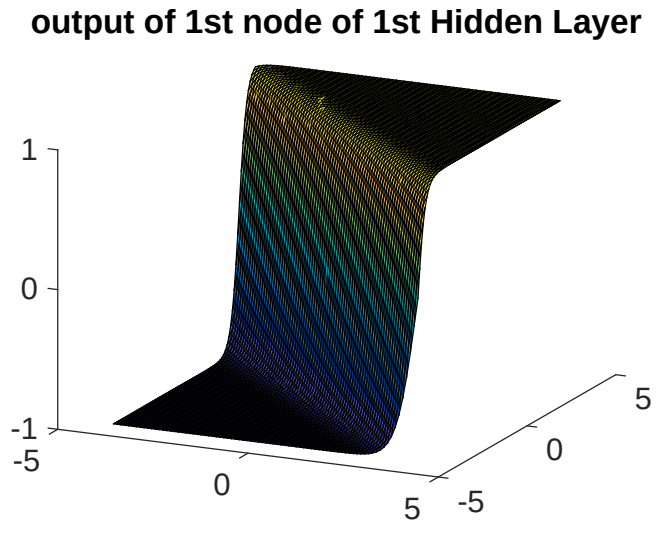
(c) Epoch 10



(d) Epoch 50

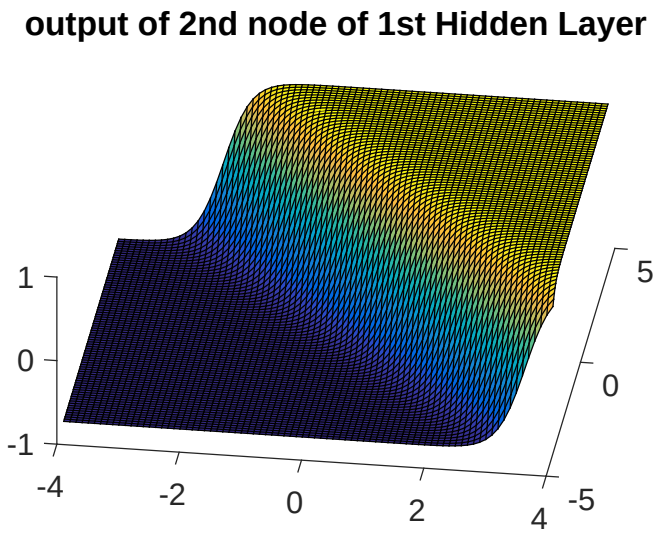


(e) Epoch 100

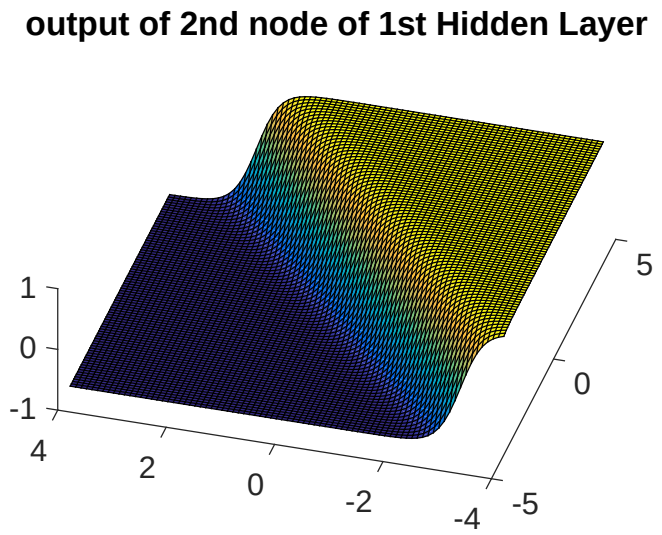


(f) Epoch 154

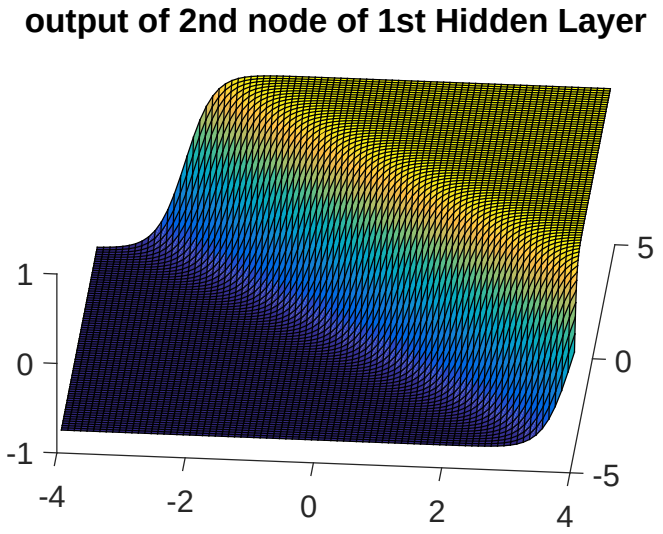
Figure 7



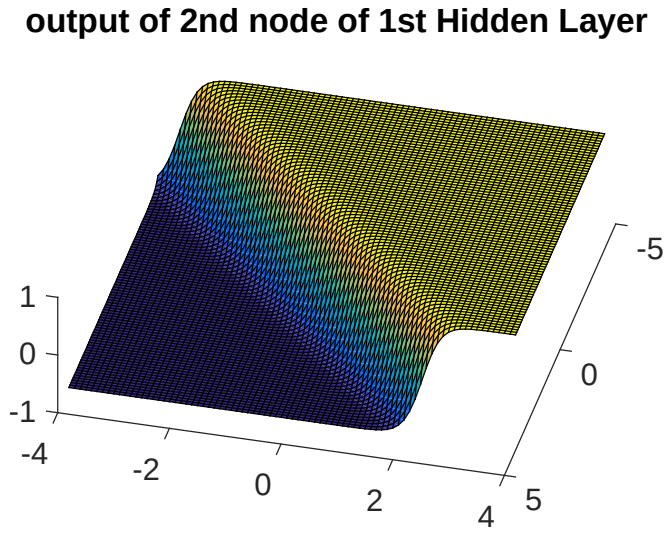
(a) Epoch 1



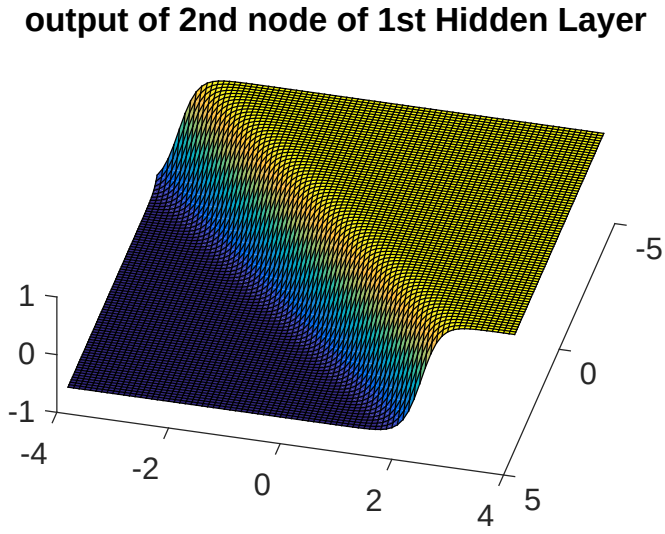
(b) Epoch 2



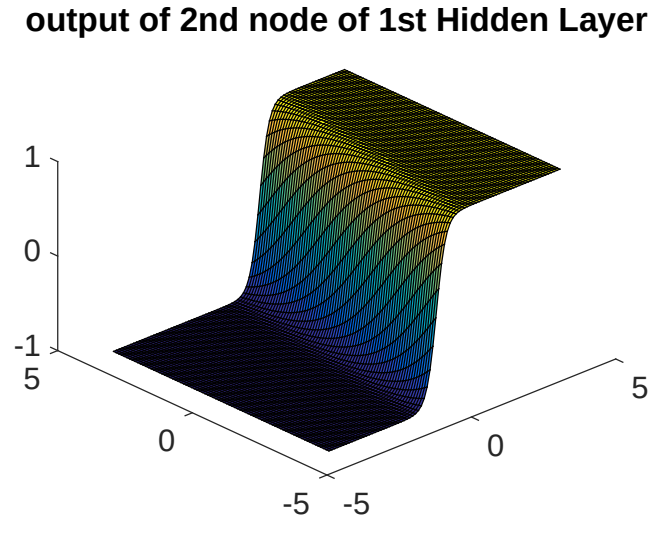
(c) Epoch 10



(d) Epoch 50

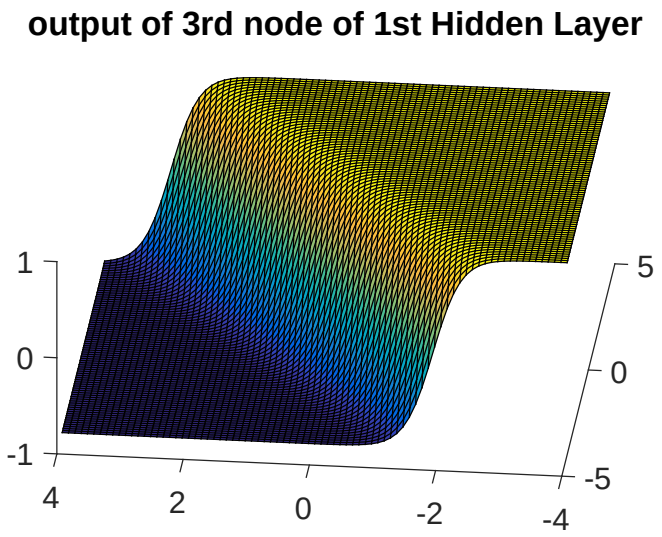


(e) Epoch 100

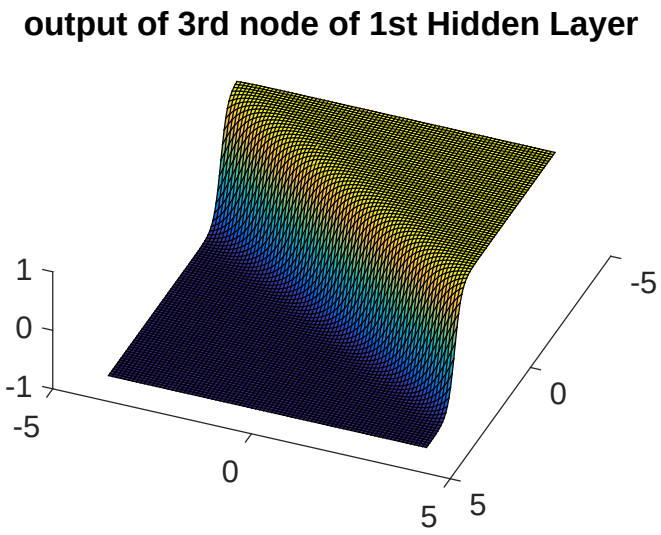


(f) Epoch 154

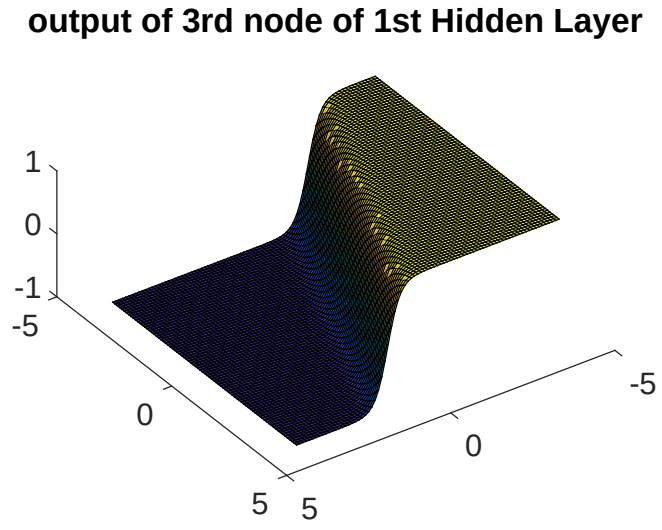
Figure 8



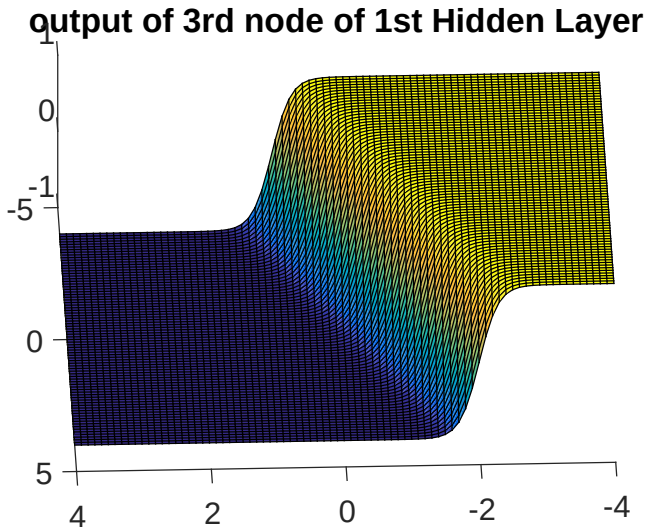
(a) Epoch 1



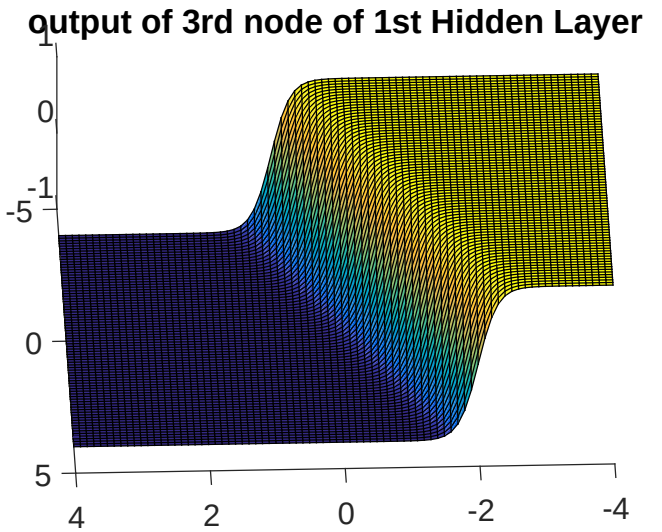
(b) Epoch 2



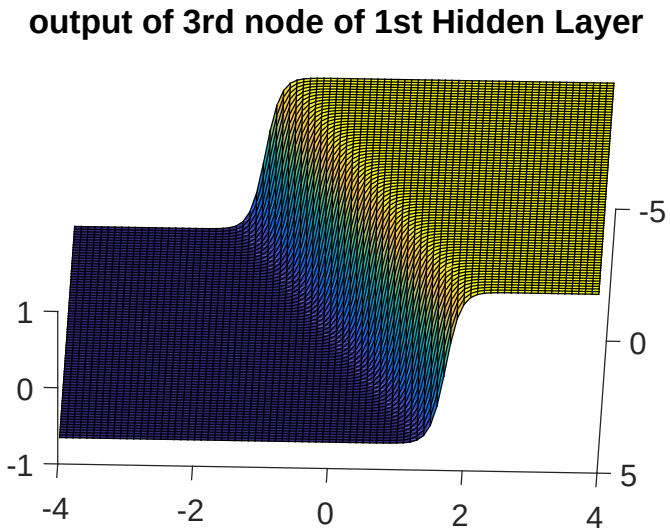
(c) Epoch 10



(d) Epoch 50

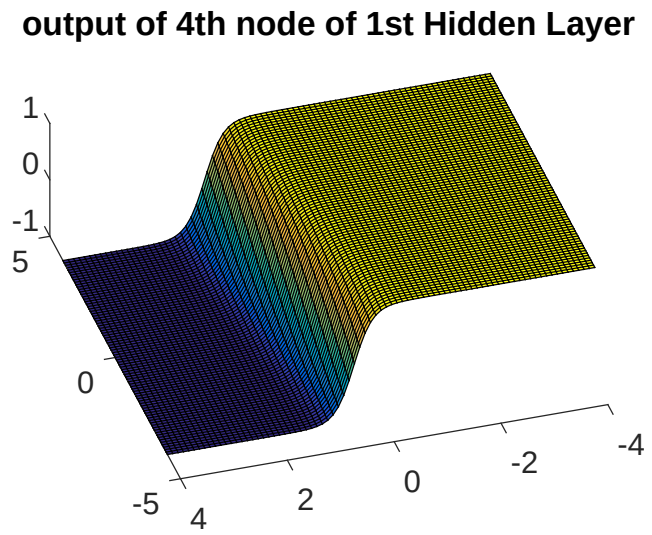


(e) Epoch 100

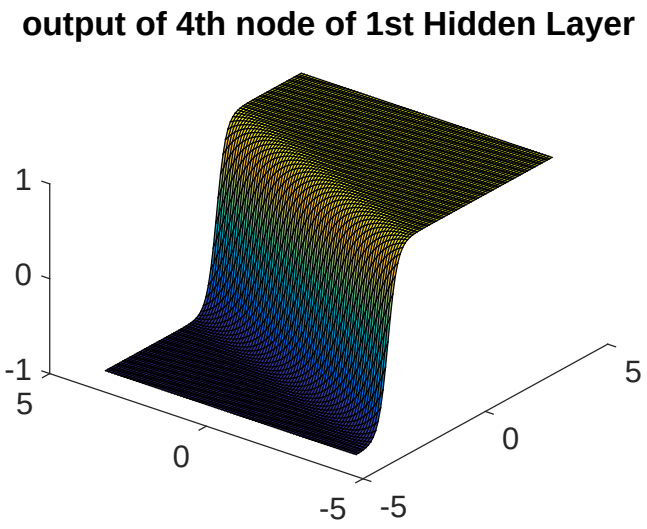


(f) Epoch 154

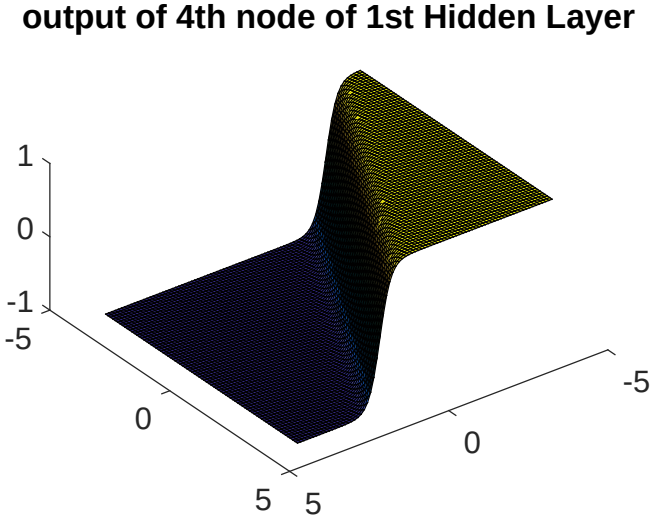
Figure 9



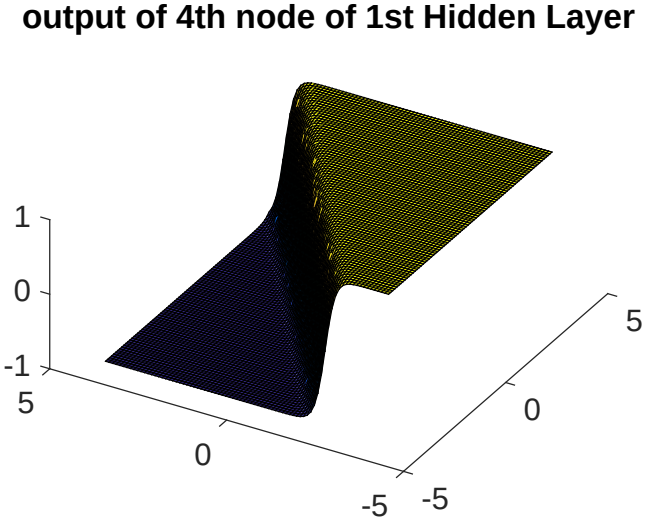
(a) Epoch 1



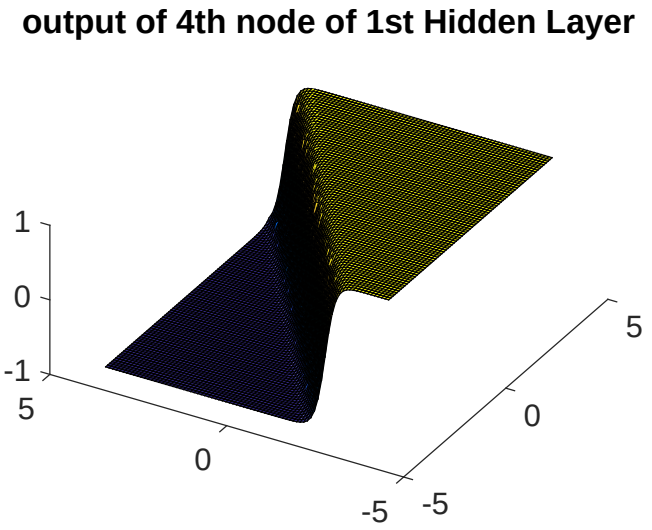
(b) Epoch 2



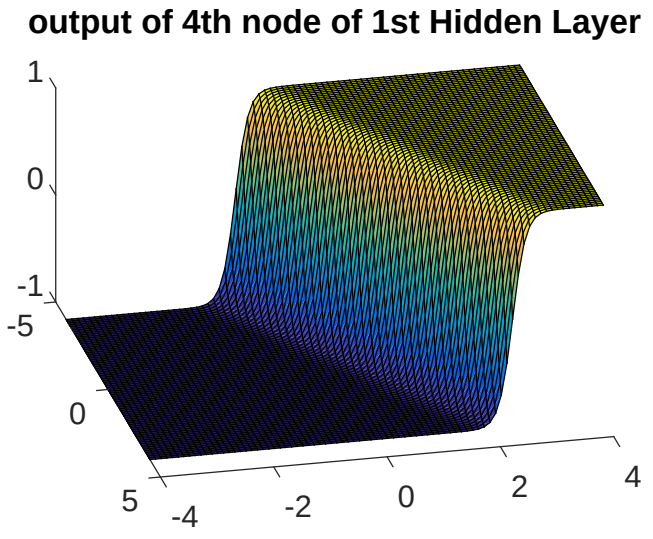
(c) Epoch 10



(d) Epoch 50

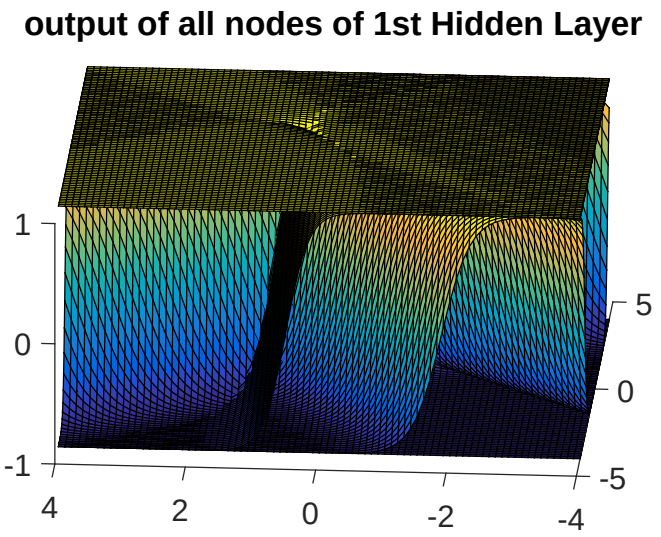


(e) Epoch 100

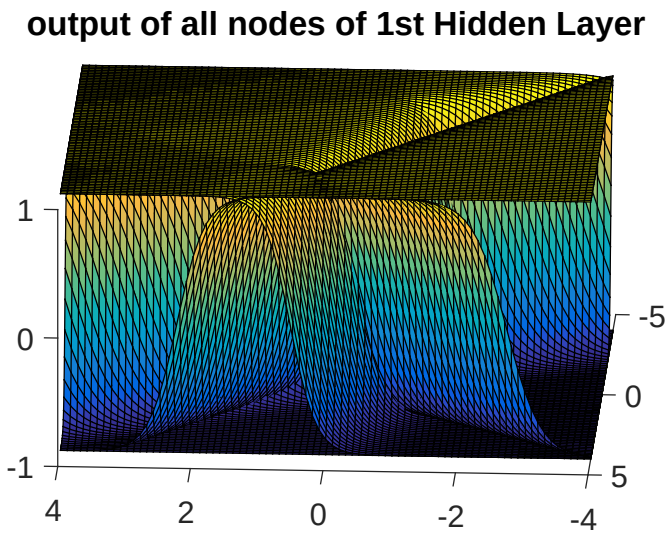


(f) Epoch 154

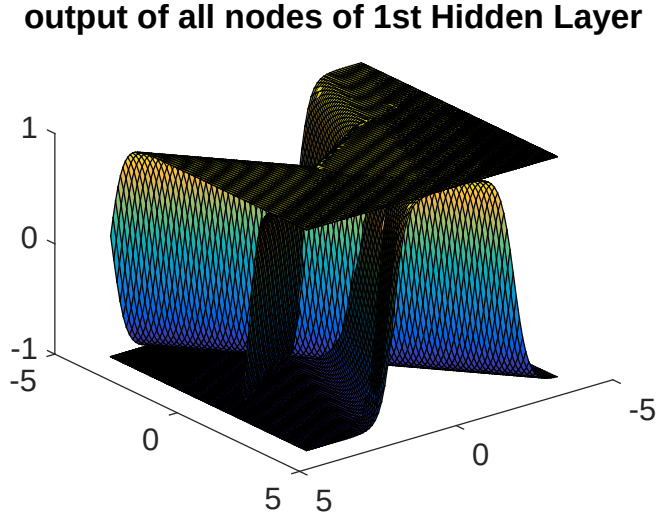
Figure 10



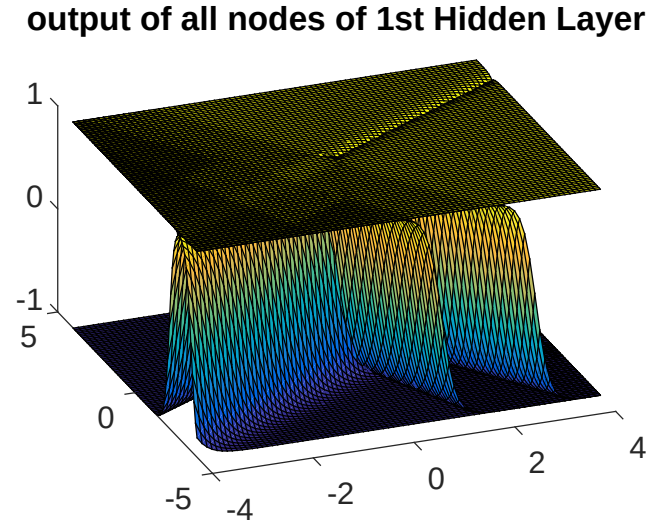
(a) Epoch 1



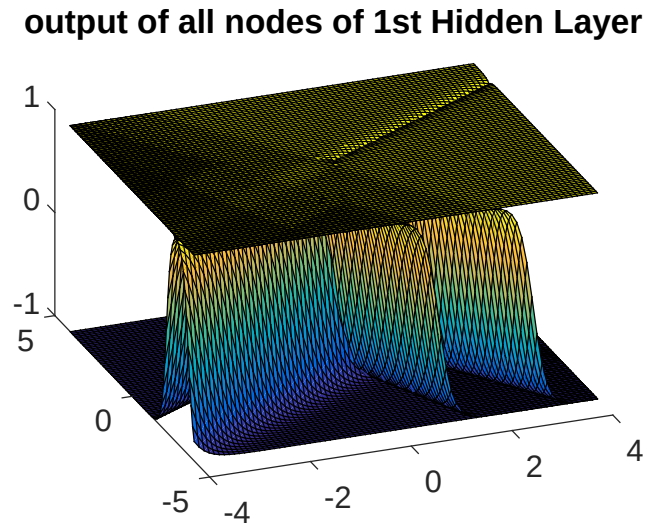
(b) Epoch 2



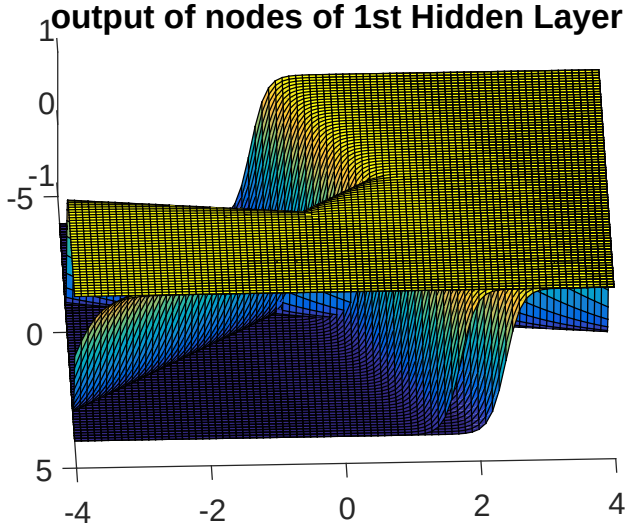
(c) Epoch 10



(d) Epoch 50

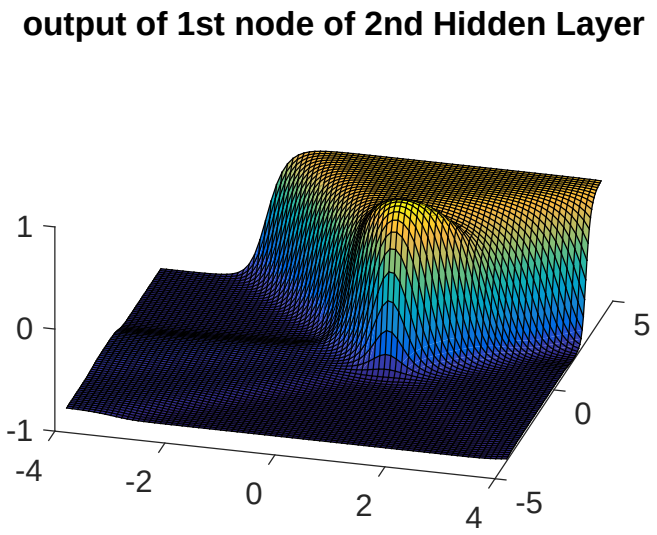


(e) Epoch 100

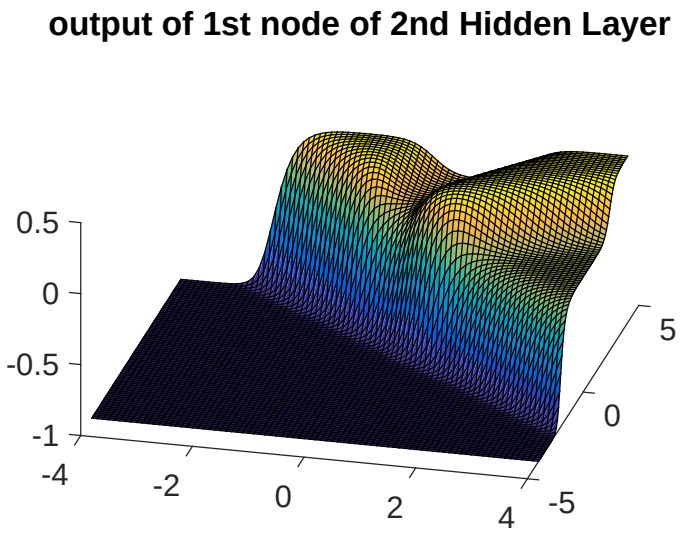


(f) Epoch 154

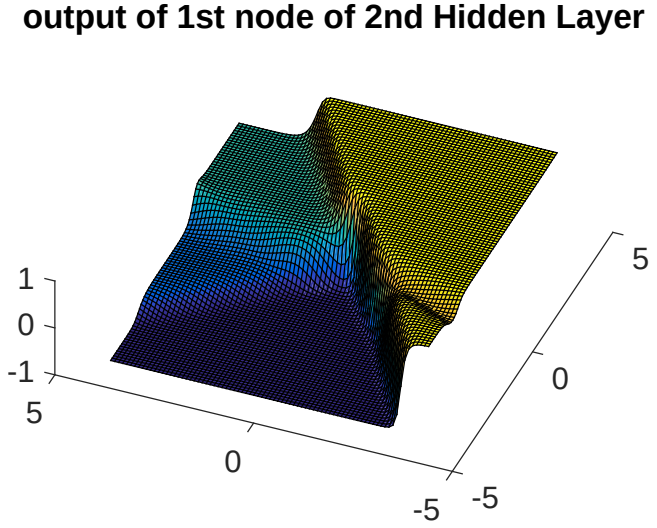
Figure 11



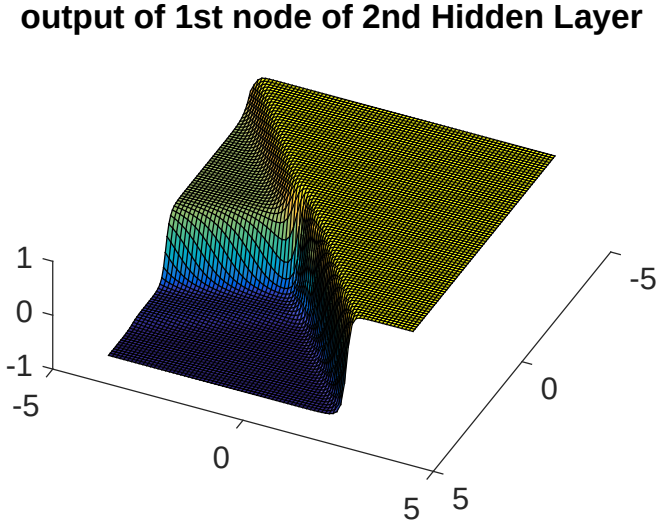
(a) Epoch 1



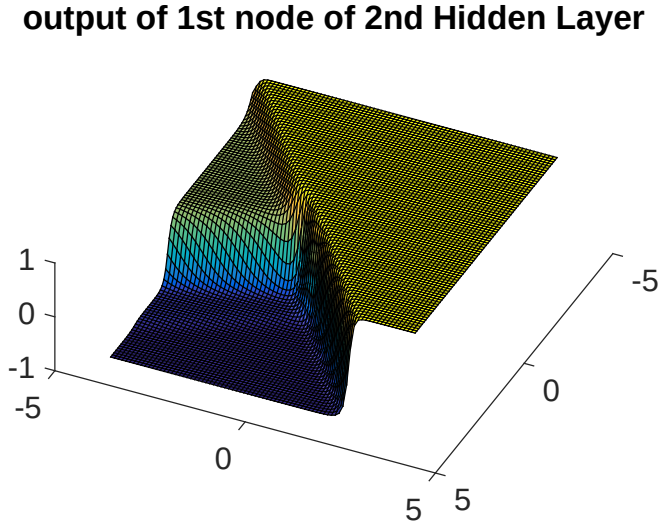
(b) Epoch 2



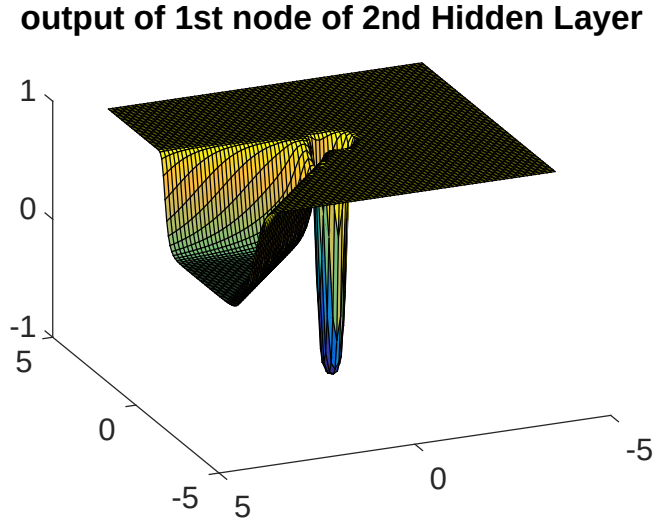
(c) Epoch 10



(d) Epoch 50

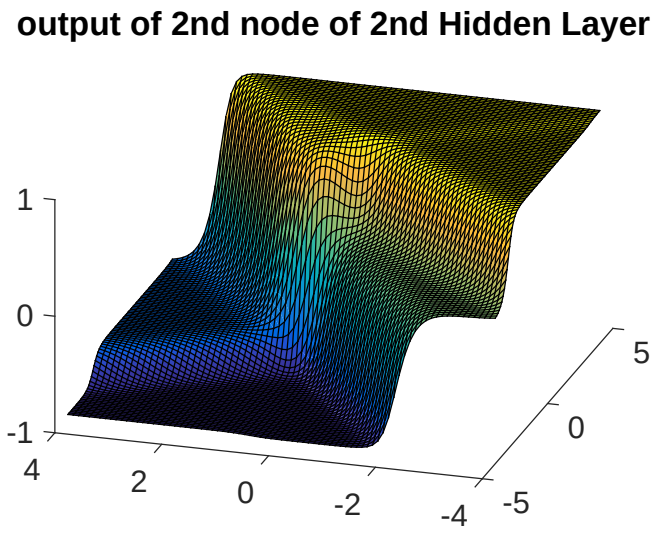


(e) Epoch 100

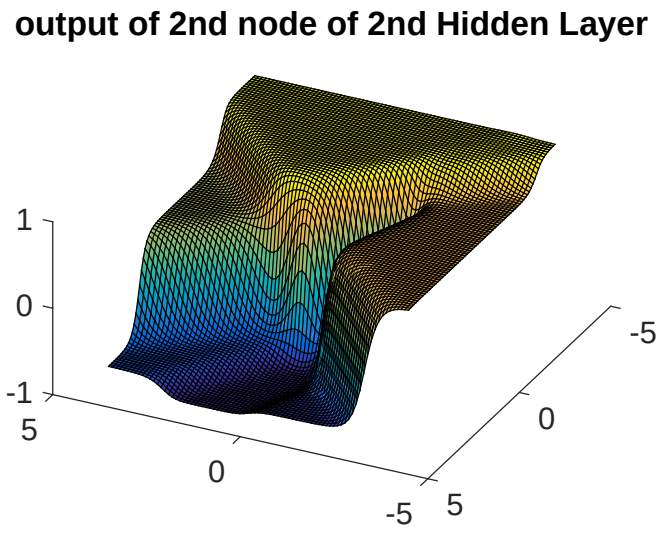


(f) Epoch 154

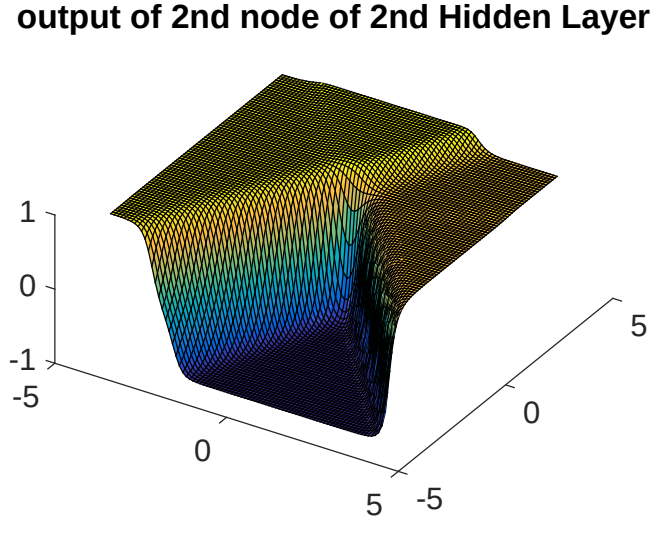
Figure 12



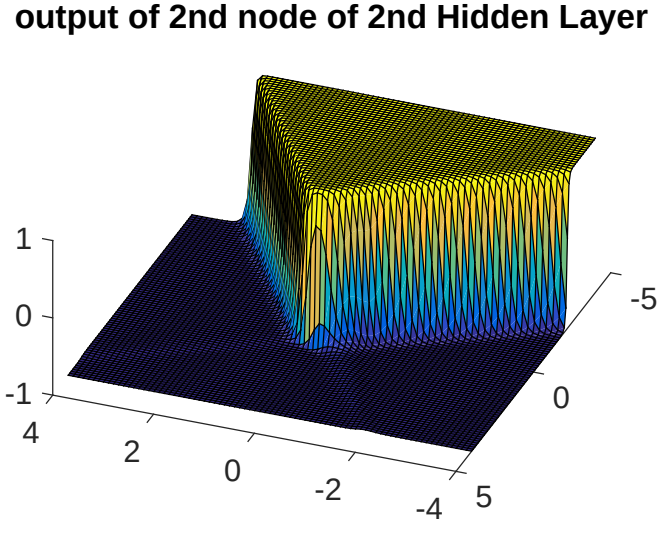
(a) Epoch 1



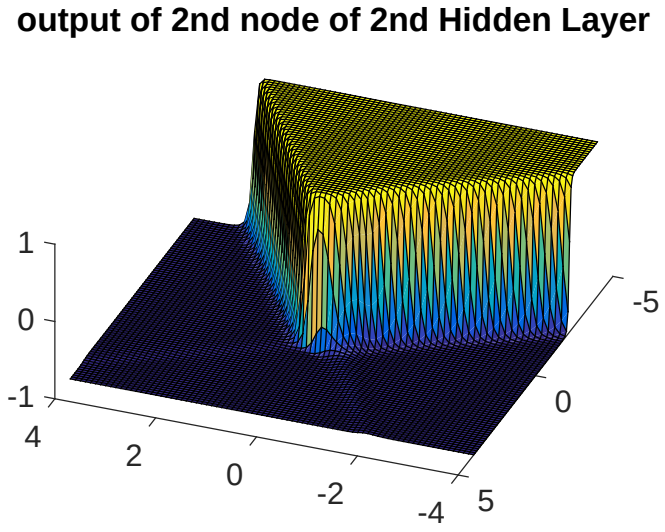
(b) Epoch 2



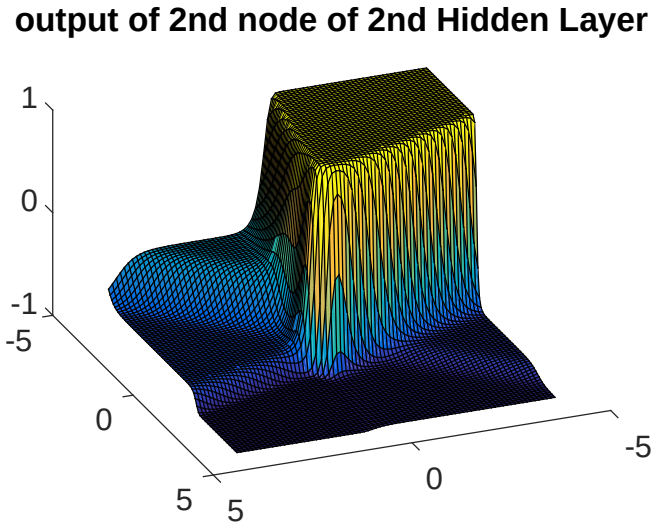
(c) Epoch 10



(d) Epoch 50

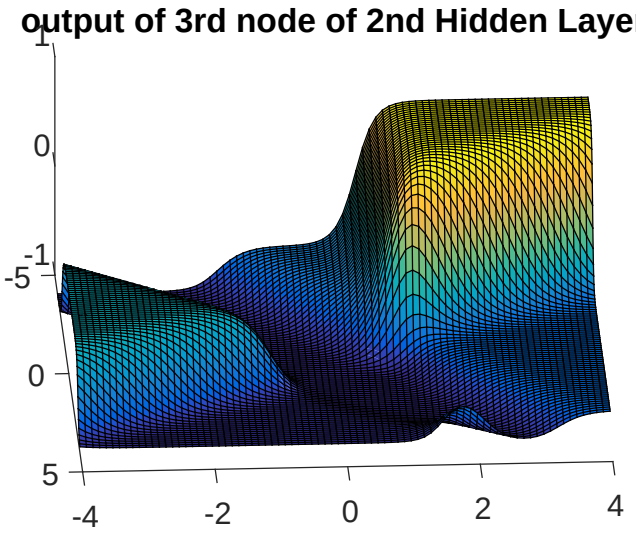


(e) Epoch 100

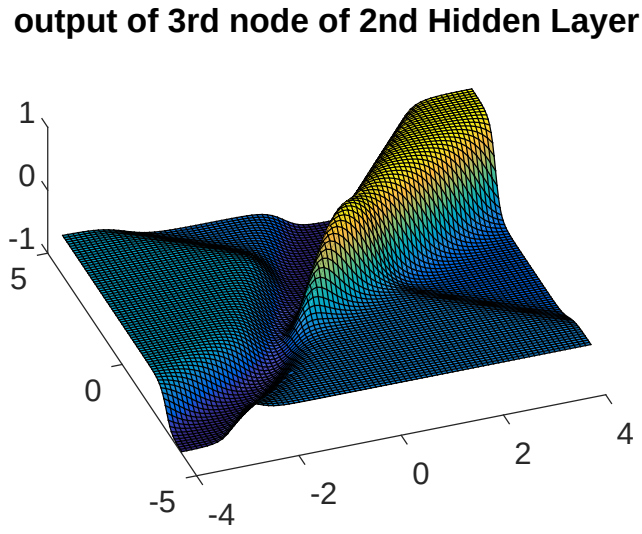


(f) Epoch 154

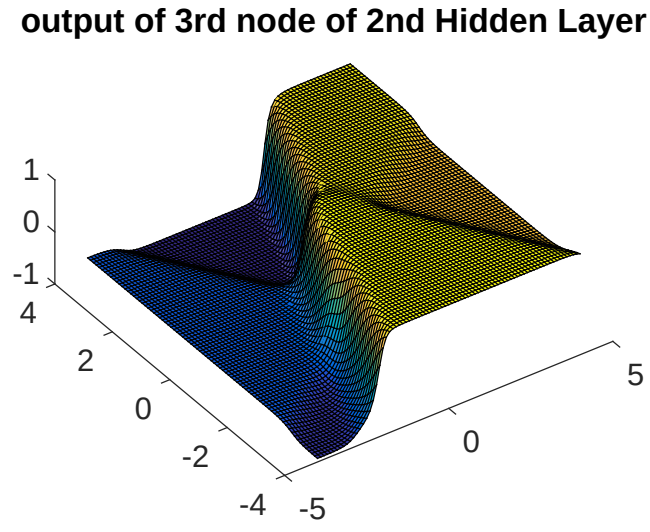
Figure 13



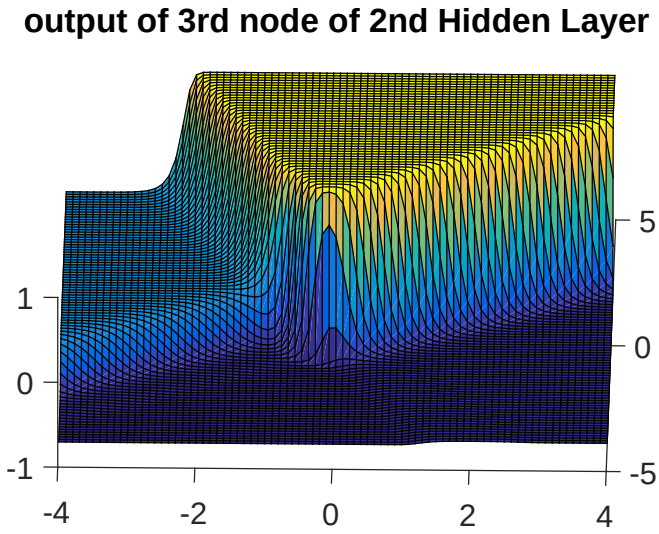
(a) Epoch 1



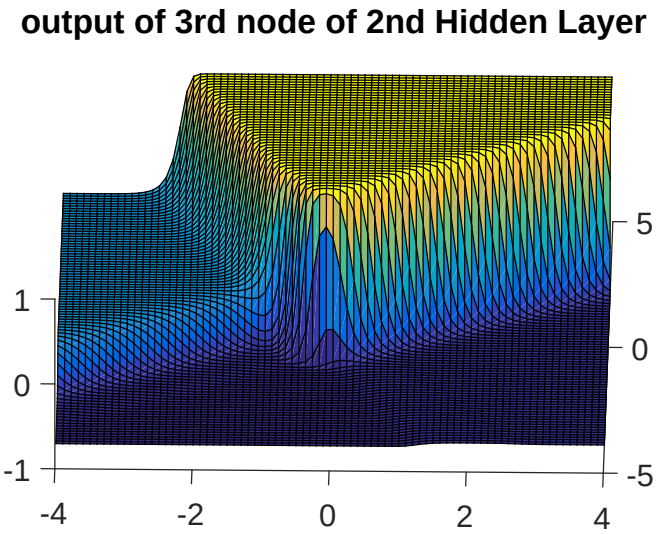
(b) Epoch 2



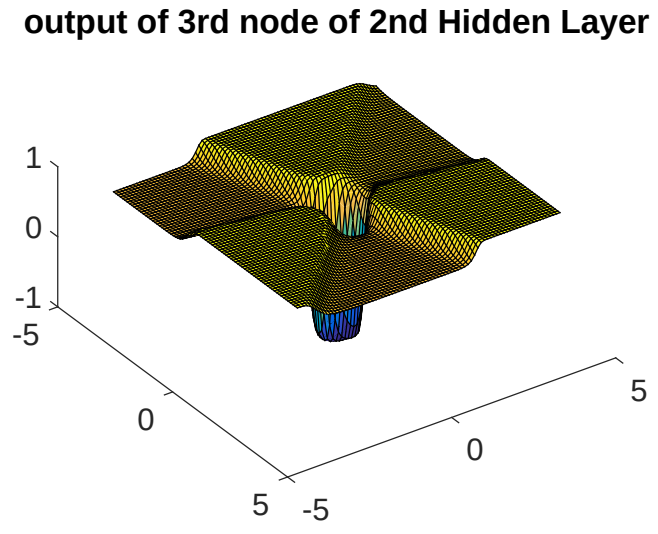
(c) Epoch 10



(d) Epoch 50

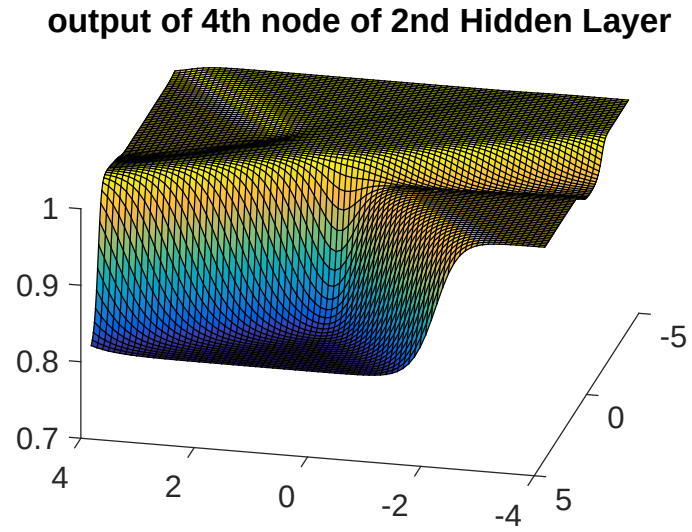


(e) Epoch 100

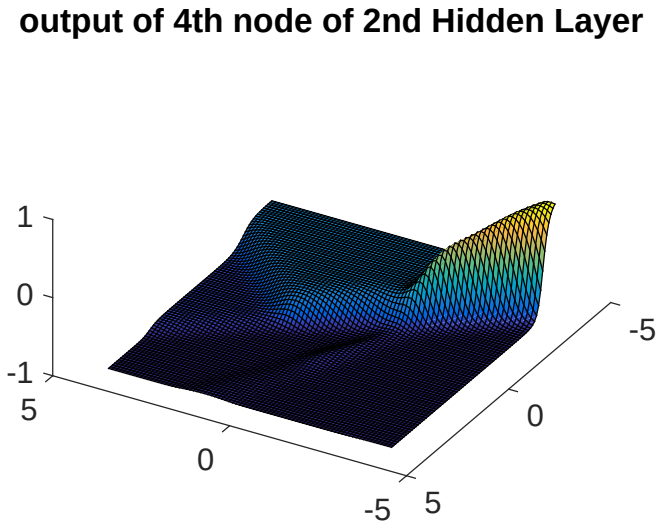


(f) Epoch 154

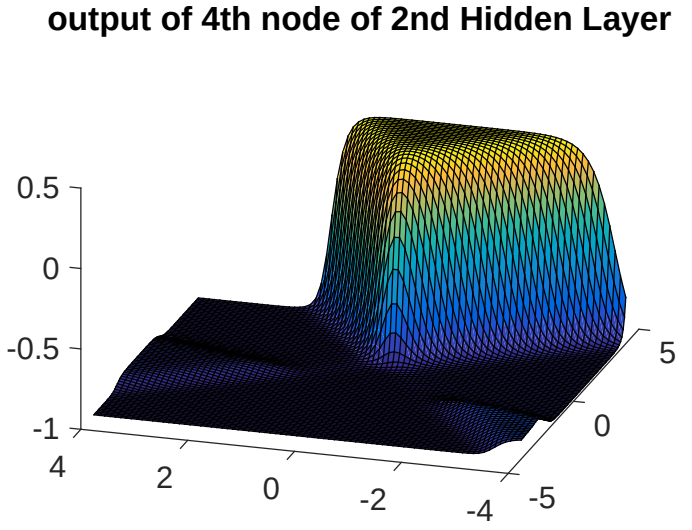
Figure 14



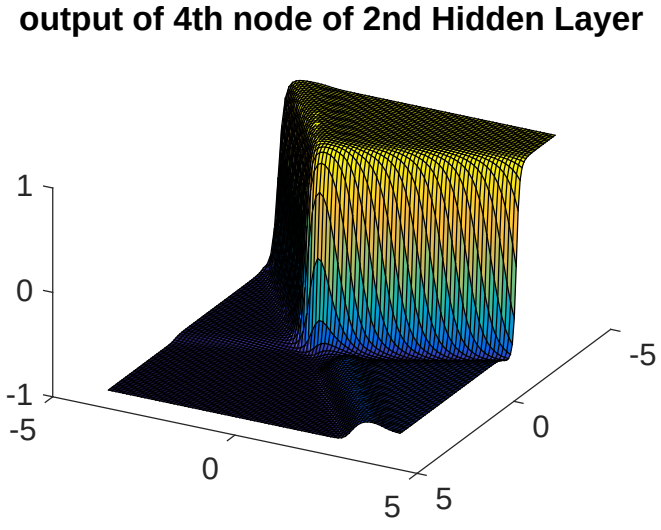
(a) Epoch 1



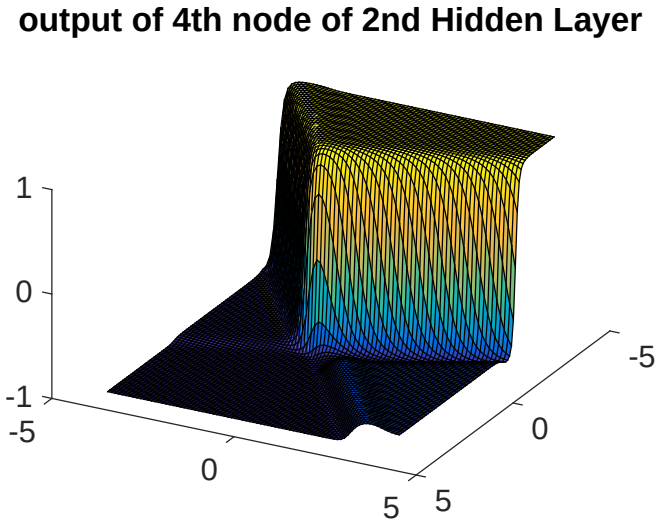
(b) Epoch 2



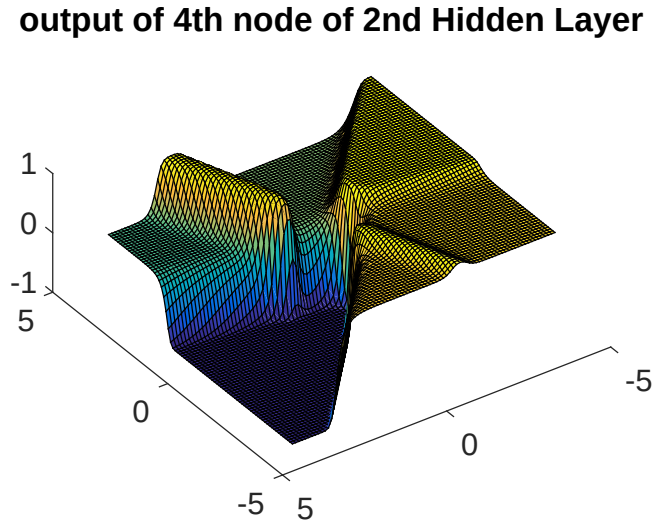
(c) Epoch 10



(d) Epoch 50

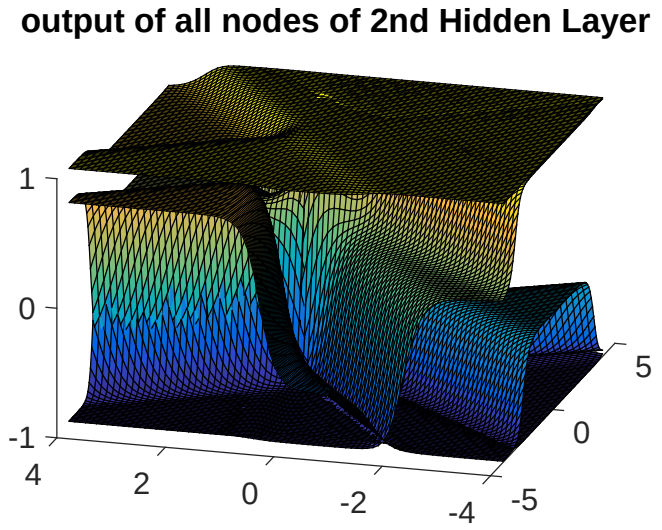


(e) Epoch 100

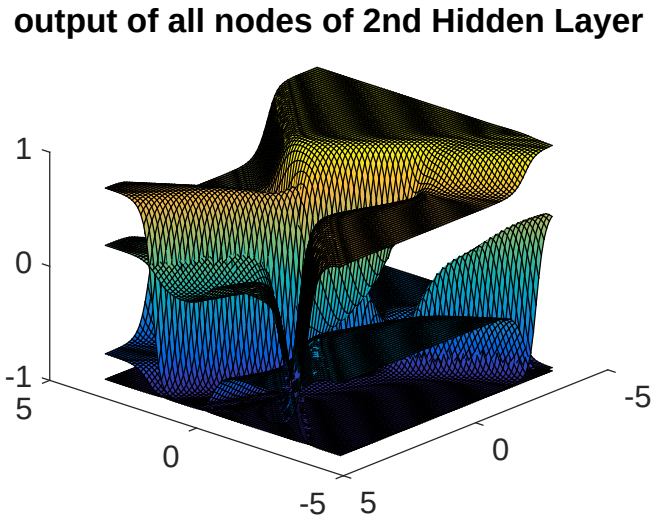


(f) Epoch 154

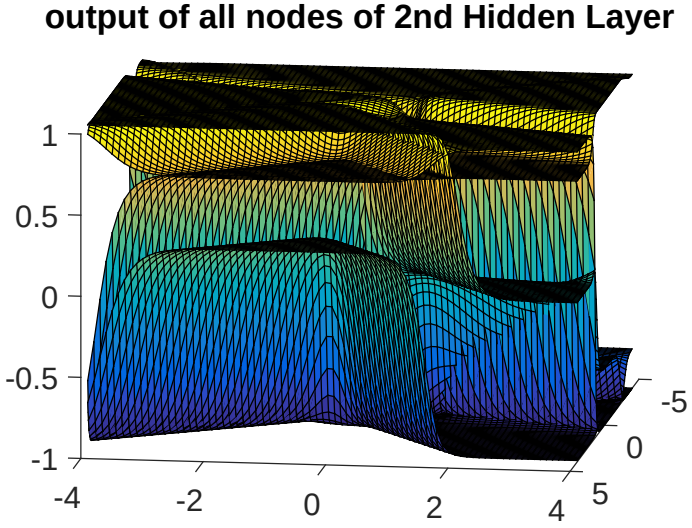
Figure 15



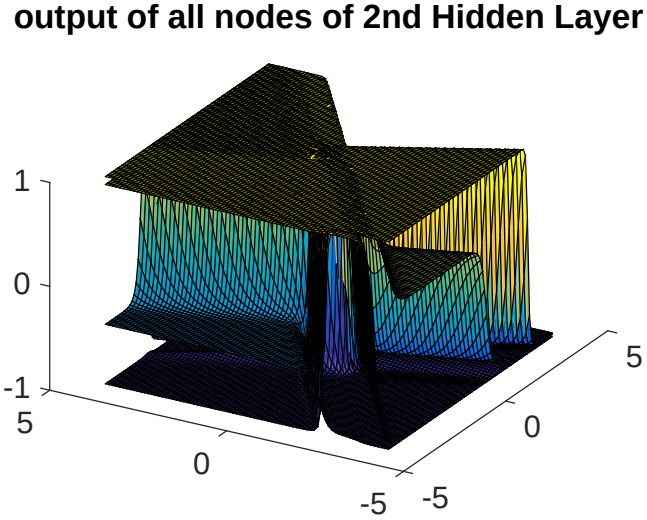
(a) Epoch 1



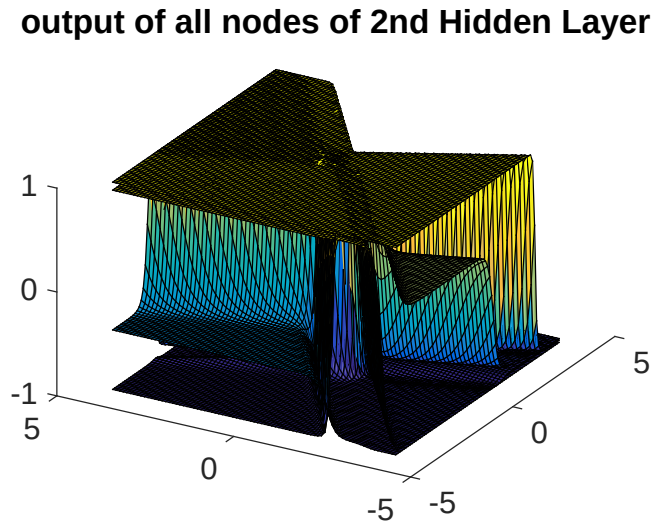
(b) Epoch 2



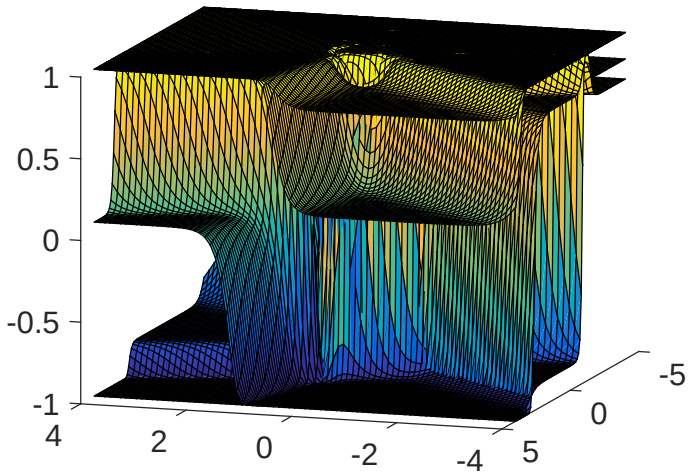
(c) Epoch 10



(d) Epoch 50

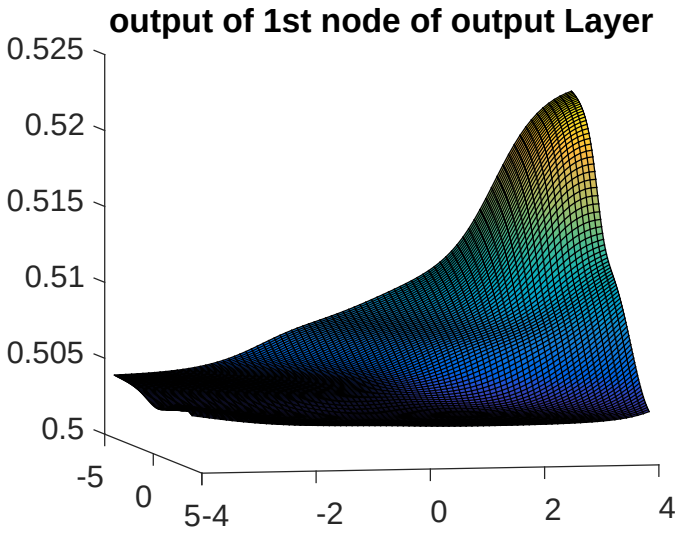


(e) Epoch 100

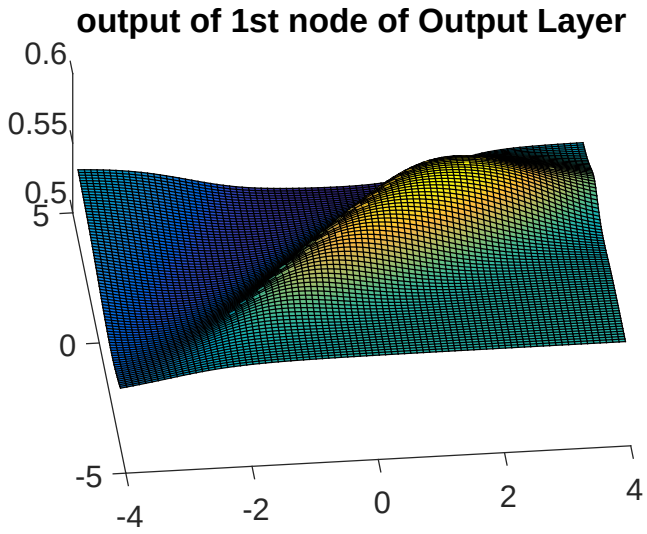


(f) Epoch 154

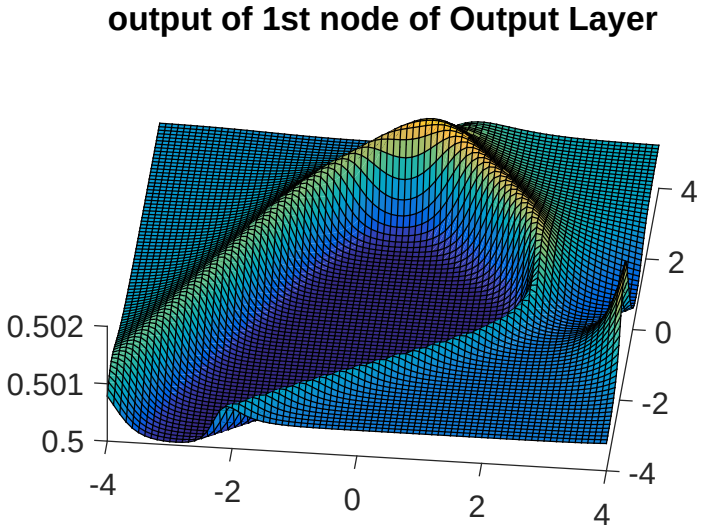
Figure 16



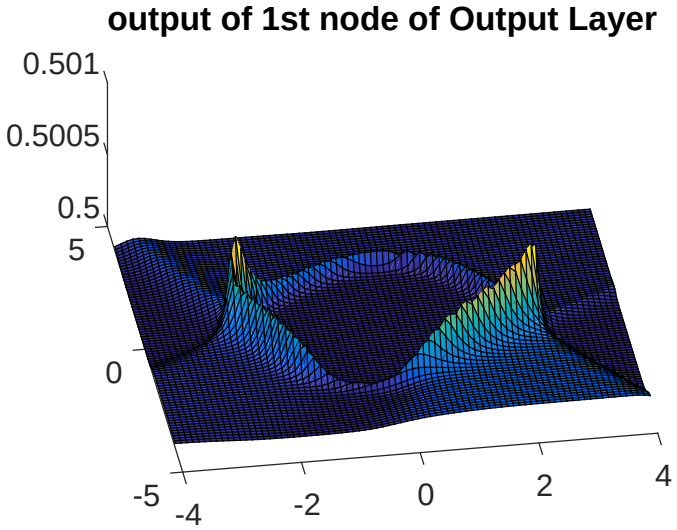
(a) Epoch 1



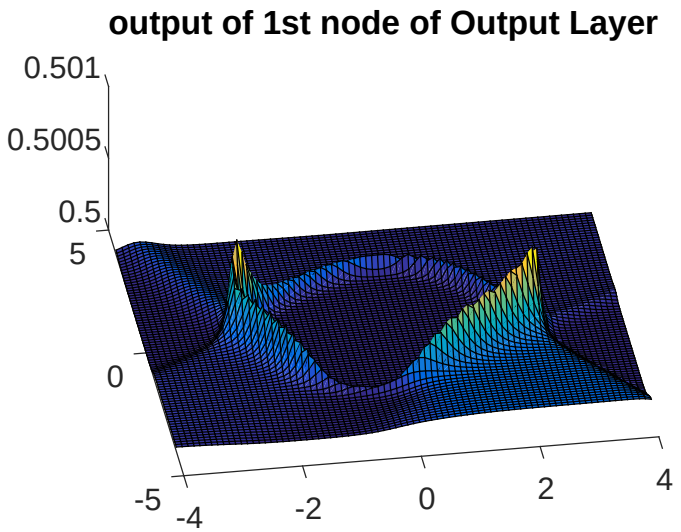
(b) Epoch 2



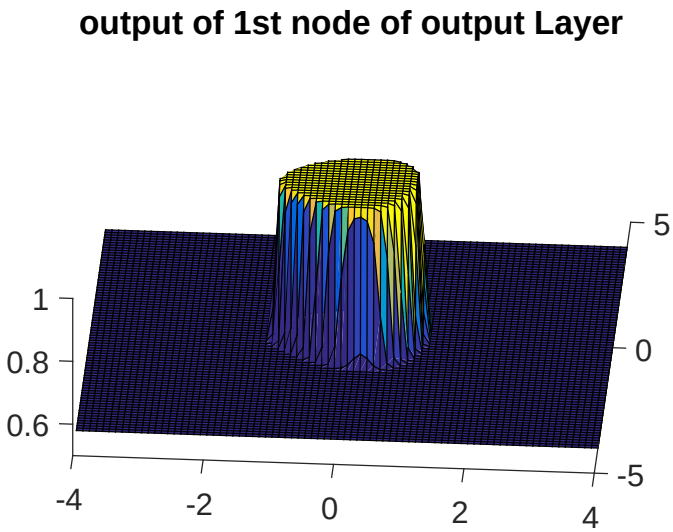
(c) Epoch 10



(d) Epoch 50

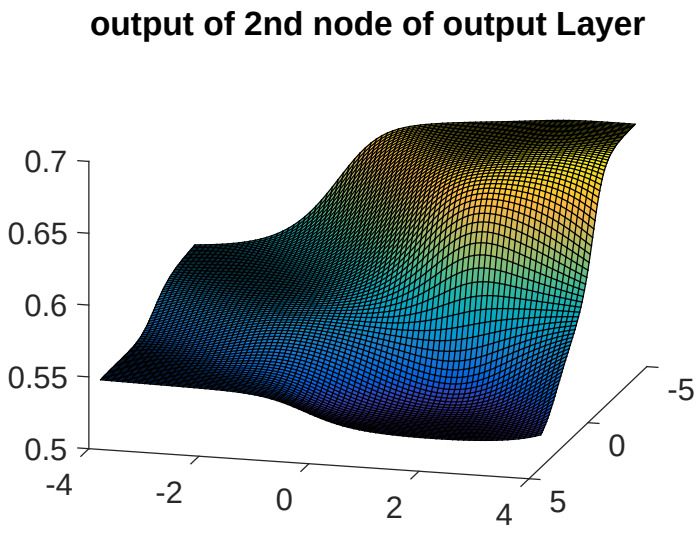


(e) Epoch 100

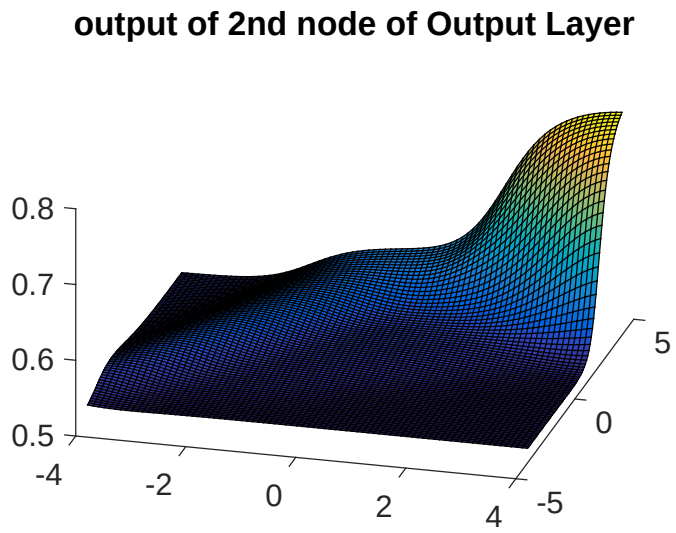


(f) Epoch 154

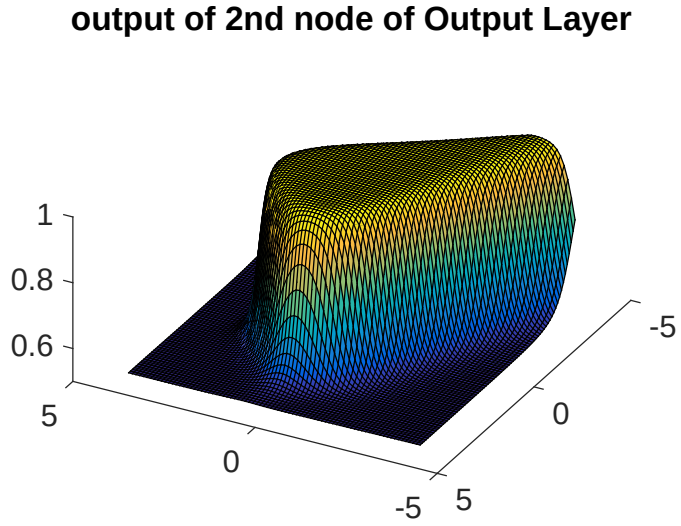
Figure 17



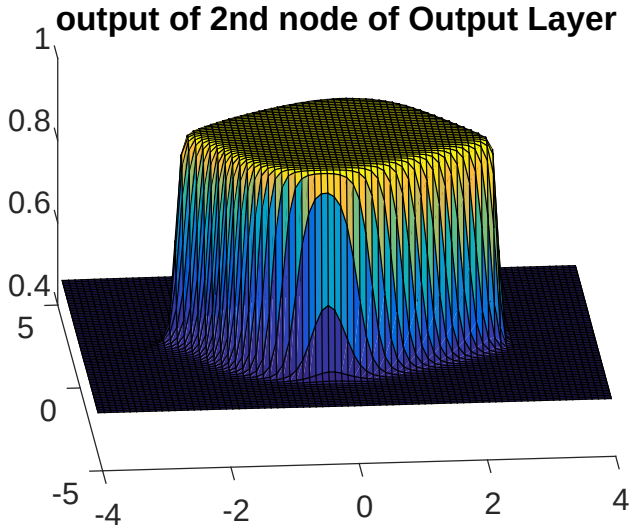
(a) Epoch 1



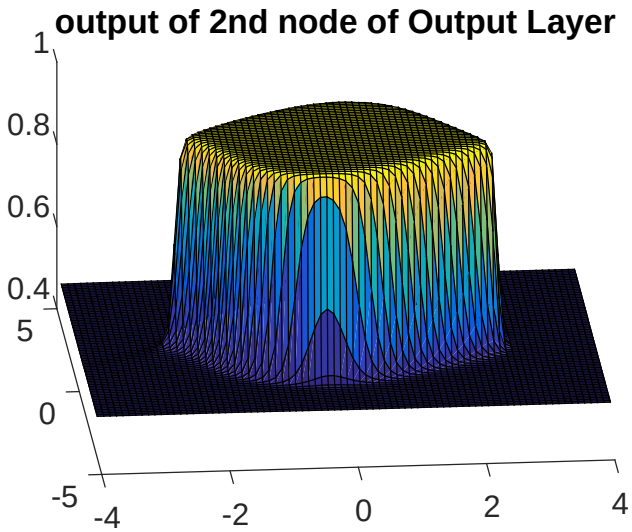
(b) Epoch 2



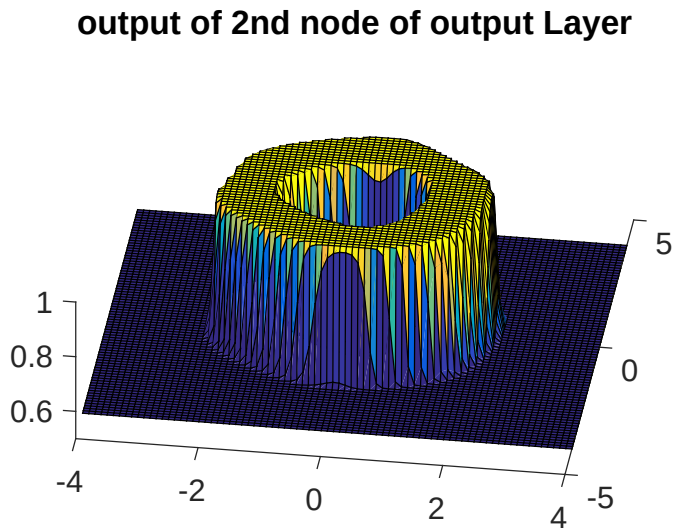
(c) Epoch 10



(d) Epoch 50

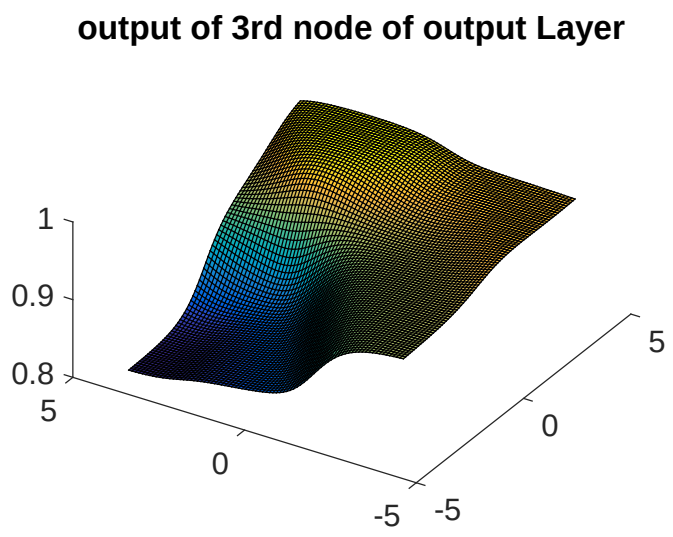


(e) Epoch 100

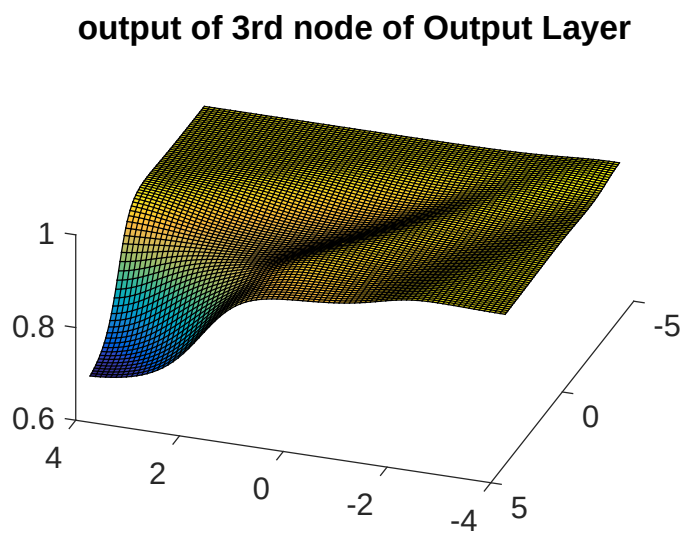


(f) Epoch 154

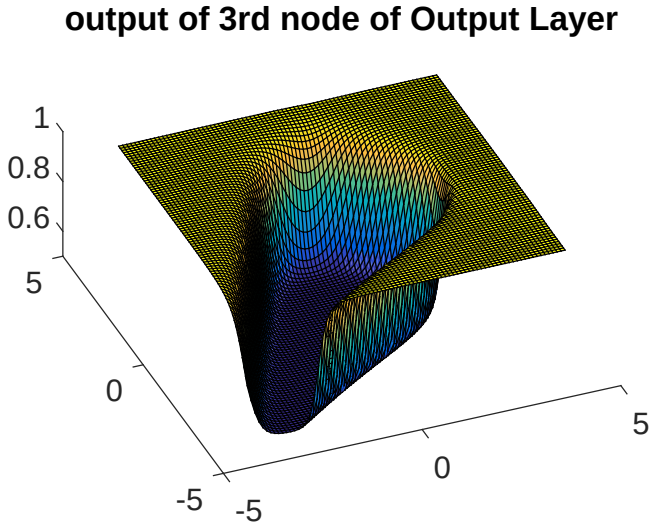
Figure 18



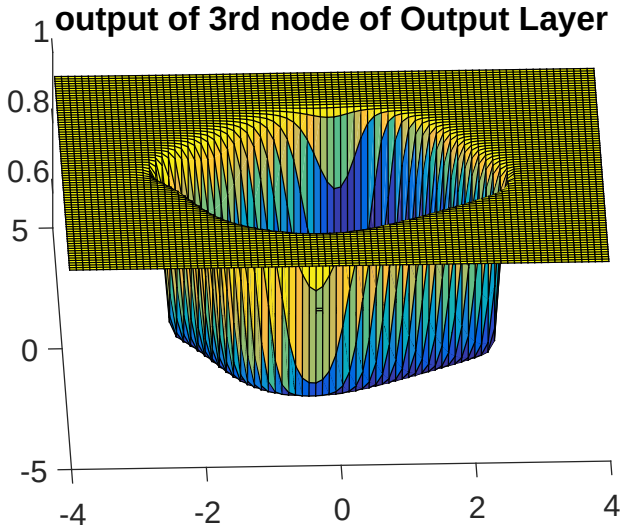
(a) Epoch 1



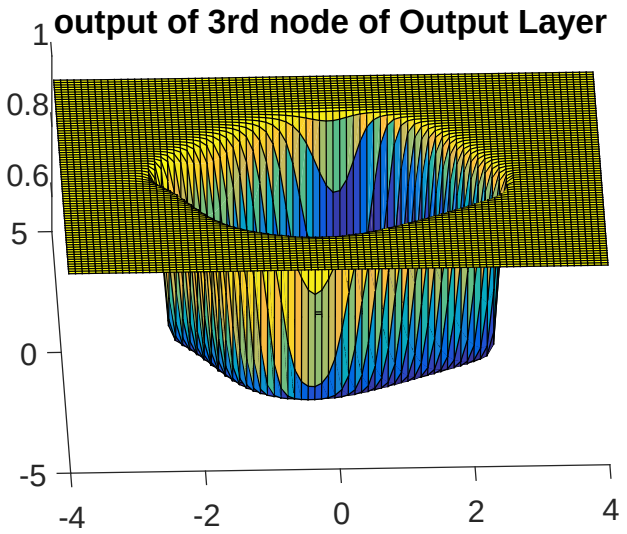
(b) Epoch 2



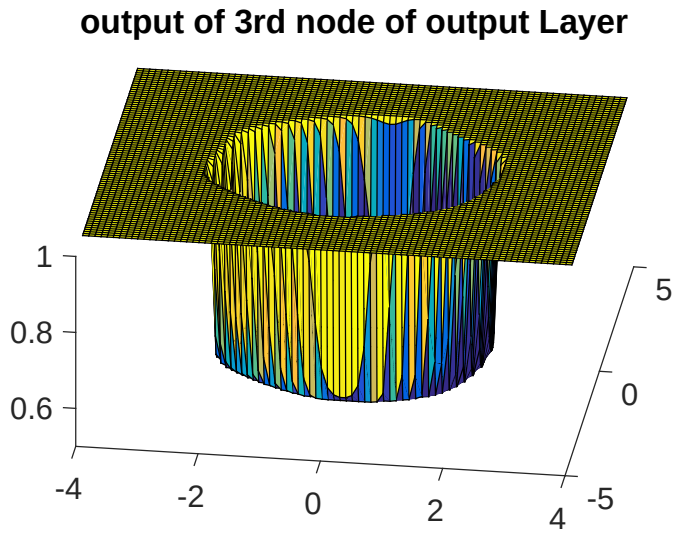
(c) Epoch 10



(d) Epoch 50

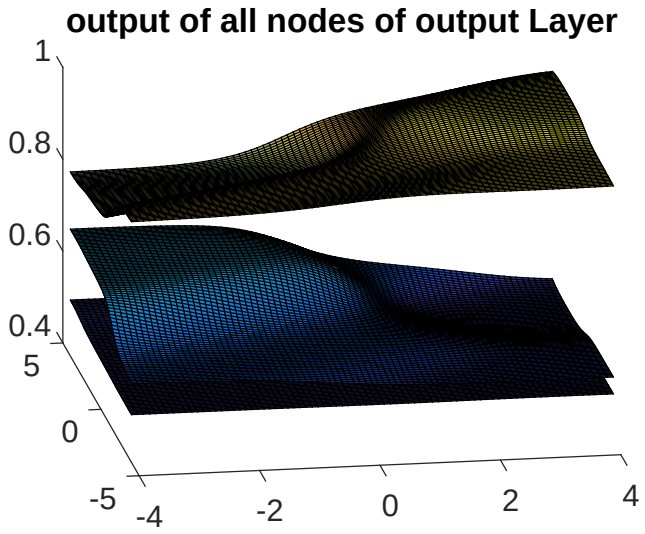


(e) Epoch 100

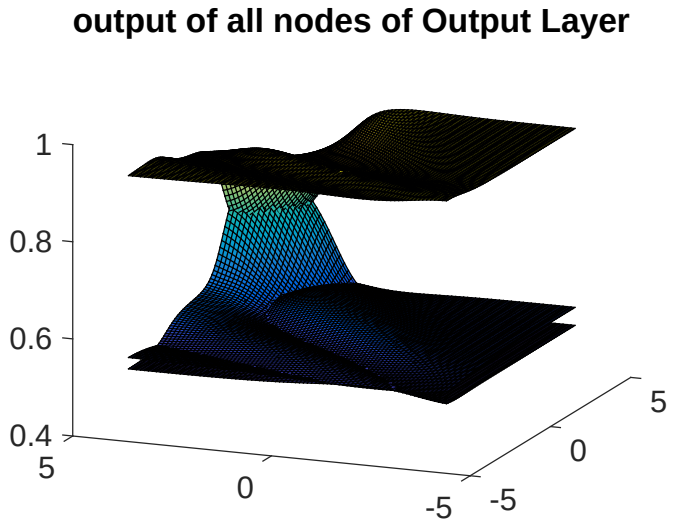


(f) Epoch 154

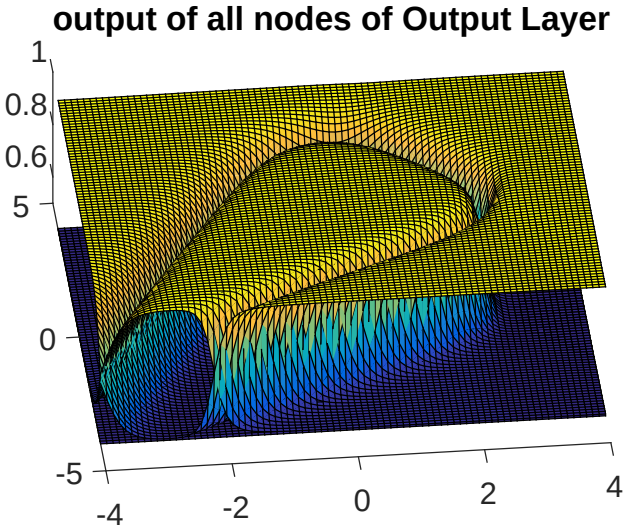
Figure 19



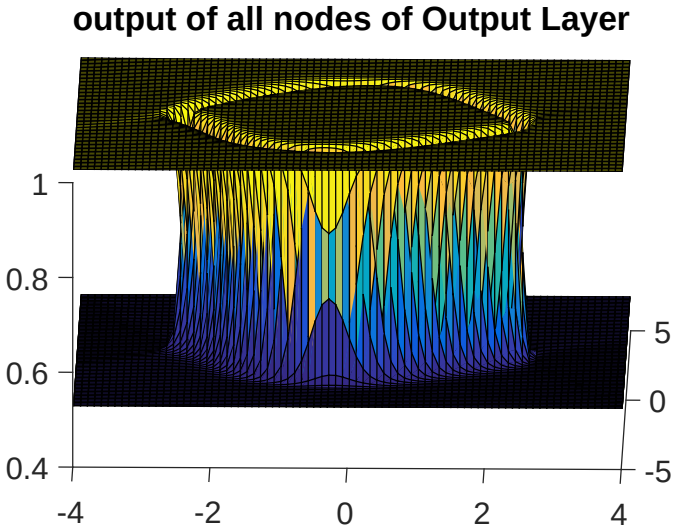
(a) Epoch 1



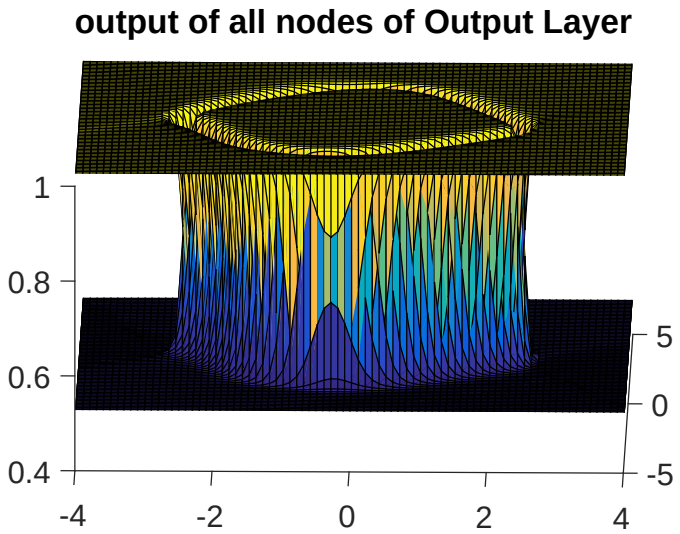
(b) Epoch 2



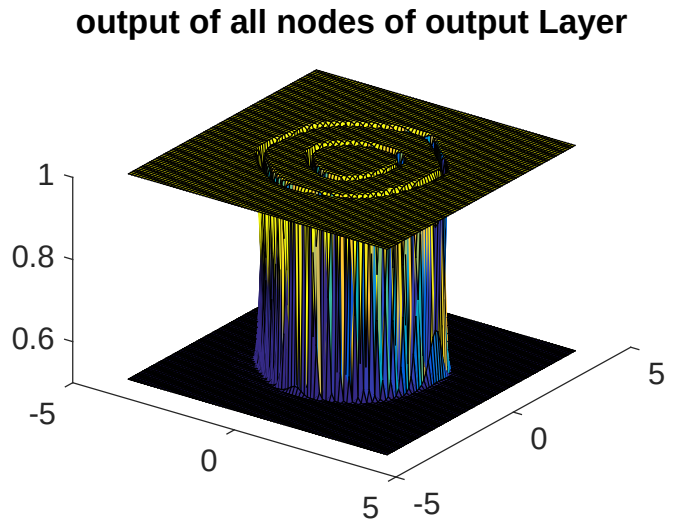
(c) Epoch 10



(d) Epoch 50



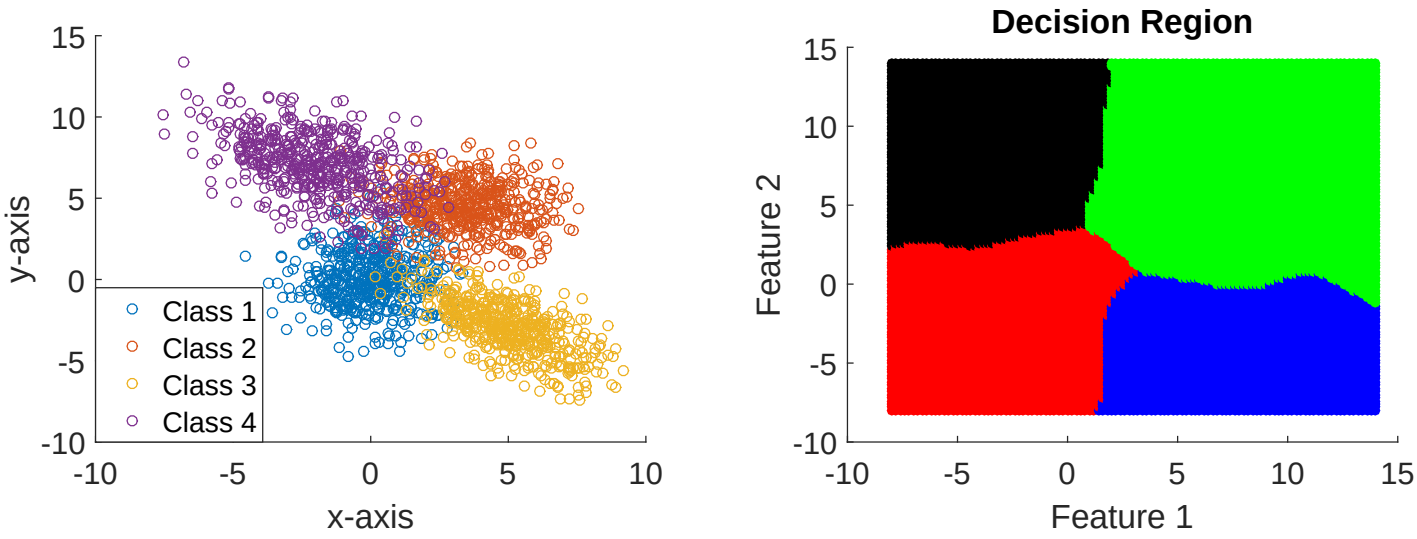
(e) Epoch 100



(f) Epoch 154

Figure 20

1.3.3 Overlapping classes



(a) Scatter Plot

(b) Decision Regions

Figure 21

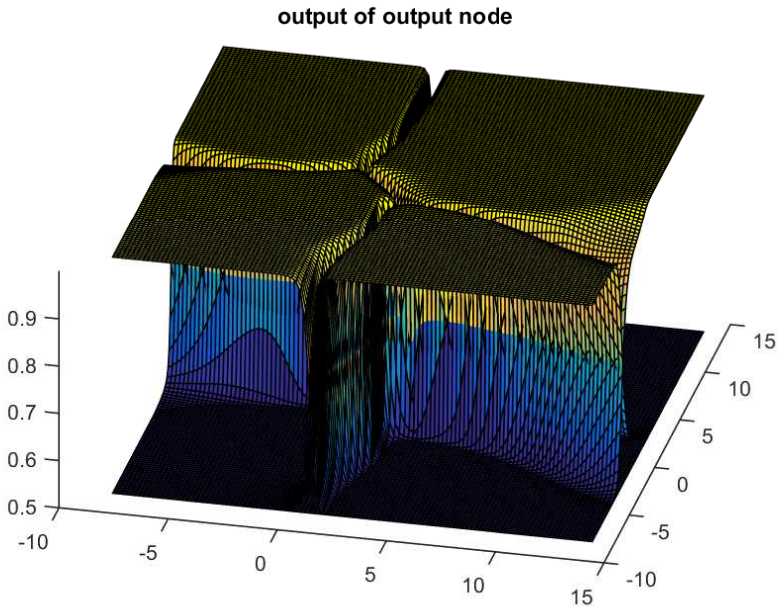


Figure 22: Output of all output nodes

Figure 23: Confusion matrix for 2 hidden layer with 9 nodes each

1.3.4 Image data

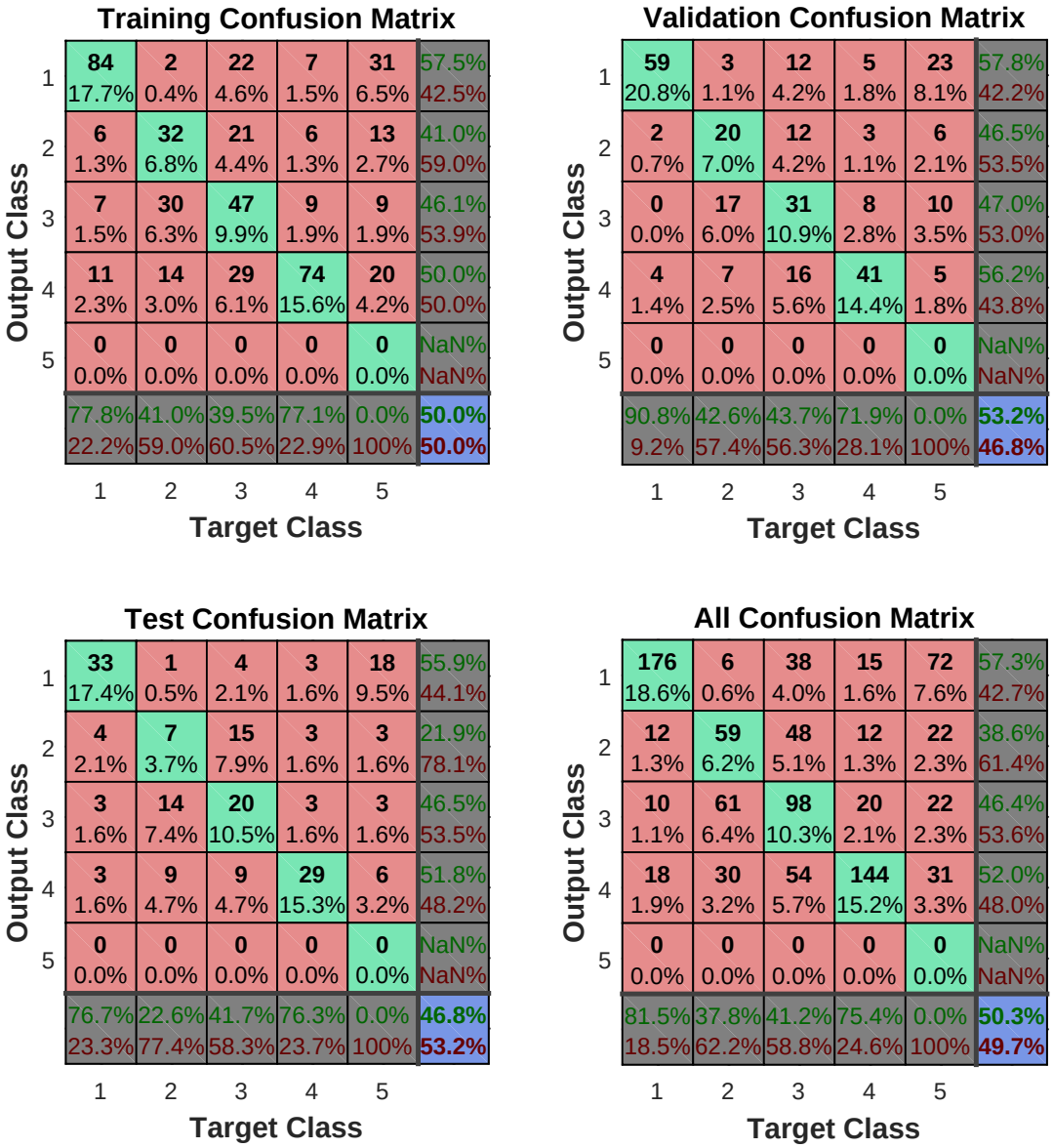


Figure 24: Confusion matrix for 2 hidden layers with 15 nodes each

1.4 Observations and Inferences

- We get 100% accuracy for a single hidden layer with 3 nodes since a single hidden layer acts as a perceptron classifier for linearly separable data.
- For nonlinearly separable data we achieved 100% accuracy with atleast 4 nodes in each hidden layer.
- For overlapping we achieved a maximum of 93%
- For image data due to limited number of examples it always gave negligible accuracy for atleast one class. For two class problems we got an accuracy of around 85% and for 5 class problem we got accuracy of around 52%

2 Regression

2.1 Models

2.1.1 Polynomial Curve Fitting

Polynomial curve fitting is one of the simplest method for curve fitting. It is of the form

$$y(x, \bar{w}) = \omega_0 + \omega_1 x + ... + \omega_M x^M = \sum_{j=0}^{M} \omega_j x^M \tag{1}$$

Where $\bar{w} = \begin{bmatrix} \omega_0 & \omega_1 & \cdots & \omega_M \end{bmatrix}^T$ is a vector of dimension M+1. Even though the function is a non linear function of x, it is linear function of weights ω.

We use the set of training samples $D = \left\{x_n, t_n\right\}_{n=1}^{N}$ to find the weights. This is done by minimizing the error between the predicted values $y(x_n, \bar{w})$ and actual measurement t_n in the training data set. We chose an error function which is a sum of squared errors between predicted values and target values. So our objective is to minimize the function

$$\xi(\bar{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \bar{w}) - t_n)^2 \tag{2}$$

We are choosing an squared error function because we are not bothered about the sign of error and factor $\frac{1}{2}$ is introduced for convenience when we differentiate this function w.r.t $\bar{w}$. We could have expressed the error function in absolute error form but to make mathematical analysis easy we are using squared error form. Error function is a quadratic function in terms of weights but when we differentiate it is linear in weights.

$$\frac{\partial \xi(\bar{w})}{\partial w_j} = 0 \quad \forall j = 0, 1, 2, ..., M \tag{3}$$

$$\frac{\partial \xi(\bar{w})}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \left[\sum_{n=1}^{N} \left(\sum_{i=0}^{M} \omega_i x_n^i - t_n \right)^2 \right] = 0 \tag{4}$$

Neglecting the constant terms and equating it to zero we get

$$\Rightarrow \sum_{n=1}^{N} \left(\sum_{i=0}^{M} \omega_i x_n^i - t_n \right) x_n^j = 0. \tag{5}$$

In the next step we change the order of summation

$$\Rightarrow \sum_{i=0}^{M} \left(\sum_{n=1}^{N} x_n^{i+j} \right) \omega_i = \sum_{n=1}^{N} t_n x_n^j \tag{6}$$

Where $\sum_{n=1}^{N} x_n^{i+j}$ is a term dependent on i and j for a particular data set and we can denote it as a_{ij} and $\sum_{n=1}^{N} t_n x_n^j$ is dependent on j and denote it as c_j.

$$\Rightarrow \sum_{i=0}^{M} a_{ij}\, \omega_i = c_j \quad for\ j = 0, 1, 2, ..., M. \tag{7}$$

Solving these M+1 equations we can get the values of weight coefficients.

Let $\bar{w}^*$ be the optimal weights obtained by solving these equations. This solution is dependent on the training data set. When we change the training data set, the weights we obtain is also different.

2.1.2 Regularization

One of the methods to control over-fitting is by regularization. This is done by adding an extra term to the error function which prevents the weight factor from reaching high values and avoids the oscillation of those weights. Regularization can be seen as a technique to improve the generalization of a learned model and to control the smoothness of the function.

In our case modified error function becomes

$$\xi(\bar{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n, \bar{w}) - t_n)^2 + \frac{\lambda}{2}\left\|w^2\right\| \tag{8}$$

Where $\|w^2\| = w^T w = \omega_0^2 + \omega_1^2 + ... + \omega_M^2$

When we differentiate this error function with respect to weights we get a new set of simultaneous equations which we will use to solve the weight values.

$$\sum_{i=0}^{M} a_{ij}\omega_j + \lambda\omega_j = c_j \;\; for \;\; j = 0, 1, ...M \tag{9}$$

This type of regression is also called ridge regression.

2.1.3 Linear model for regression

Given training data with the target output, predict the result for a new input. This is done by constructing an appropriate function which represent the distribution.

Gaussian basis functions:

Here we use gaussian basis function. $\phi_j(x) = e^{-\frac{(x-\mu_j)^2}{2s^2}}$

These are local; A small change in x only affect nearby basis functions. ϕ_j and s control the location and scale(width).

Least square method:

Earlier we tried polynomial functions to fit the data sets by minimising the sum of squares error function. The error function is given by

$$\xi(\overline{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(\overline{x}, \overline{w}) - t_n)^2 \tag{10}$$

Where

$$y(\overline{x}, \overline{w}) = \overline{w}^t.\phi(\overline{x}) \tag{11}$$

So the error function becomes

$$\xi(\overline{w}) = \frac{1}{2}\sum_{n=1}^{N}(\overline{w}^t.\phi(\overline{x_n}) - t_n)^2 \tag{12}$$

By taking derivative and equating to 0

$$\frac{\partial \xi(\overline{w})}{\partial(\overline{w})} = 0 \tag{13}$$

$$\Rightarrow \sum_{n=1}^{N} (\overline{w}^t.\phi(\overline{x_n}) - t_n)\phi(\overline{x_n})^t = 0 \tag{14}$$

$$\Rightarrow \sum_{n=1}^{N} (\overline{w}^t.\phi(\overline{x_n}))\phi(\overline{x_n})^t = \sum_{n=1}^{N} t_n\phi(\overline{x_n})^t \tag{15}$$

This is a set of M linear equations and represented by a design matrix: $\Phi_{N\times M} = \begin{bmatrix} \phi^t(\overline{x_1}) \\ \phi^t(\overline{x_2}) \\ \vdots \\ \phi^t(\overline{x_n}) \end{bmatrix}$ so,

$$(\Phi^t\Phi)\overline{w} = \Phi^t\bar{t} \tag{16}$$

$$w^* = (\Phi^t\Phi)^{-1}\Phi^t\bar{t} \tag{17}$$

where $\bar{t}$ is a column vector containing the target values $\{t_n\}_{n=1}^N$.
The term

$$\Phi^\dagger \equiv \left(\Phi^T\Phi\right)^{-1}\Phi^T \tag{18}$$

is known as the *Moore-Penrose-inverse* of the matrix Φ. It is used for giving a notion of inverses to rectangular matrices. If Φ is square $\Phi^\dagger \equiv \Phi^{-1}$.

Regularized least square method:

The error function is

$$\xi(\overline{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(\overline{x},\overline{w}) - t_n)^2 + \frac{\lambda}{2}\|w\|^2 \tag{19}$$

On solving for w^* using the same method we get

$$w^* = (\Phi^t\Phi + \lambda I)^{-1}\Phi^t\bar{t} \tag{20}$$

Calculate the root-mean-square error ($\xi_r ms$) using the following equations.
$\xi_r ms = 2\xi(w^*)/N$, where N is number of samples in the dataset.
Model complexity is choosen based on the error($\xi_r ms$) on the data with good regularization parameters

2.1.4 Multilayer Feedforward Neural Network

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 tried error minimisation through gradient descent and Levenberg-Marguardt backpropogation method.

2.1.5 Radial Basis Function Neural Network (RBFNN)

This is a case of linear model for regression where the basis function is a radial basis function. In Radial basis Function networks the model output is given

$$y(x, w^*) = \sum_{1}^{N} w_i \phi(\|x_n - \mu_i\|), where \phi(a) = e^{-a^2/\sigma} \tag{21}$$

is known as the radial basis function. In Generalised RBFNN we are trying to reduce the roughness of the approximated function.

2.2 Datasets

2.2.1 Dataset 1: 1-dimensional (Univariate) input data

The function to be approximated given to us $e^{tanh(2\pi x)}$ where $x \epsilon [0, 1]$. Then zero mean noise was added to the function to produce the target outputs.

2.2.2 Dataset 2: 2-dimensional (Bivariate) input data

Training data of different sizes (20, 100, 1000, 2000) are given along with validation data and test data. The validation and testing data size is chosen according to the size of the training data. The size of validation data is set to 15% of the size of training data and that of testing data is set to 10%

2.3 Experimentations and Plots

2.3.1 Polynomial curve fitting for dataset 1

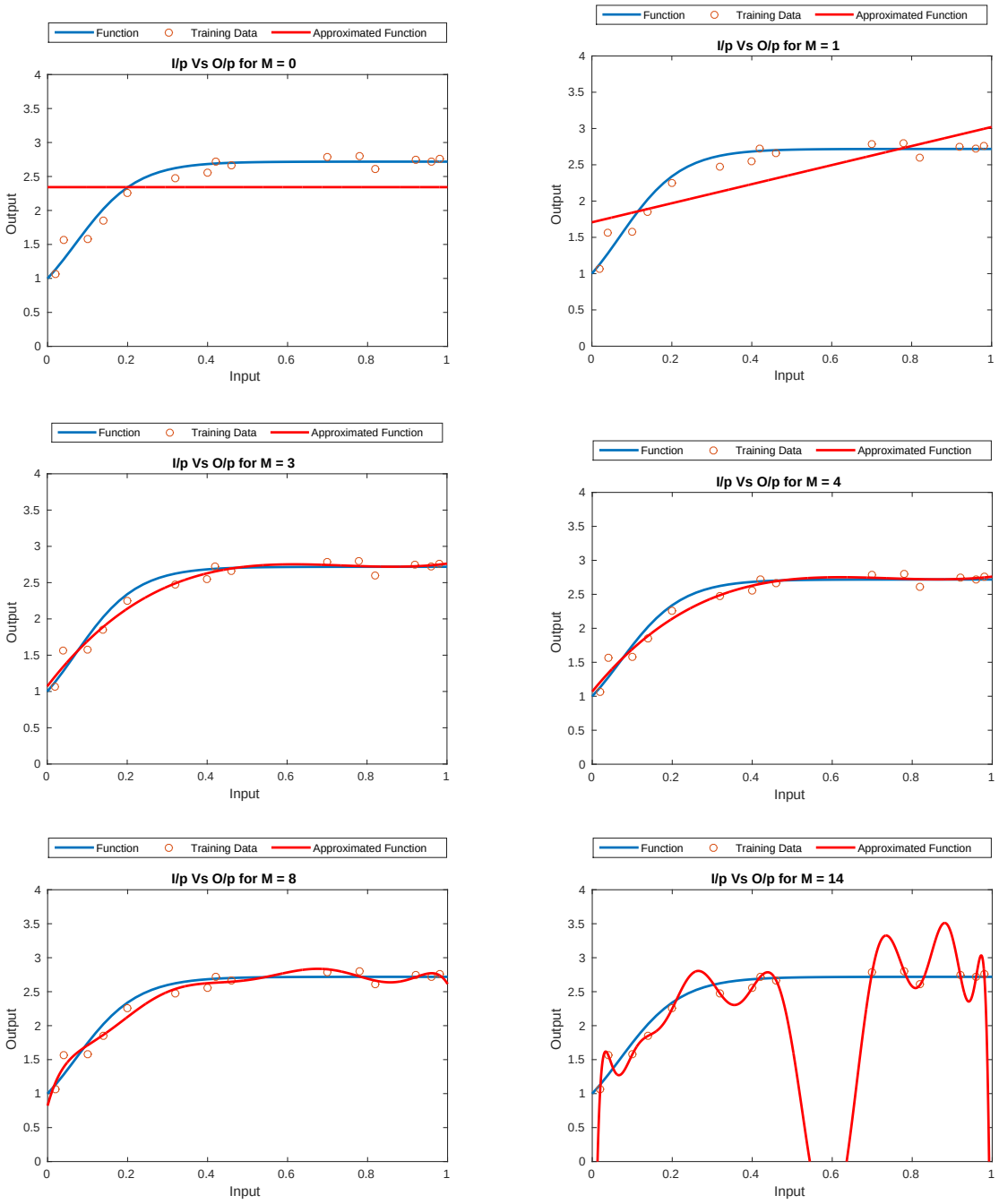


Figure 25: Input vs Output for various model complexities

M = 0 and M = 1 give poor approximation for the underlying function. M = 3,4 gives a good fit to the underlying function. M = 8 gives the mininum root mean squared error for the validation data.

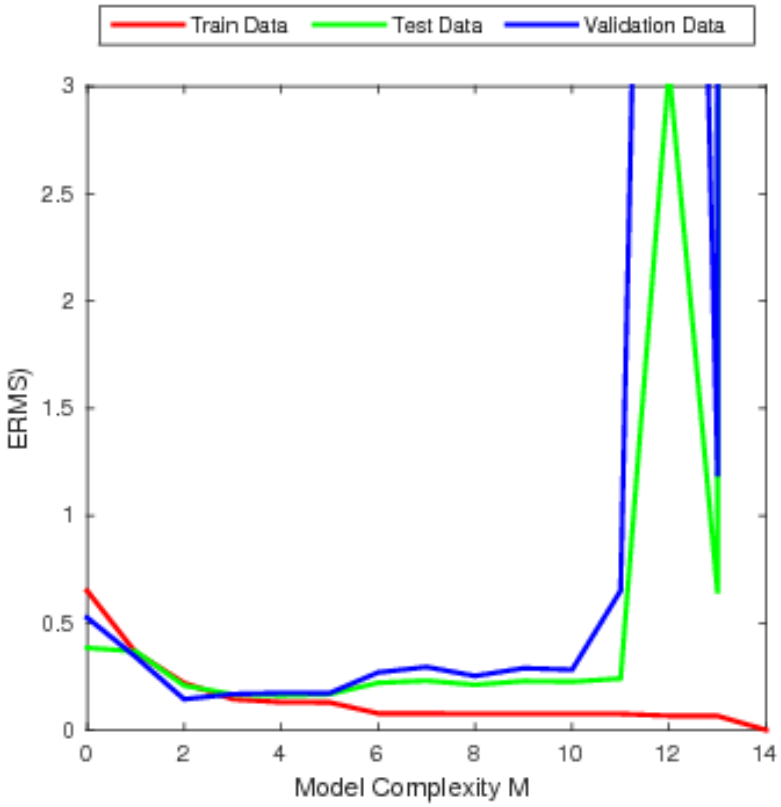


Figure 26: E_{rms} for various model complexity

For M = 14 we can see that the approximated function passes through all the points in the training data set and this is a perfect fit for the training data only. This curve has many peaks and troughs and this phenomenon is called overfitting. From Figure 29 we can observe that as the model complexity increases the training data error decreases but the error on the test and validation data increases.

When we analyse the weight coefficients for M = 14 we see that they are large positive and negative values.

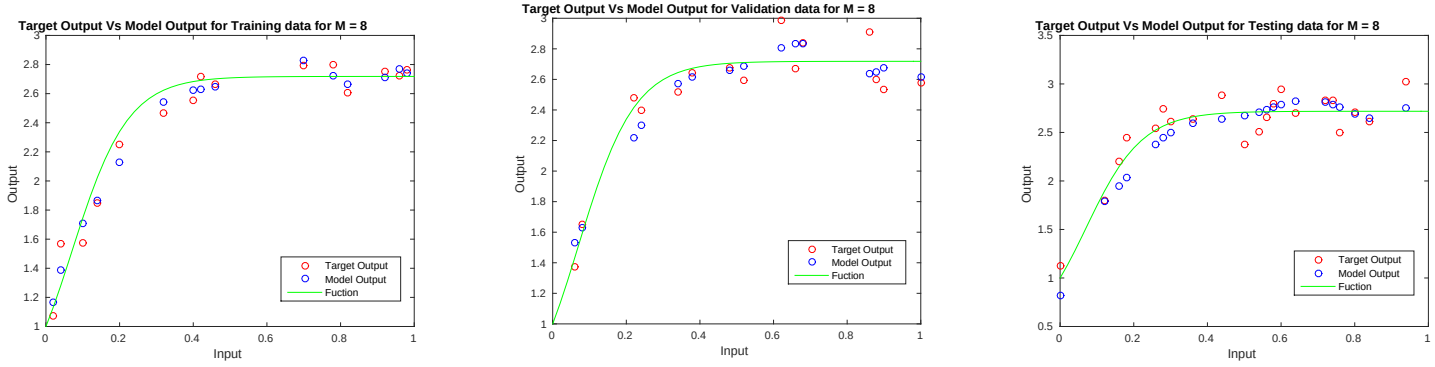


Figure 27: Plots of Target Output and Model Output for M = 8

M = 0	M = 1	M = 3	M=14	M=14 and $ln\lambda$=-15
2.34	1.707	1.071	-5.52	1.06
	1.3135	7.1510	681.28	7.007
		-9.93	-25485.9	-6.2705
		4.4732	5711935.45	-11.1643
			4945927.24	13.54
			-42474057.594	17.0571
			-213207085.45	-3.79
			-743776893.3	-18.54
			-1832529341.7	-16.9
			-32040020585.66	-4.38
			-3945779784.709	10.07
			3342687012.19	18.46
			-18527915555.45	16.46
			604642873.48	2.569
			-880515111.992	-22.532

Table 1: Weight coefficients for polynomials of different complexities

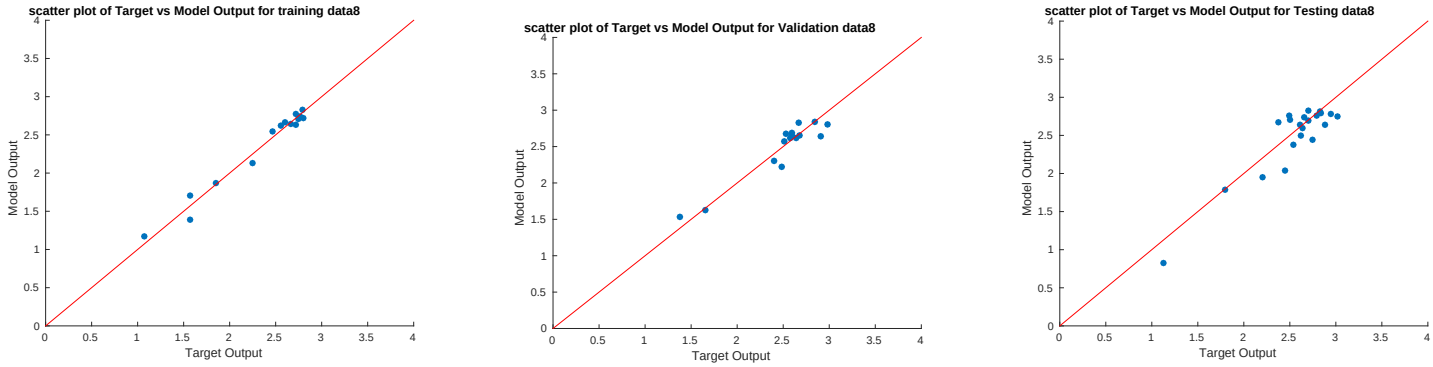


Figure 28: Scatter plots for M = 8

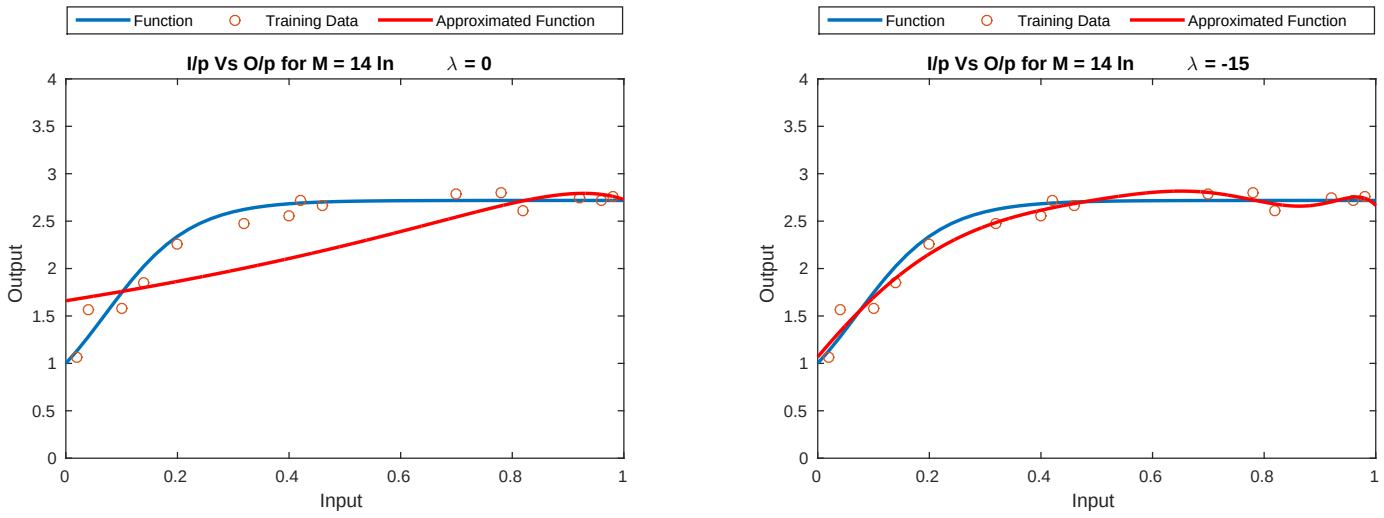


Figure 29: Plots of Inputs Vs Outputs for $ln\lambda = 0$ and $ln\lambda$ = -15

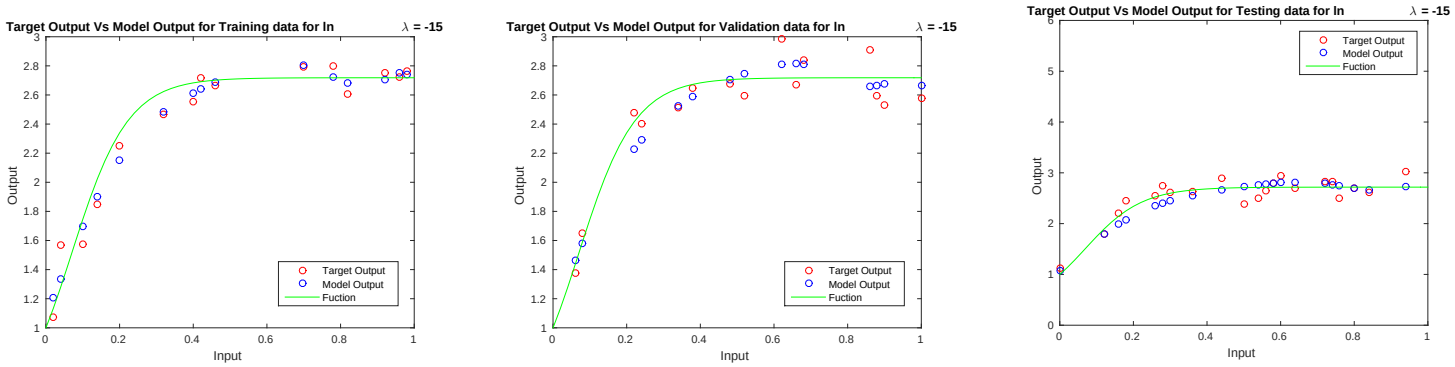


Figure 30: Plots of Target Output and Model Output for lnλ = -15

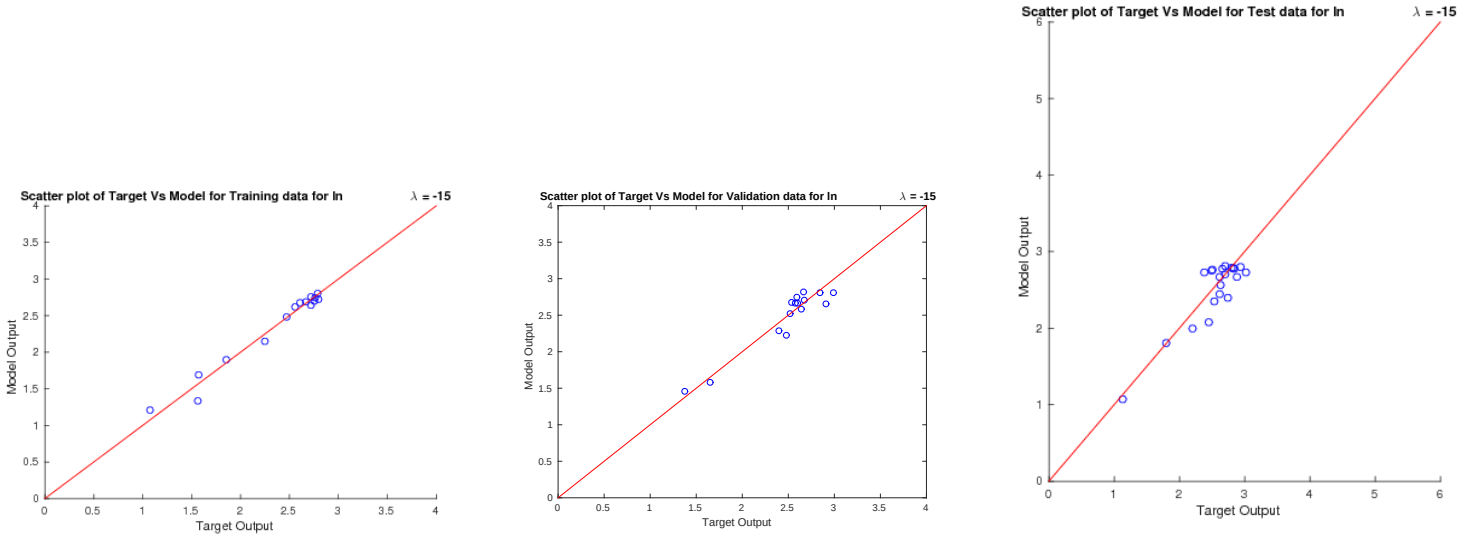


Figure 31: Scatter plots for M = 14 and lnλ = -15

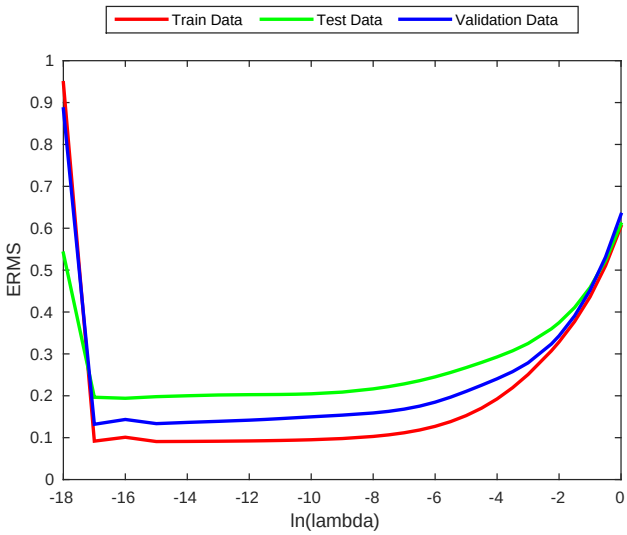


Figure 32: E_{rms} for various regularization parameters $ln\lambda$

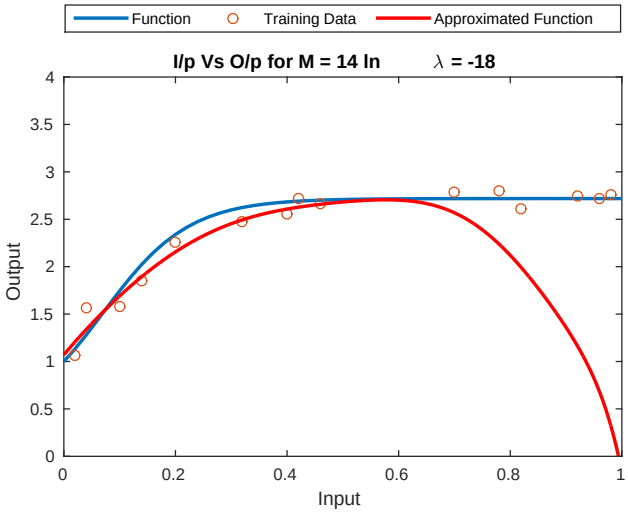


Figure 33: Plot of Input Versus Output for M = 14, $ln\lambda$ = -18

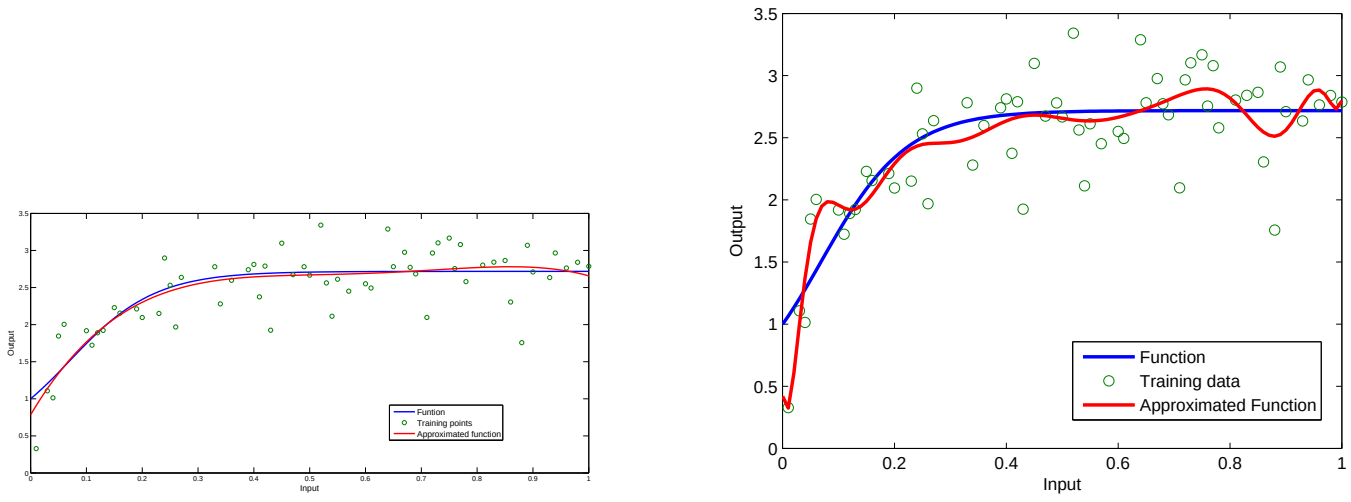
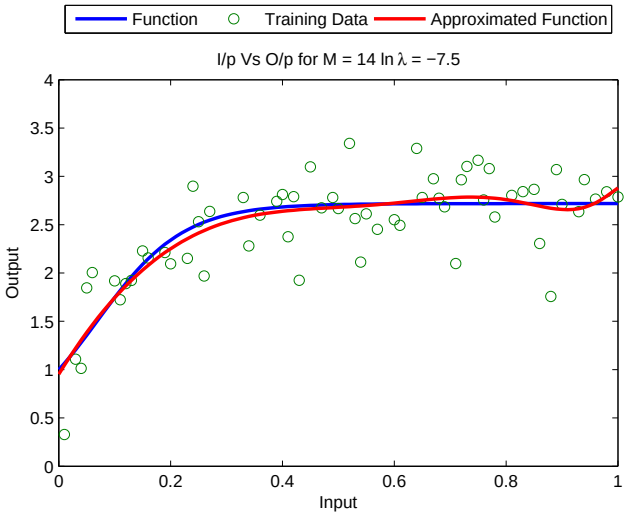
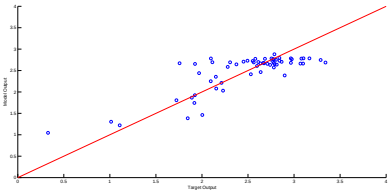
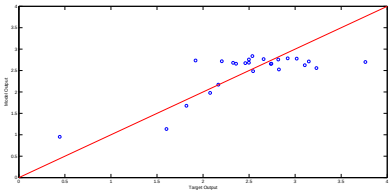


Figure 34: Input vs Output for 60 train samples for M = 4 and M = 14

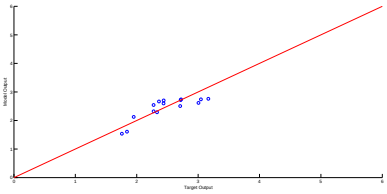




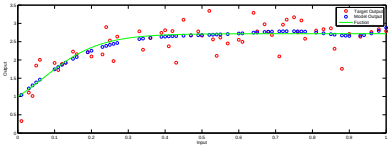
(a) Scatter plot for training data



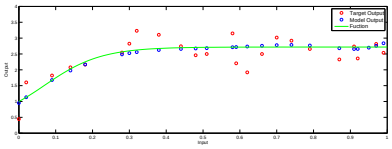
(b) Scatter plot for validation data



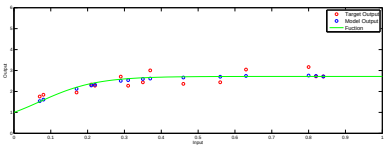
(c) Scatter plot for test data



(d) Target versus Model for training data



(e) Target versus Model for validation data



(f) Target versus Model for test data

Figure 35: Plots for 60 training samples for M = 14 and lnλ = -15

2.3.2 Regression using Gaussian basis functions for dataset 2

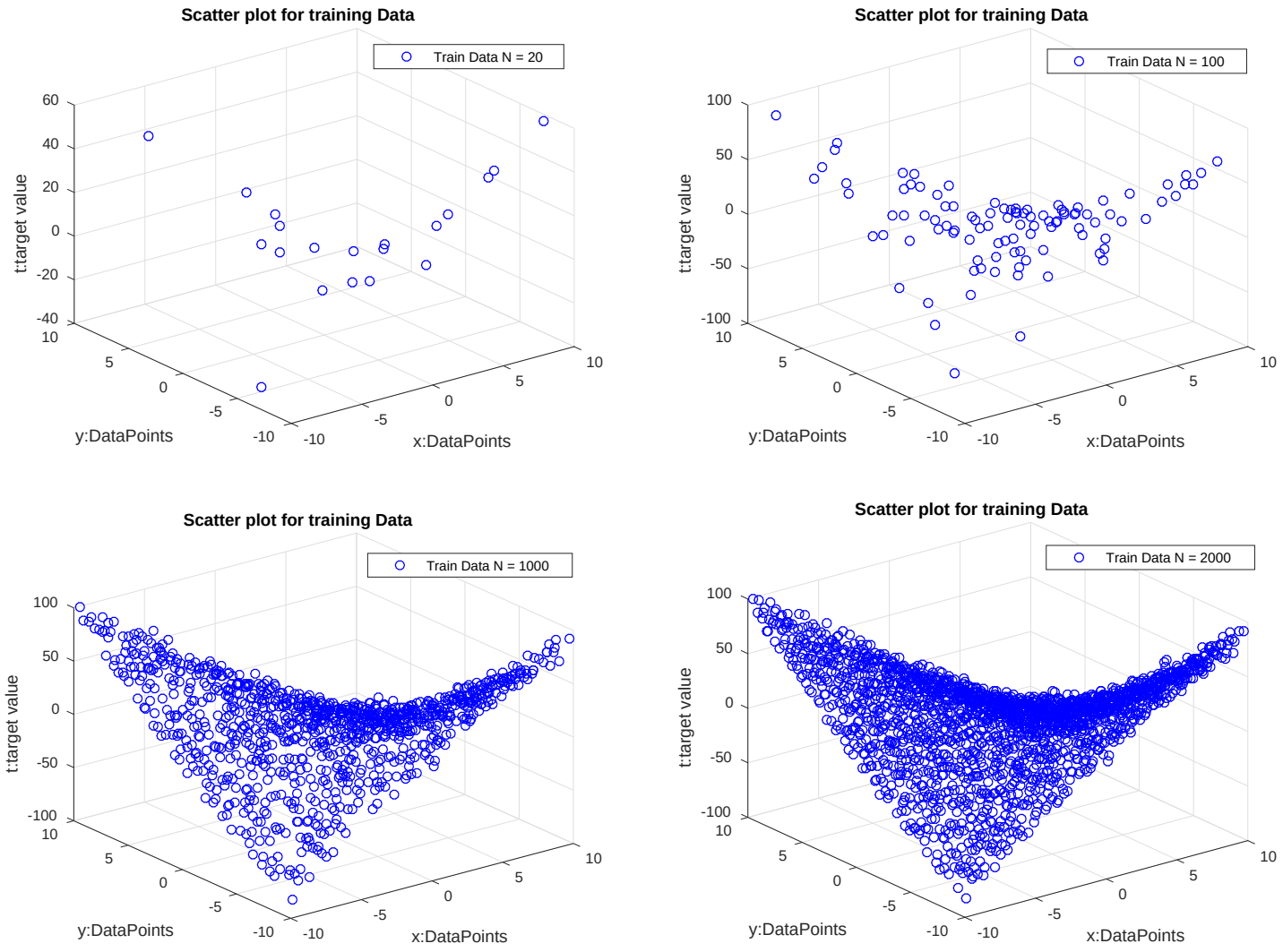


Figure 36: Scatter Plot for Training Data

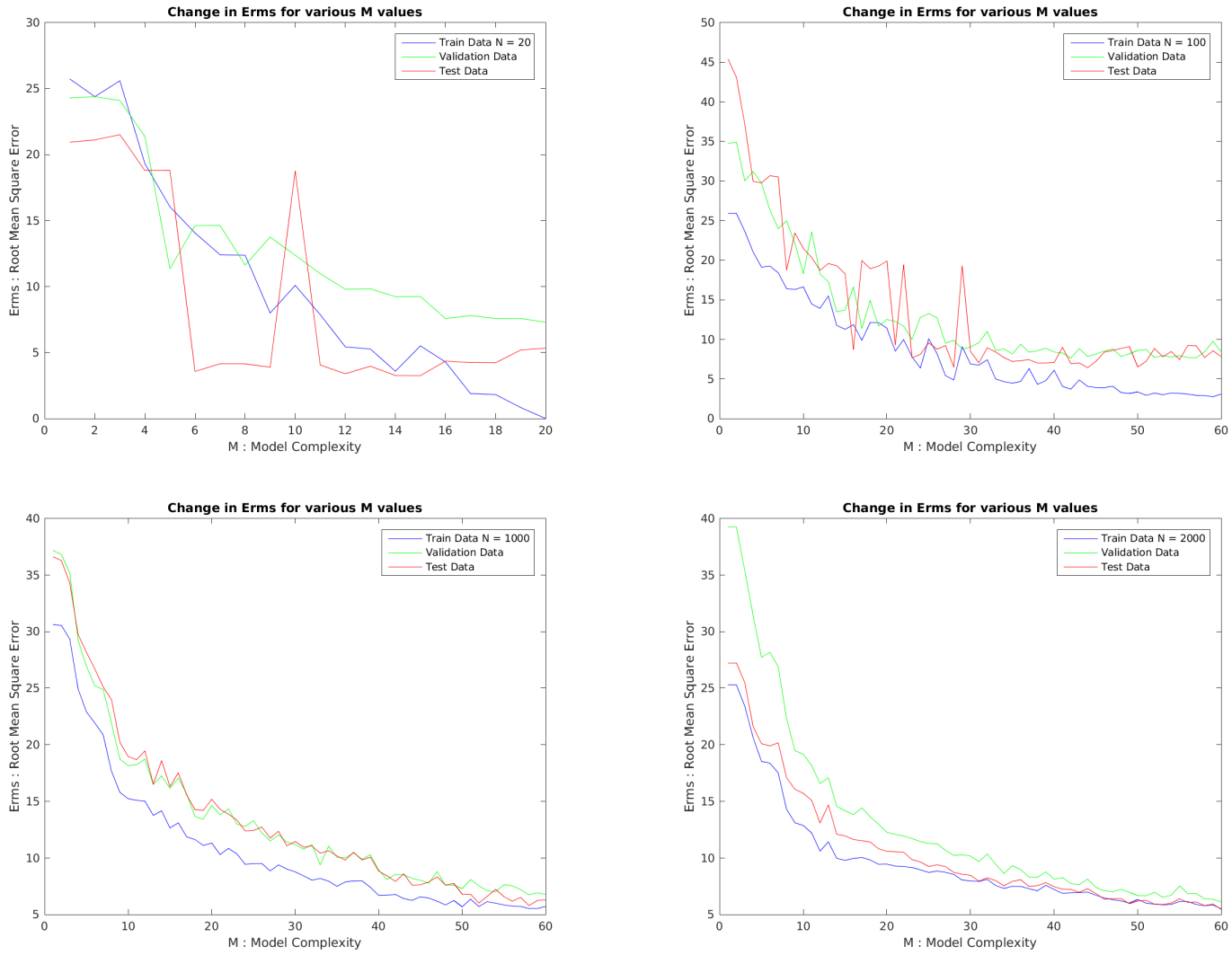


Figure 37: E_{rms} for various model complexity

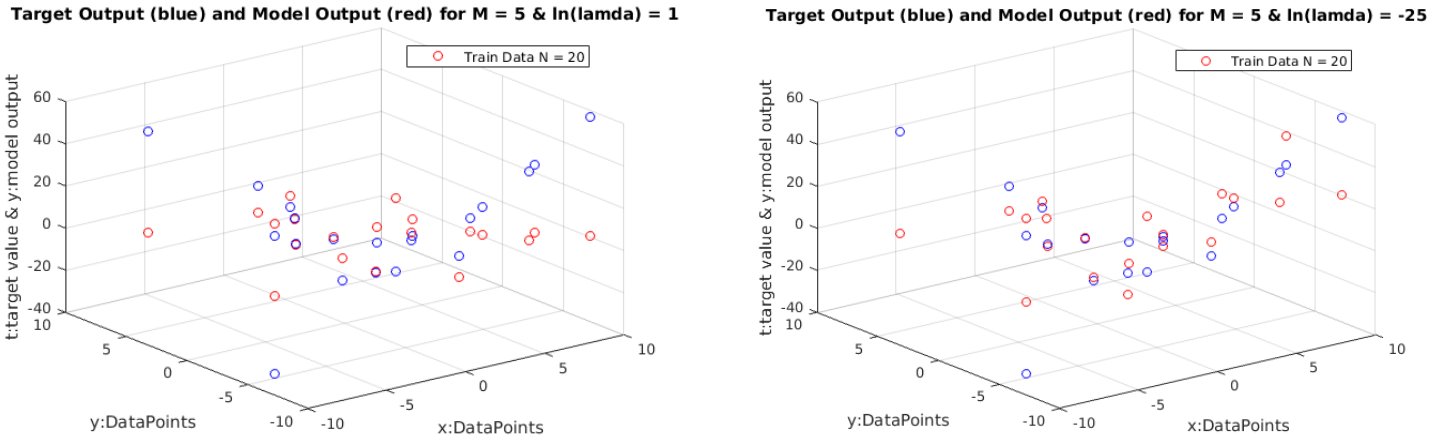
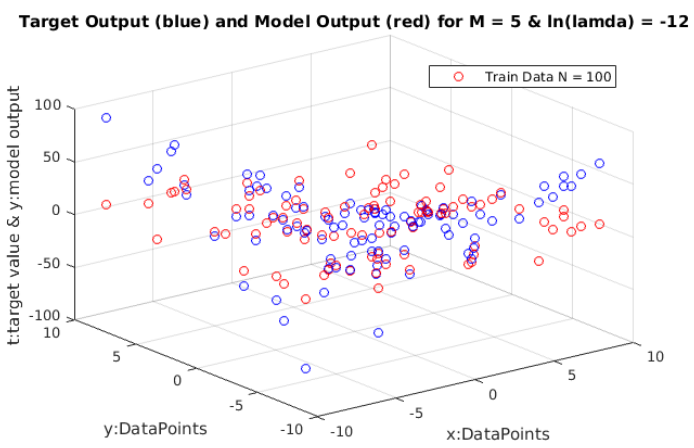


Figure 38: Target vs Model plots (N=20)



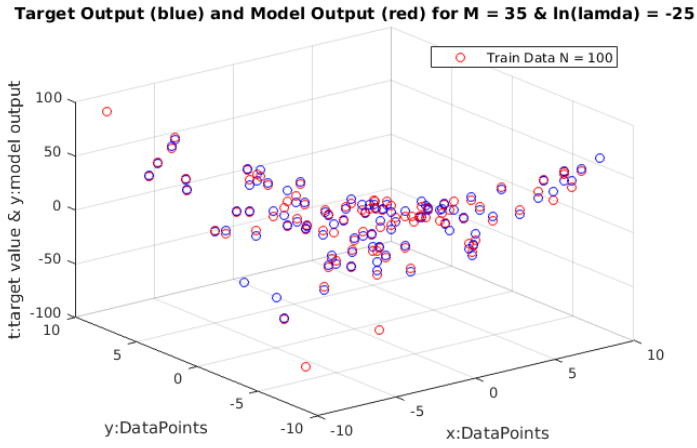
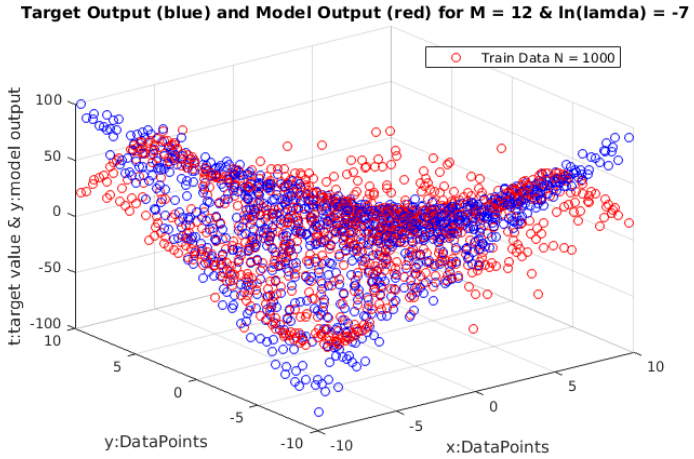


Figure 39: Target vs Model plots (N=100)



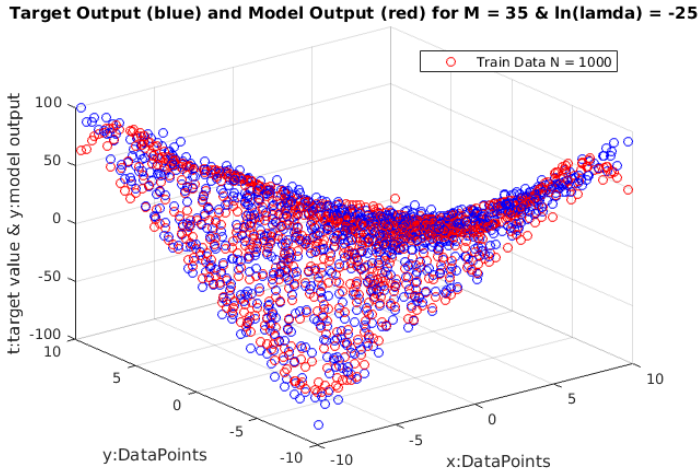
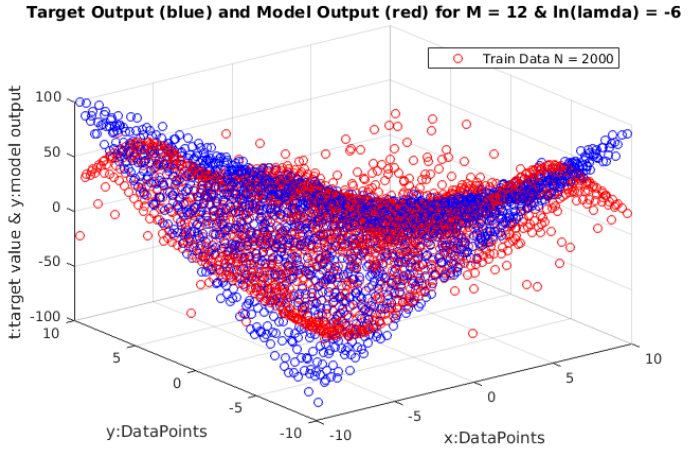


Figure 40: Target vs Model plots (N=1000)



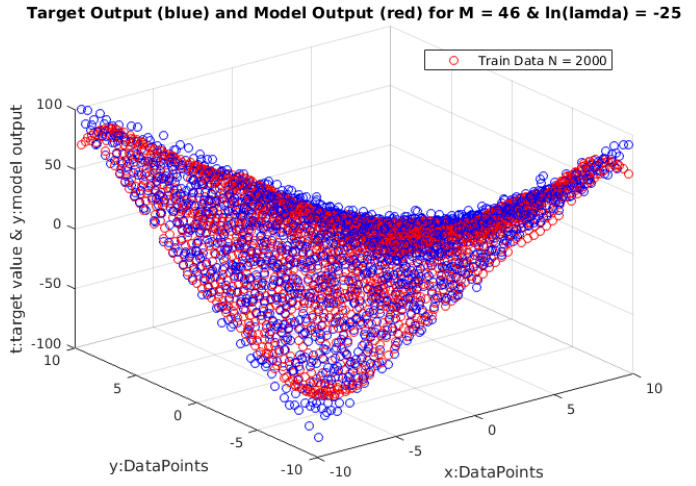


Figure 41: Target vs Model plots (N=2000)

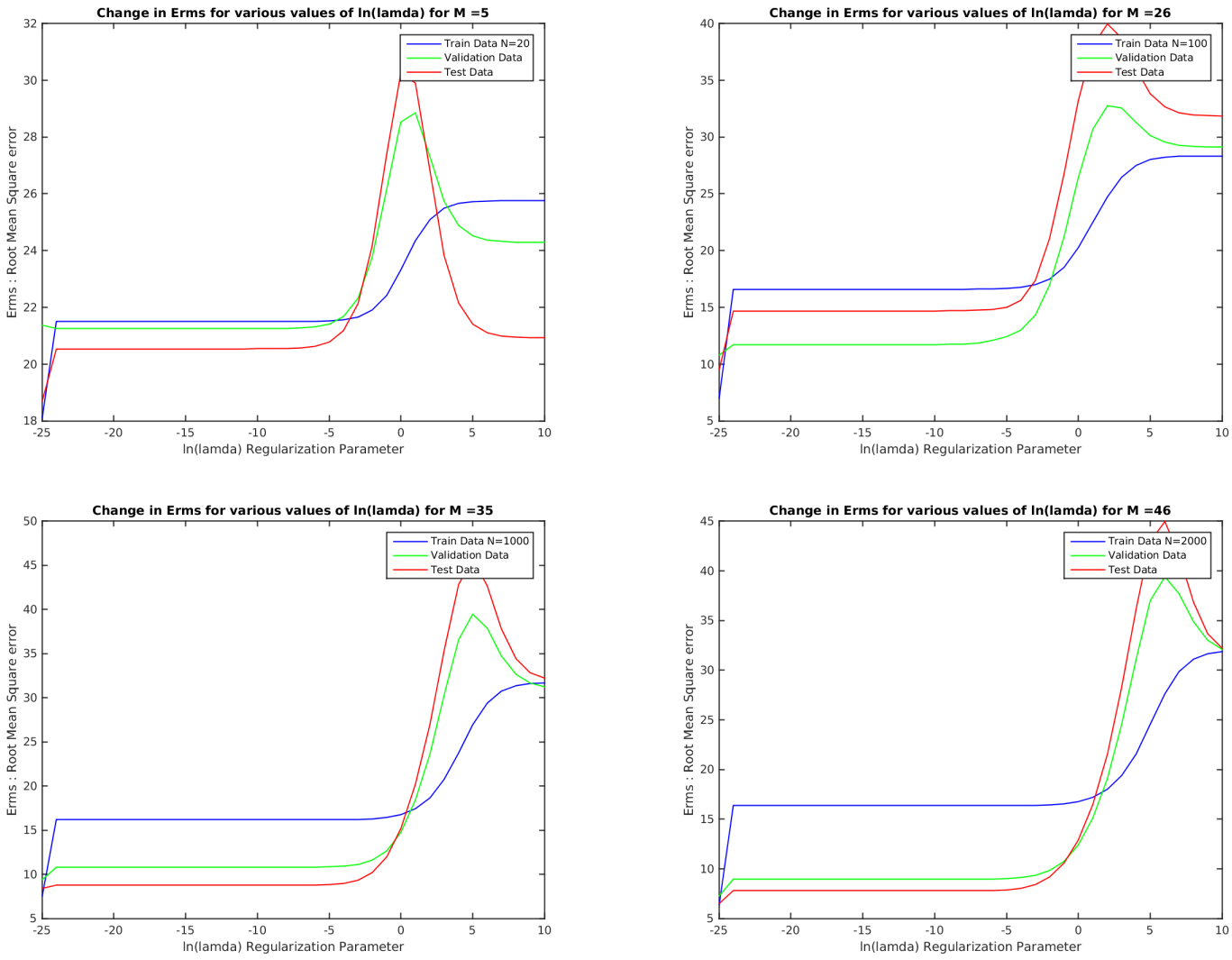
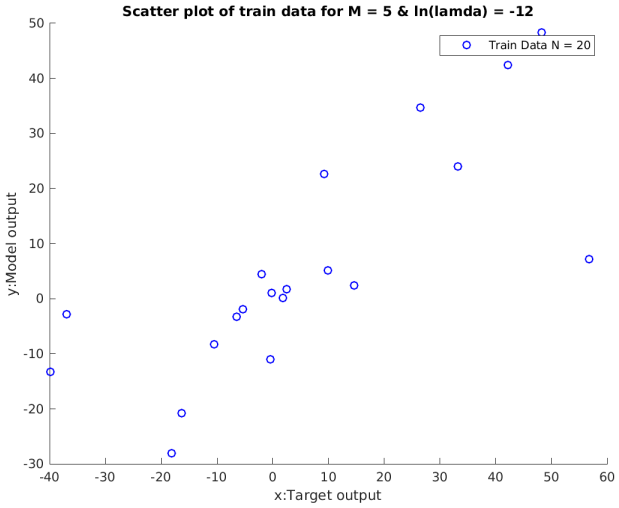
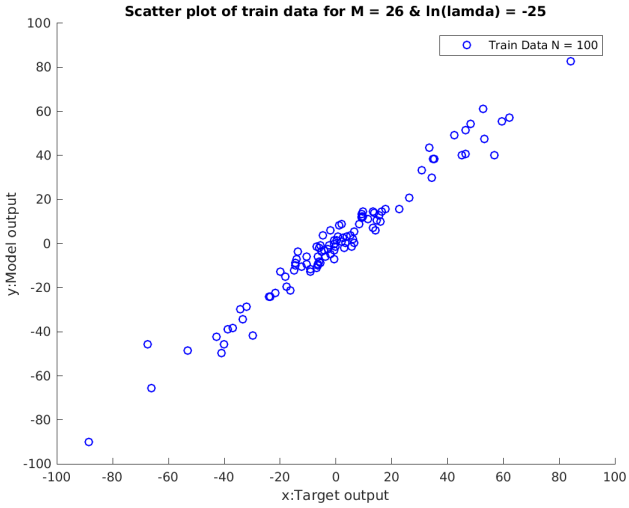


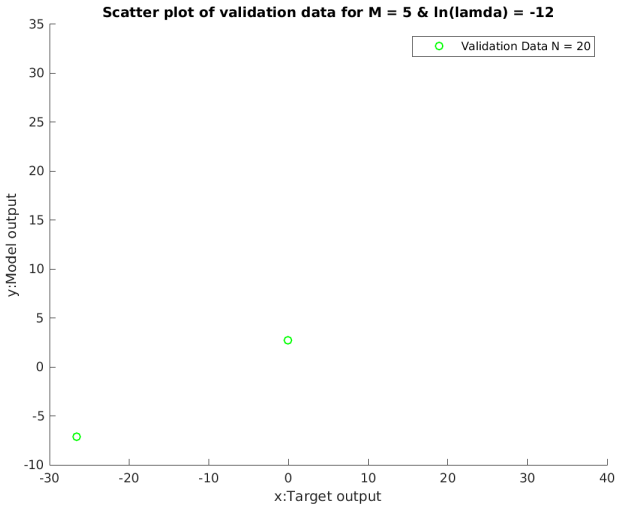
Figure 42: E_{rms} for various λ and model complexity

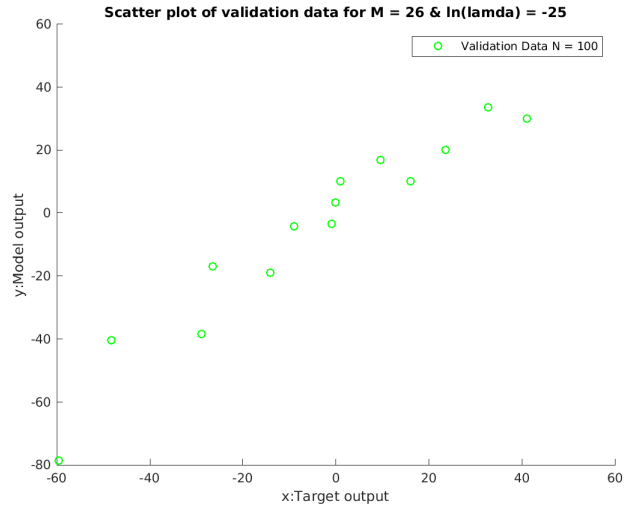


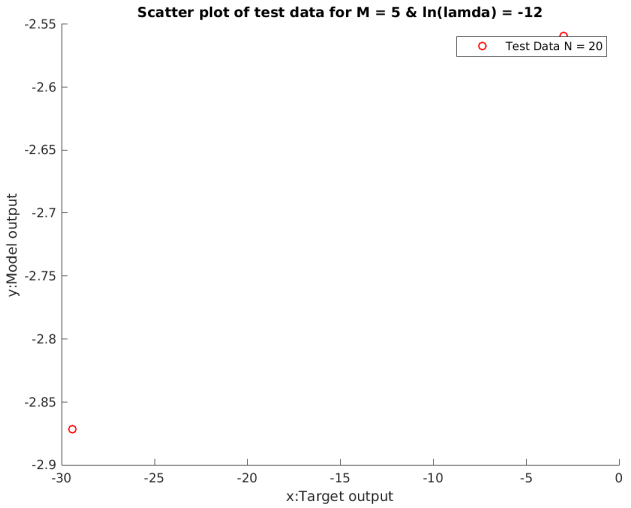
(a) test



(b) test







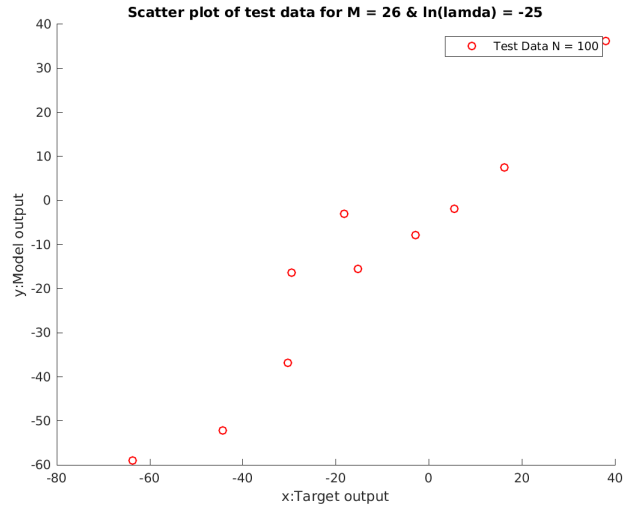
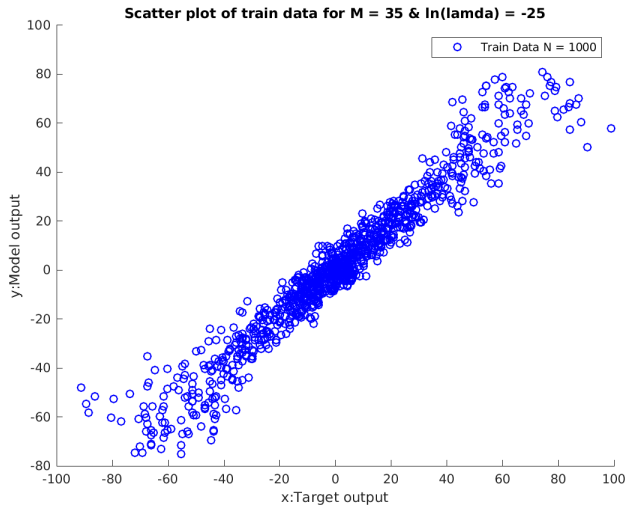
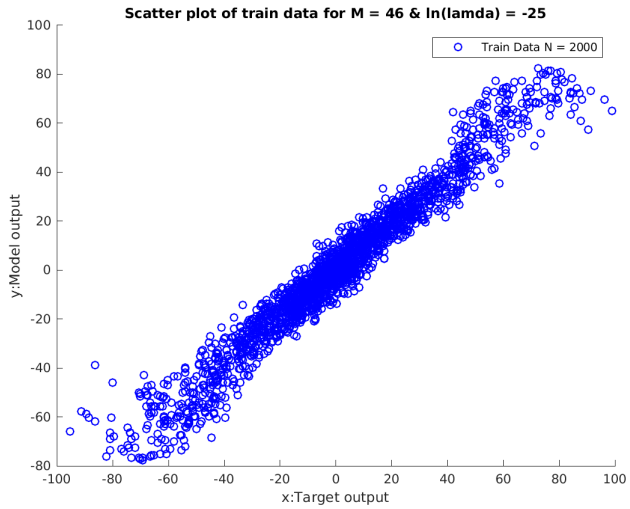
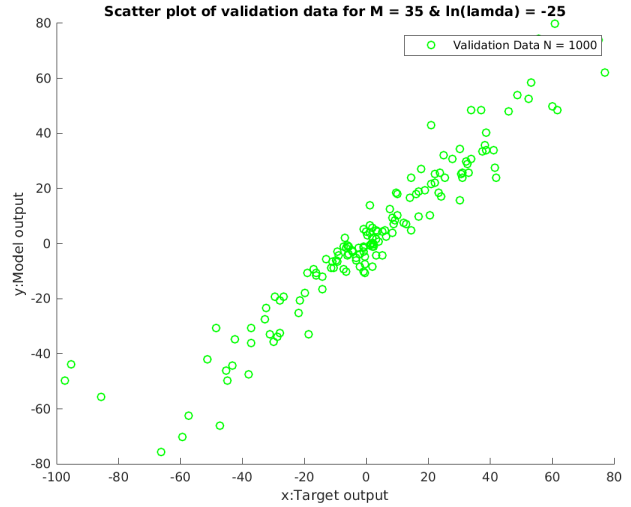
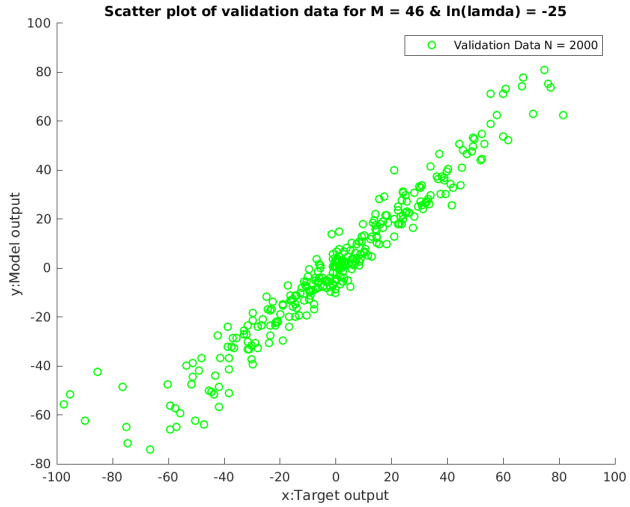


Figure 43: Scatter plots for train, validation and test data for N=20, 100









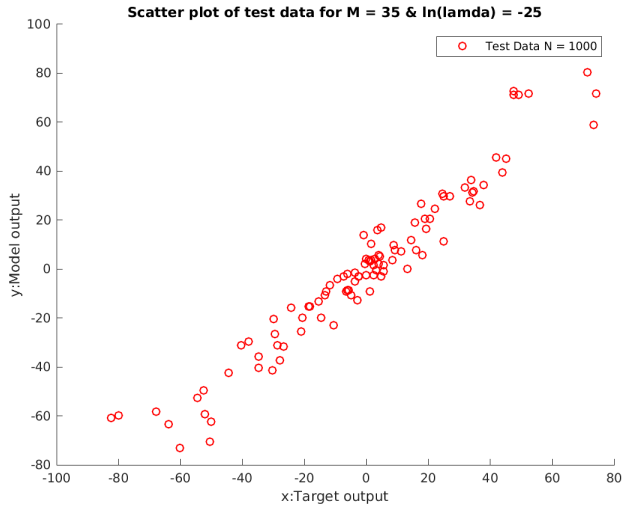


Figure 44: Scatter plots for train, validation and test data for N=1000, 2000

2.3.3 MLFFNN with 1 hidden layer for dataset 1

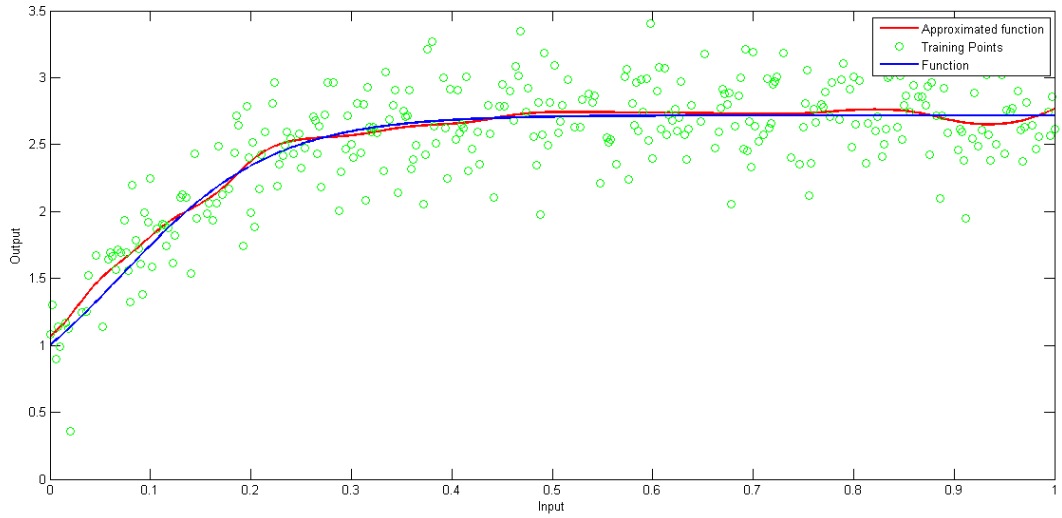


Figure 45: Approximated Function For 300 training samples (Number of nodes in hidden layer is 10)

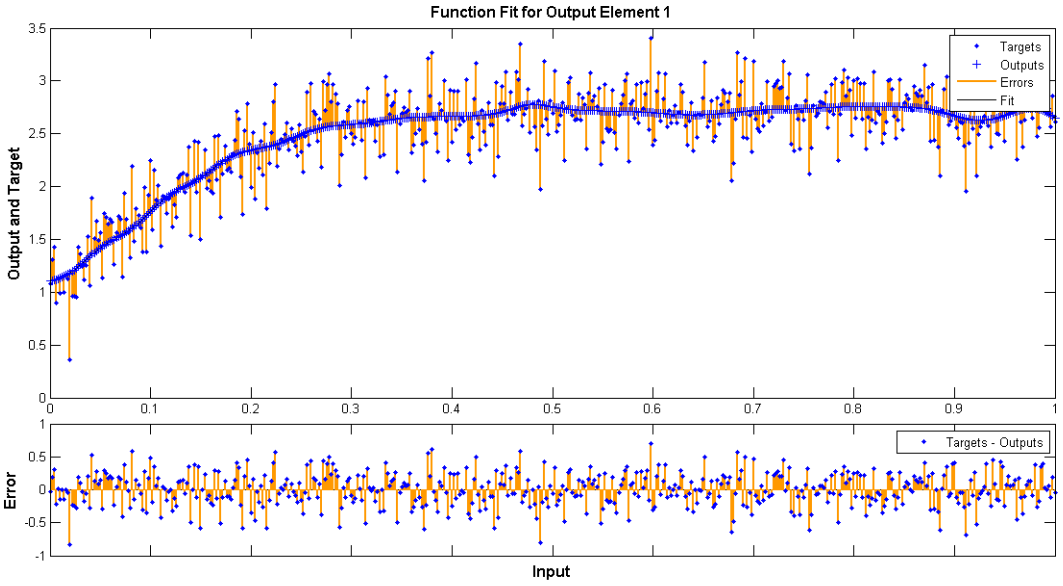


Figure 46: Plot showing error between the target and the model output values.

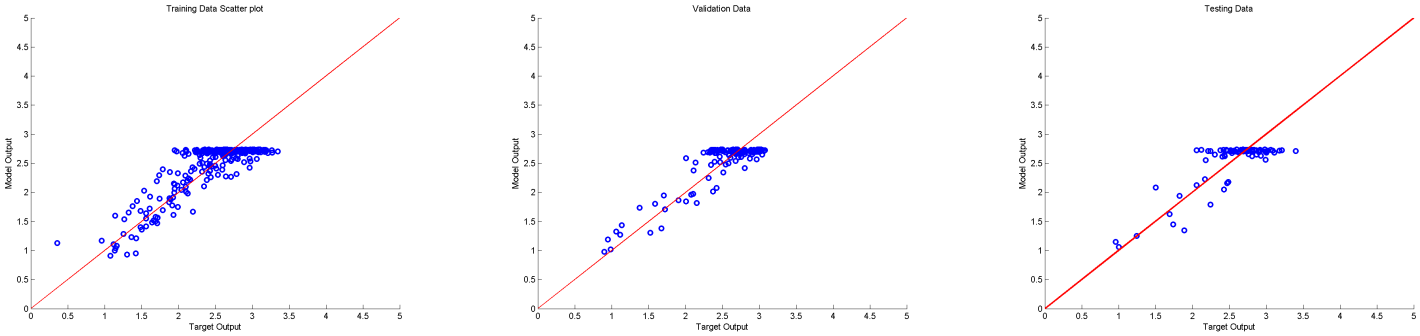


Figure 47: Scatter plot

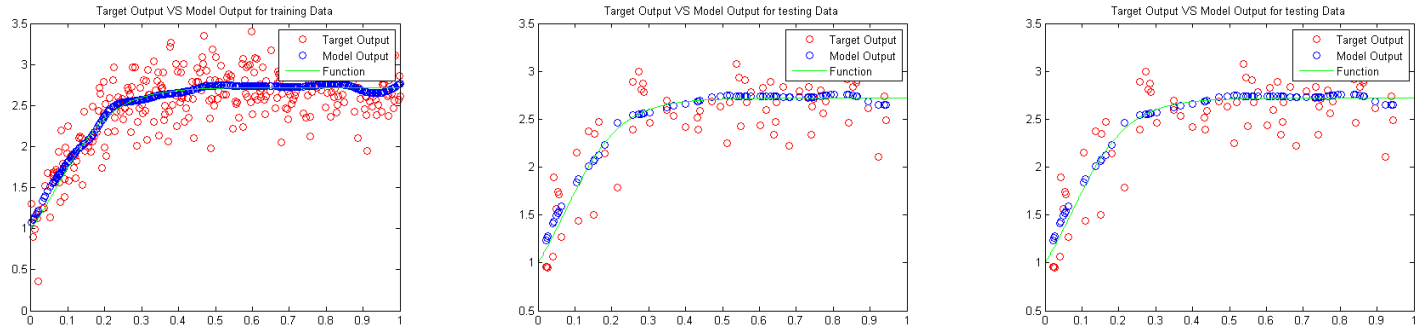


Figure 48: Plots showing Target Output Versus Model Output

2.3.4 MLFFNN with 2 hidden layers for Dataset 2

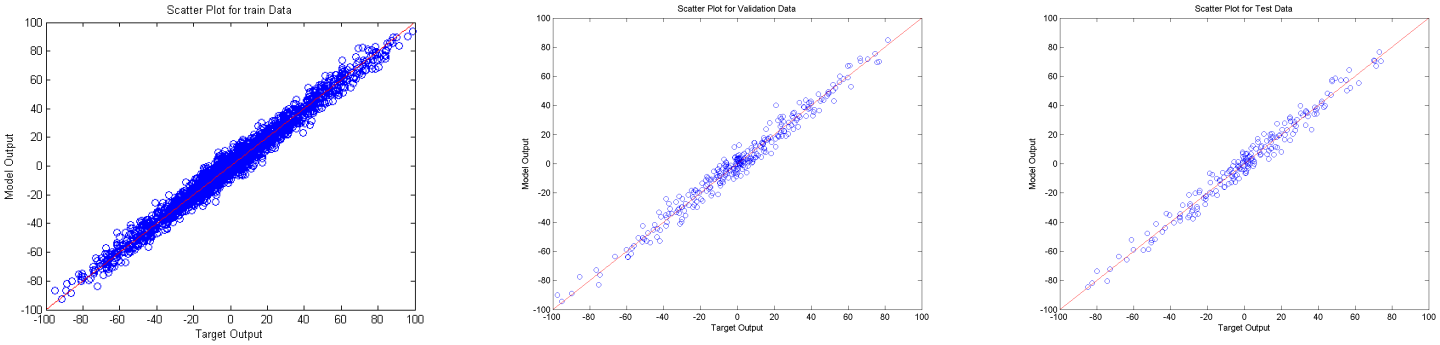


Figure 49: Scatter plots for train, validation and test data for 2000 training samples

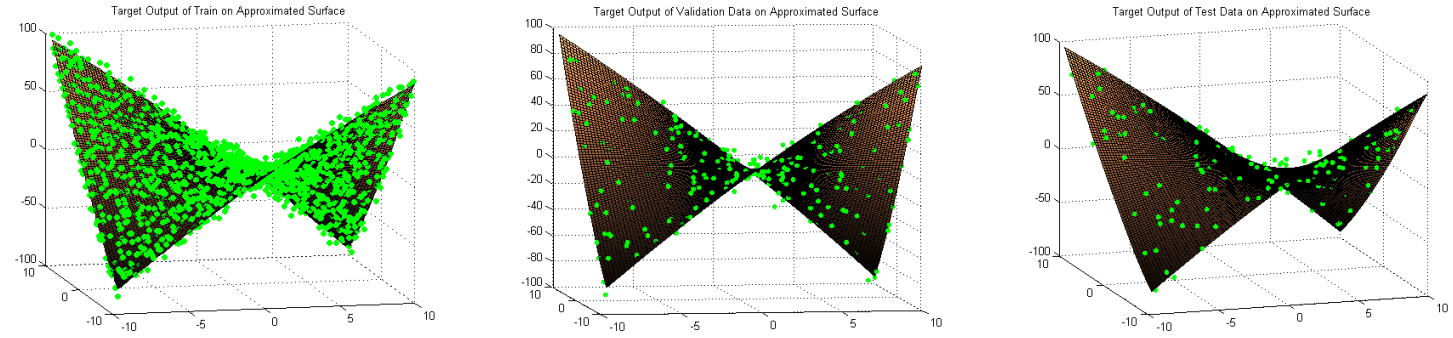


Figure 50: Plots showing the fit of Target values on the surface plotted using model outputs for train, validation and test data

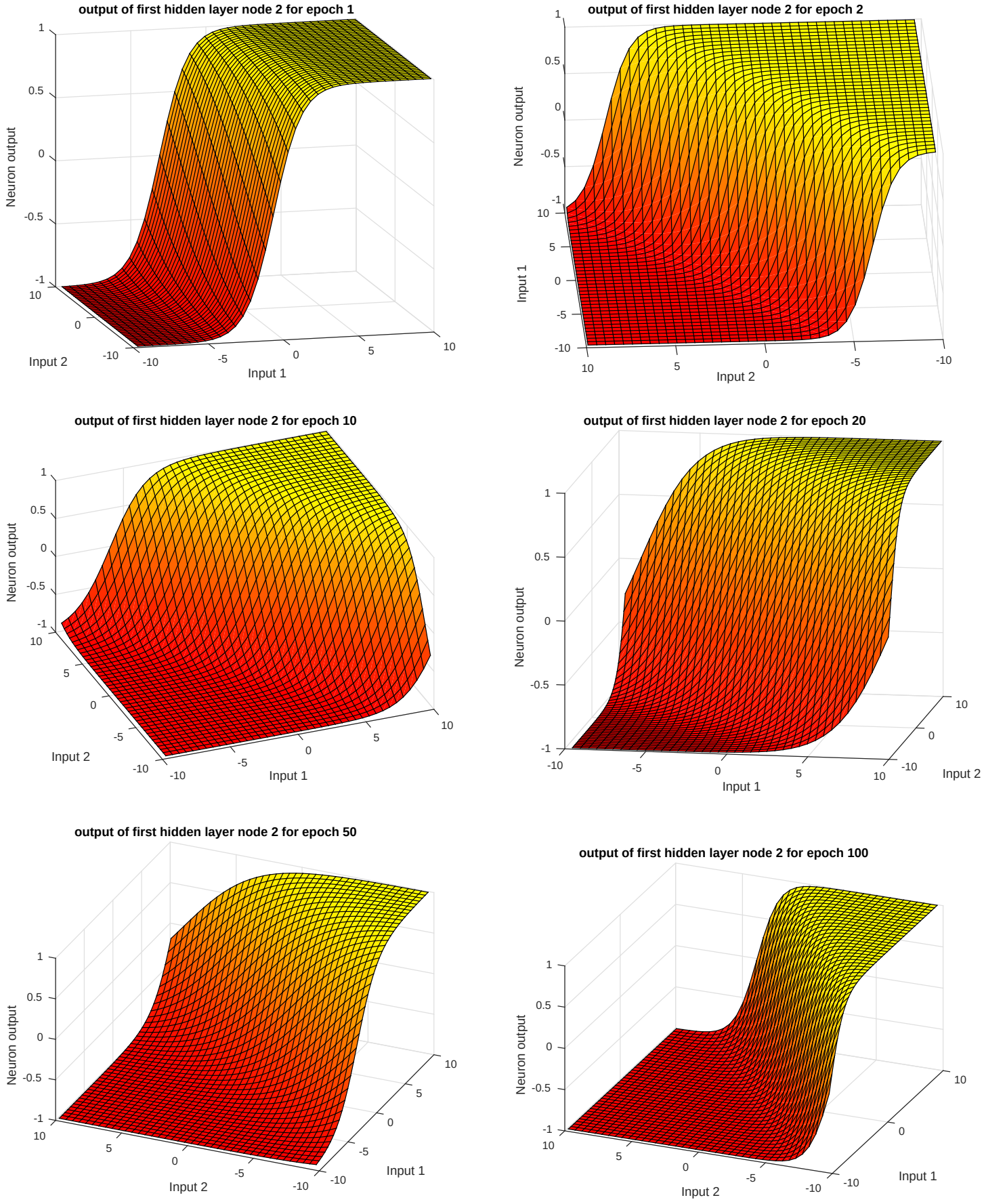


Figure 51: Output of first hidden layer node 2

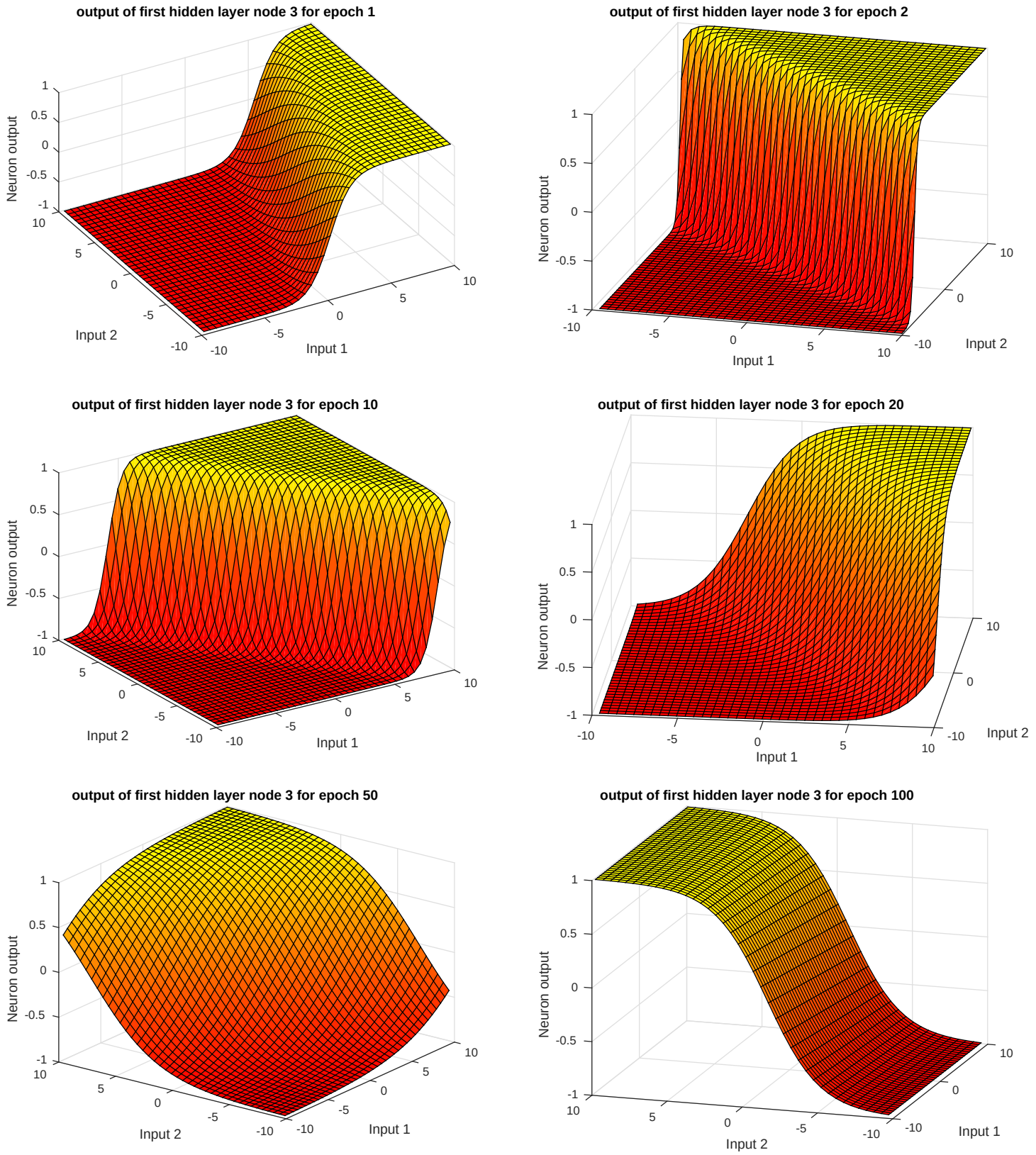
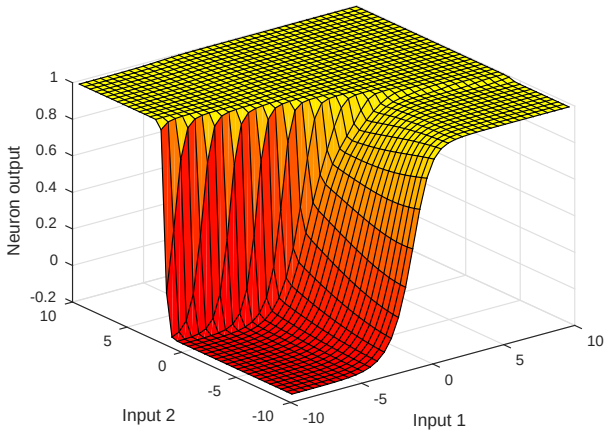
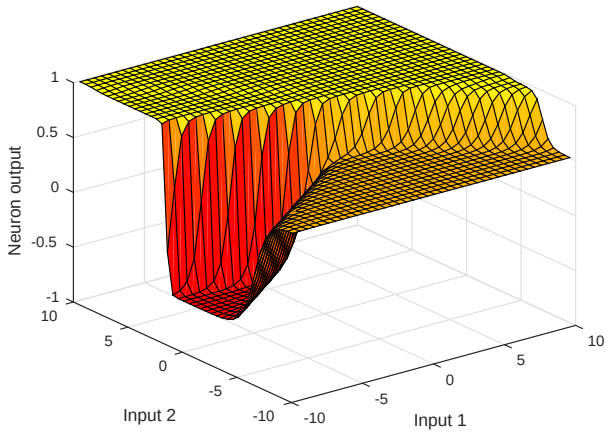


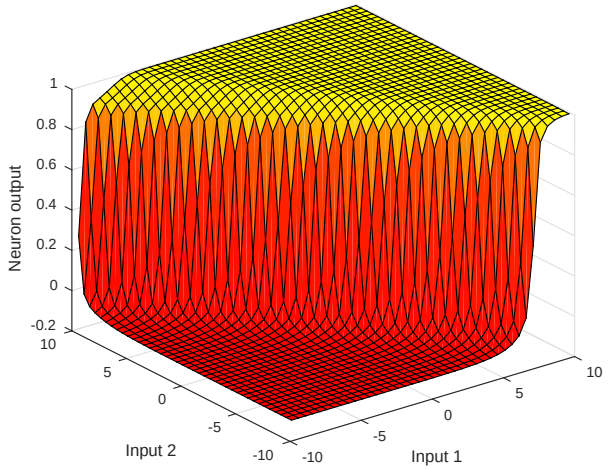
Figure 52: Output of first hidden layer node 3



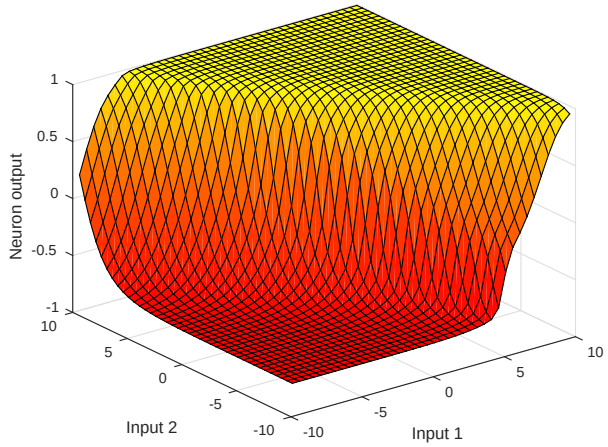
(a) Epoch 1



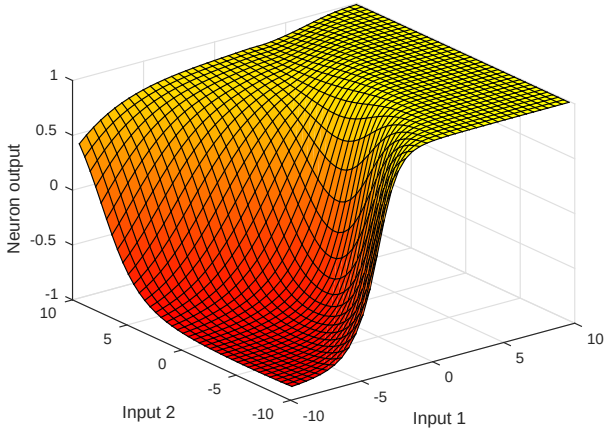
(b) Epoch 2



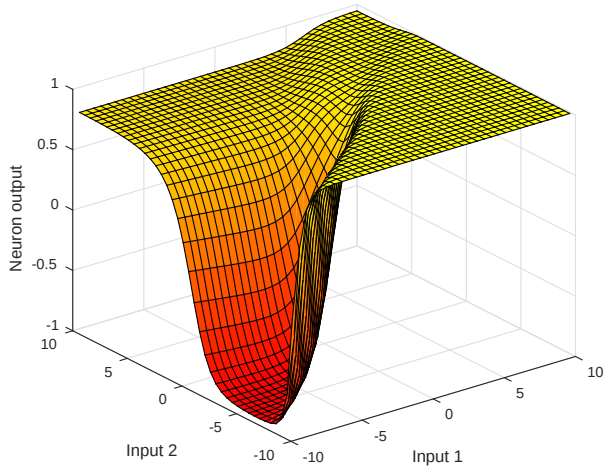
(c) Epoch 10



(d) Epoch 20

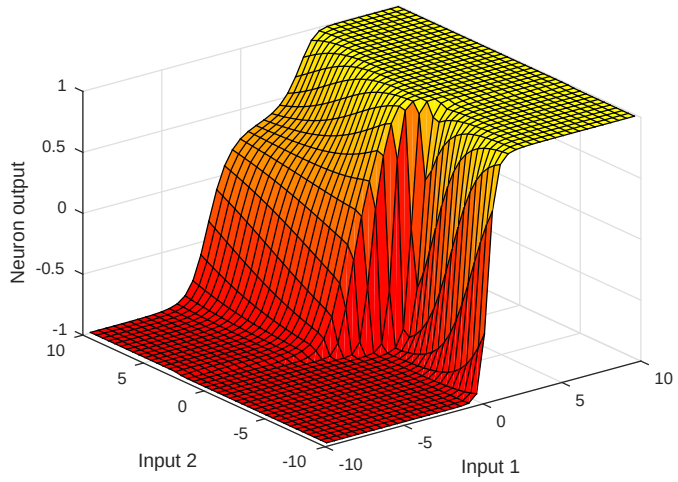


(e) Epoch 50

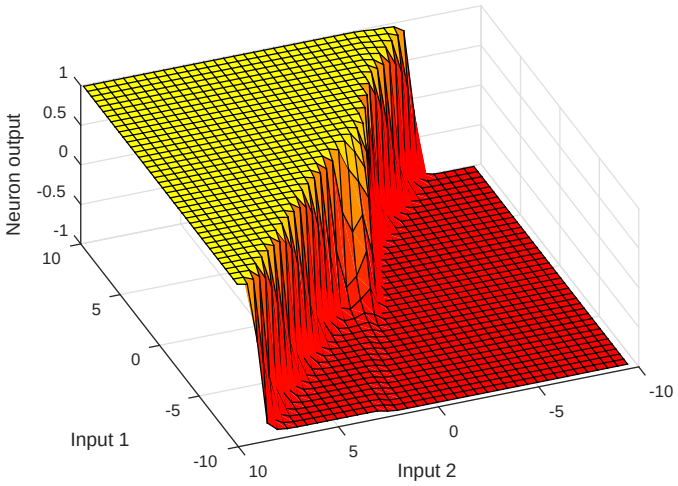


(f) Epoch 100

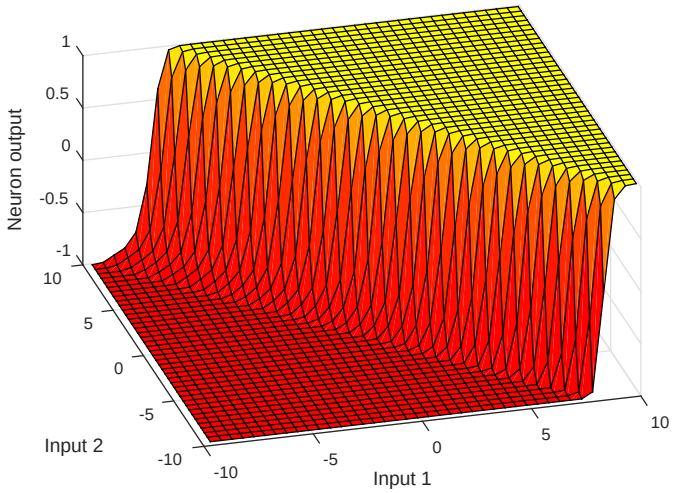
Figure 53: Output of second hidden layer node 1



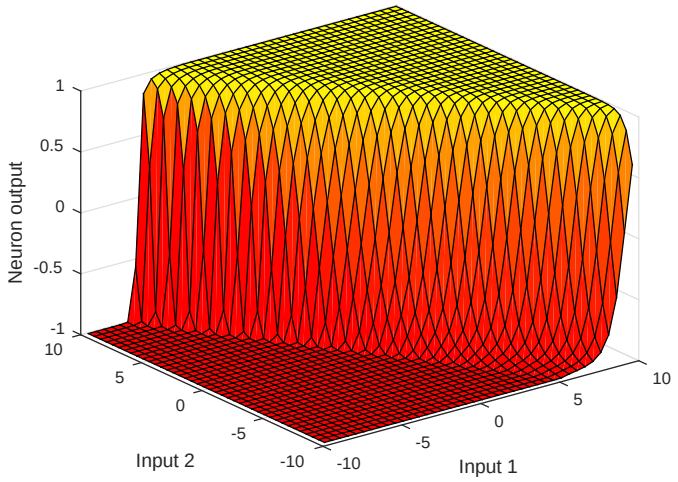
(a) Epoch 1



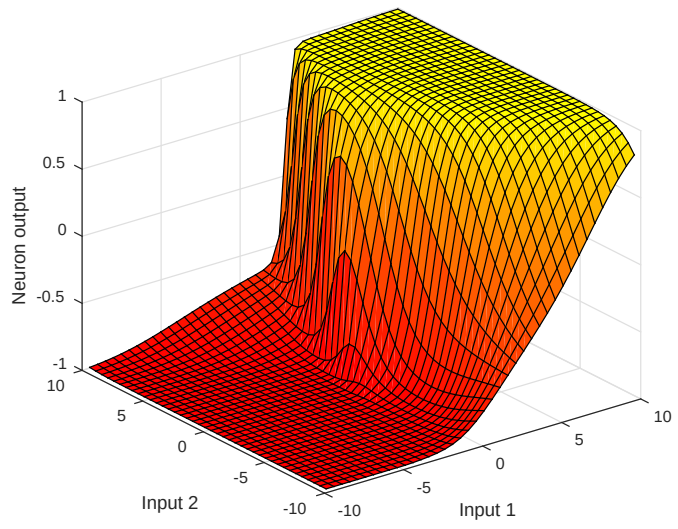
(b) Epoch 2



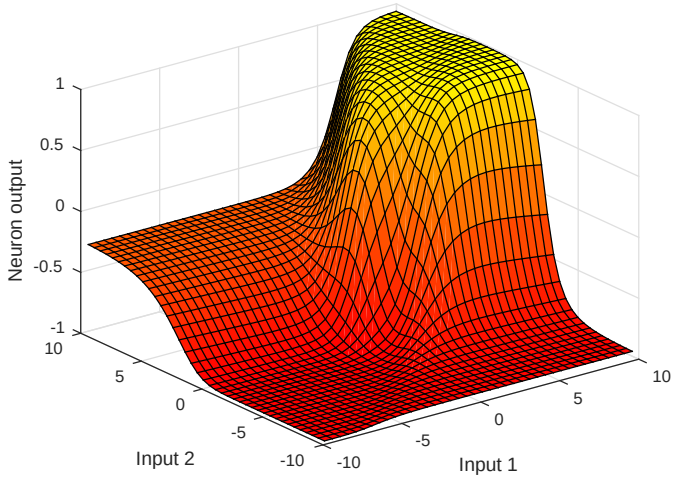
(c) Epoch 10



(d) Epoch 20

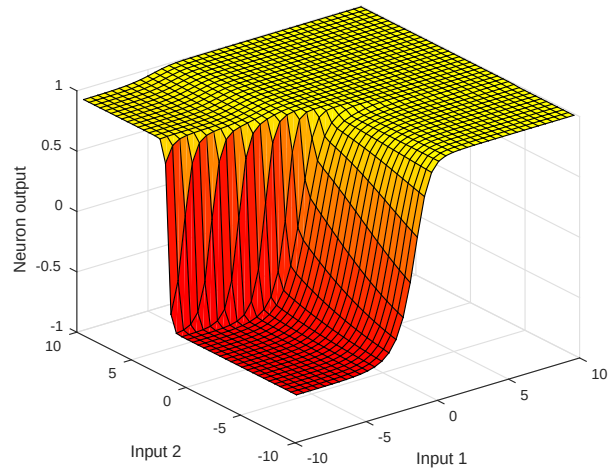


(e) Epoch 50

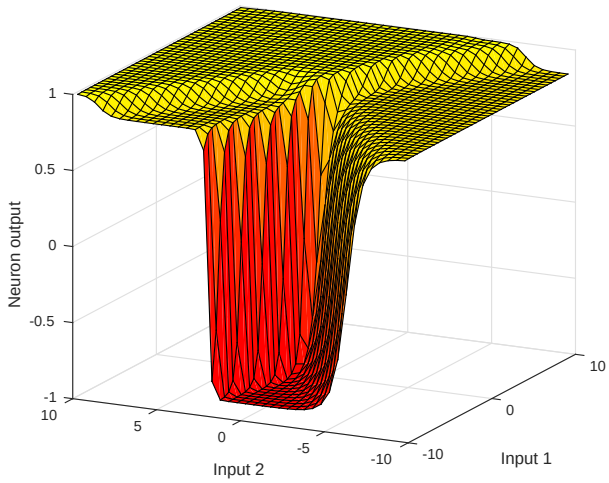


(f) Epoch 100

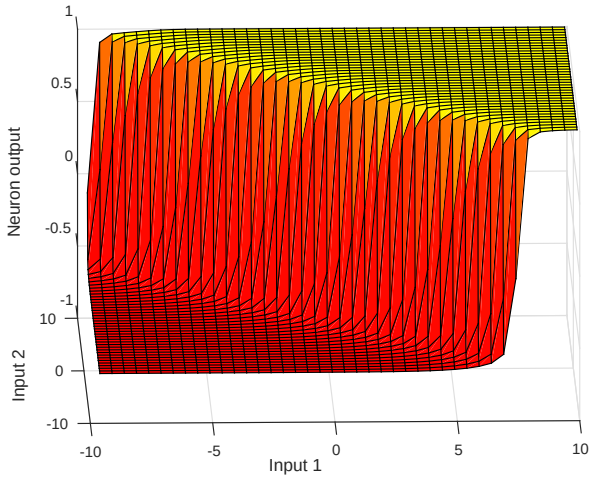
Figure 54: Output of second hidden layer node 2



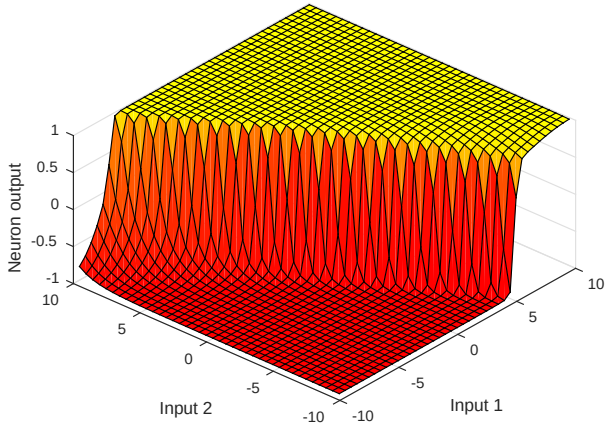
(a) Epoch 1



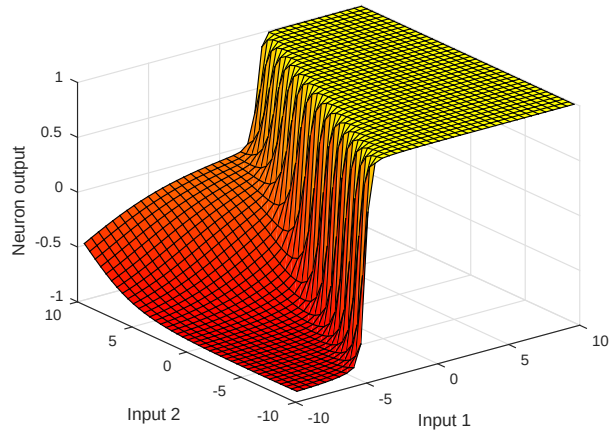
(b) Epoch 2



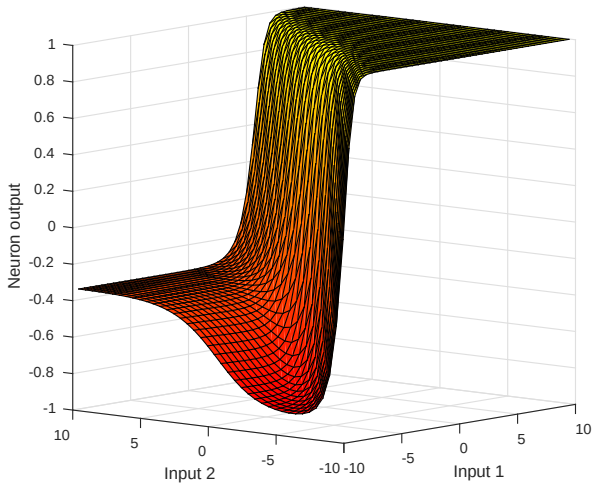
(c) Epoch 10



(d) Epoch 20

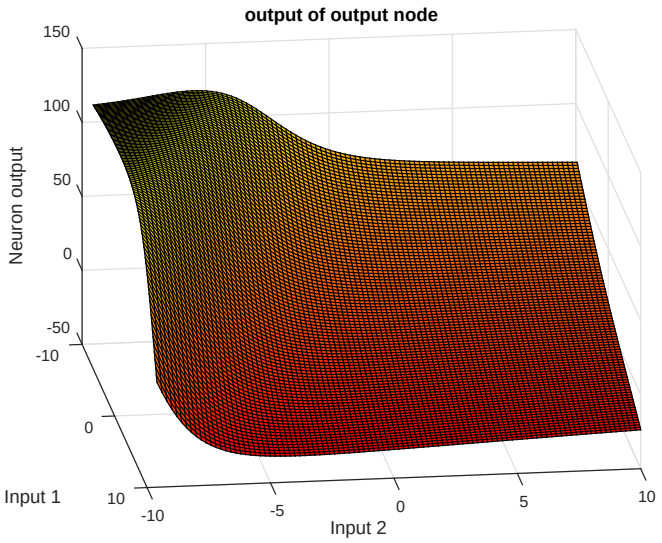


(e) Epoch 50

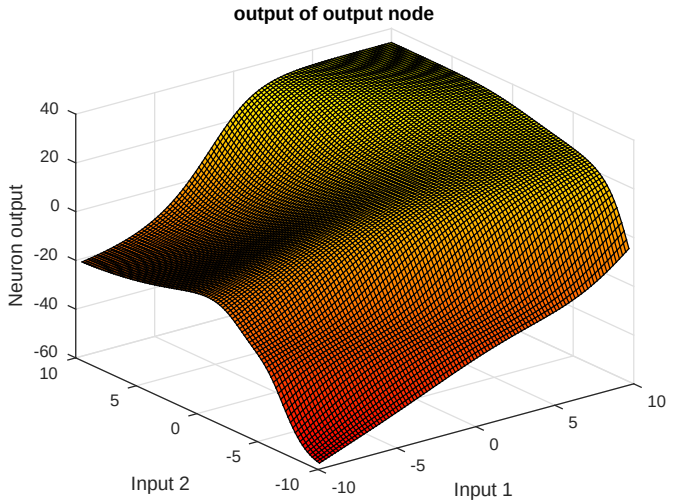


(f) Epoch 100

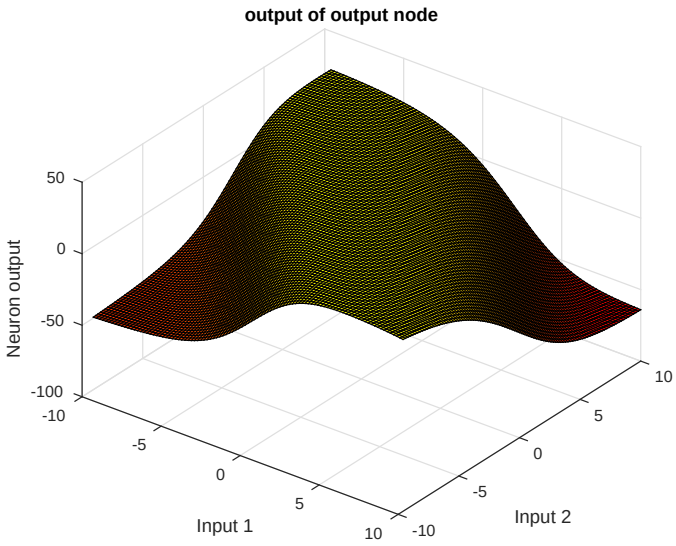
Figure 55: Output of second hidden layer node 3



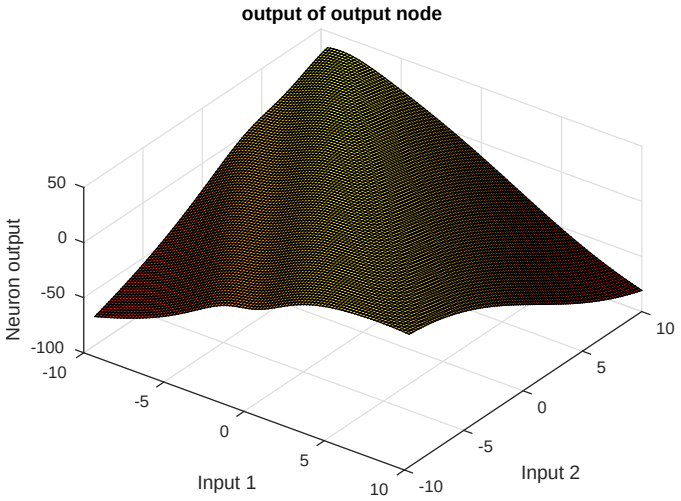
(a) Epoch 1



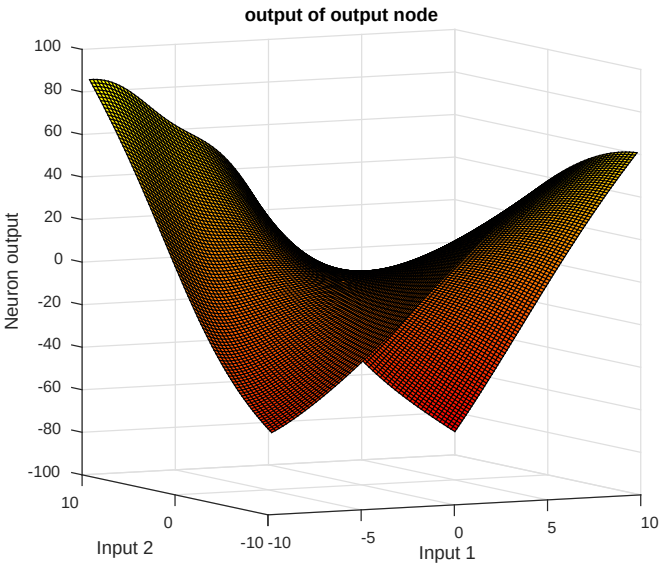
(b) Epoch 2



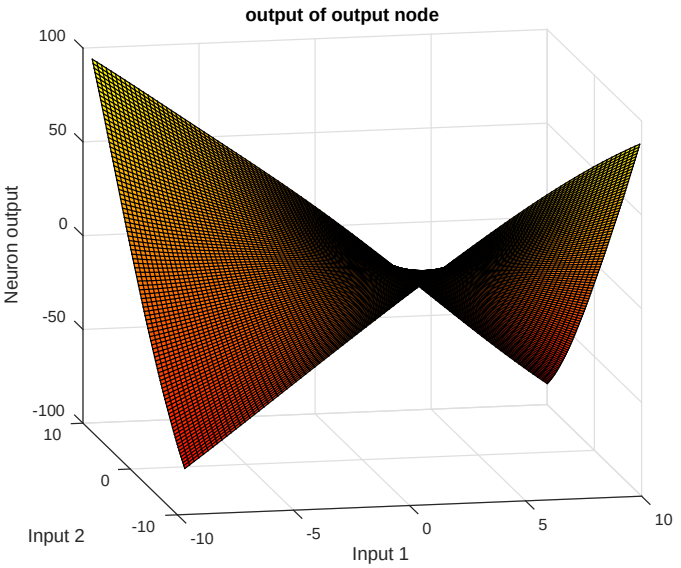
(c) Epoch 10



(d) Epoch 20



(e) Epoch 50



(f) Epoch 100

2.3.5 Generalized RBF model for Datasets 1

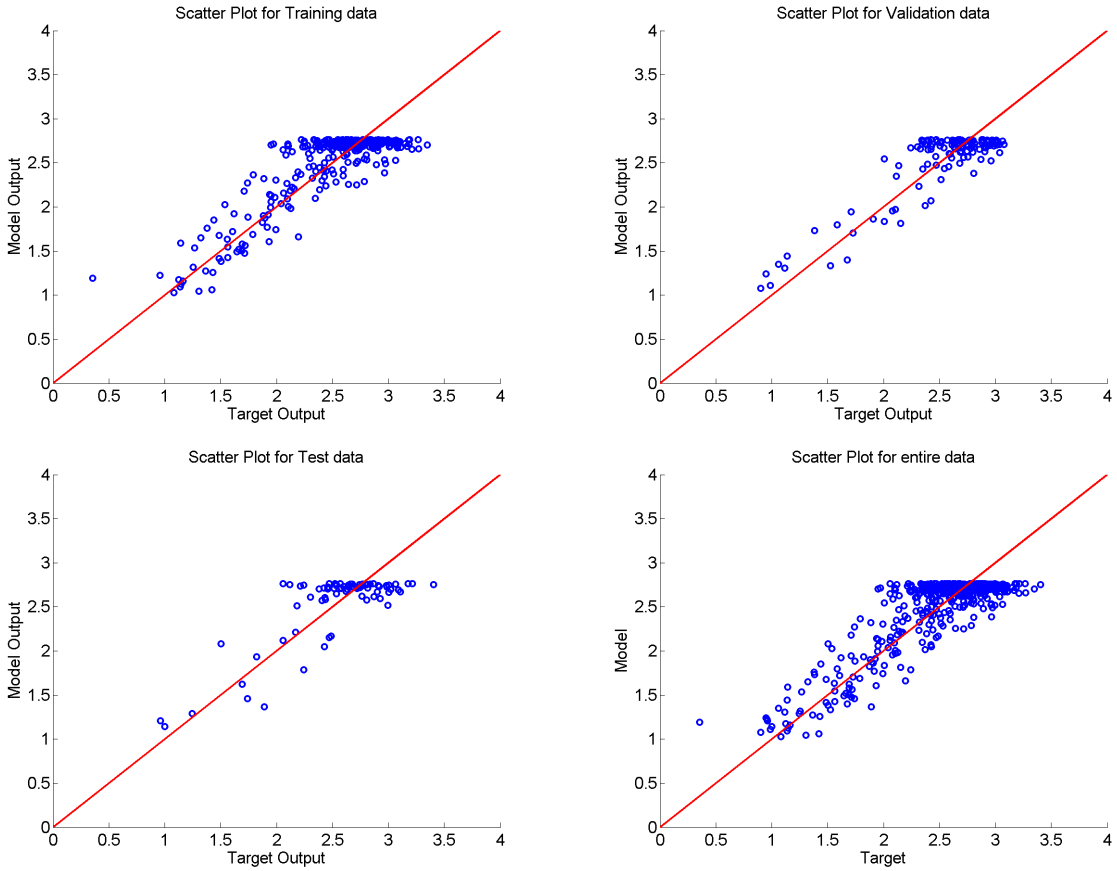
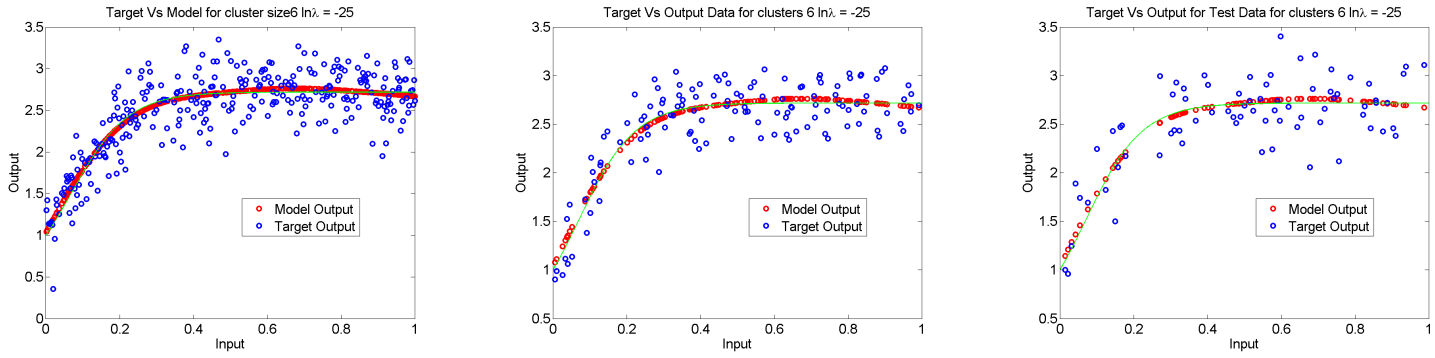


Figure 57: Scatter Plots



(a) Training Data (b) Validation Data (c) Test Data

Figure 58: Plots showing Target output vs model output

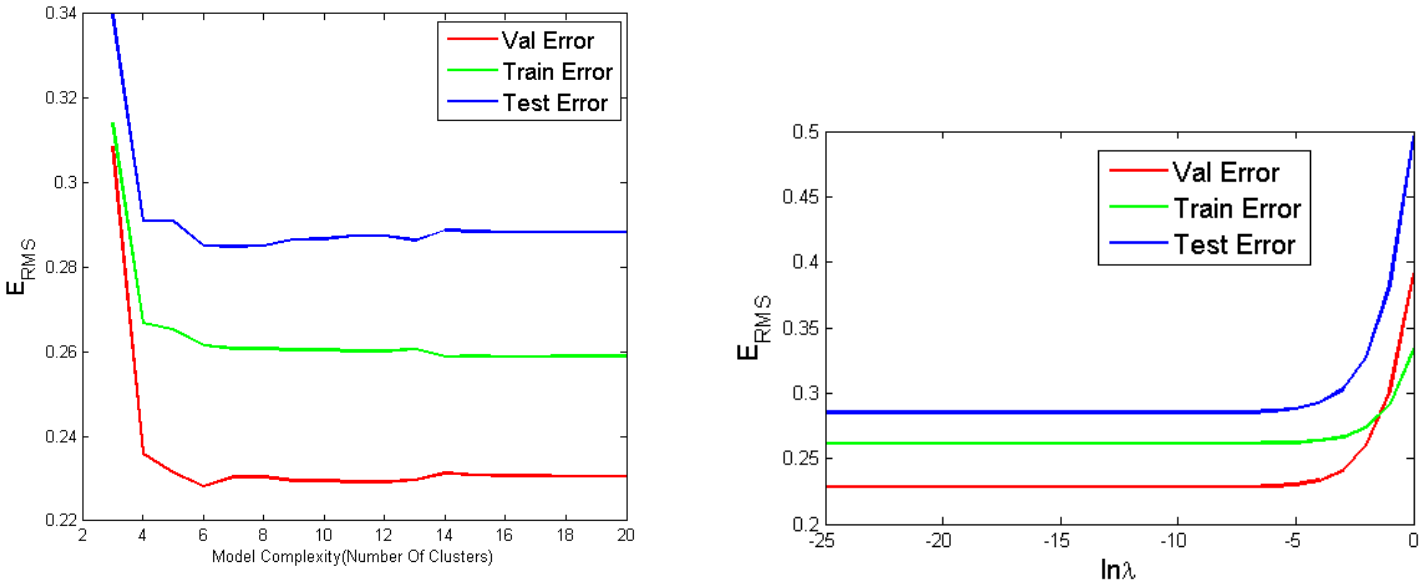


Figure 59: Variation of E_{RMS} based on model complexity and λ

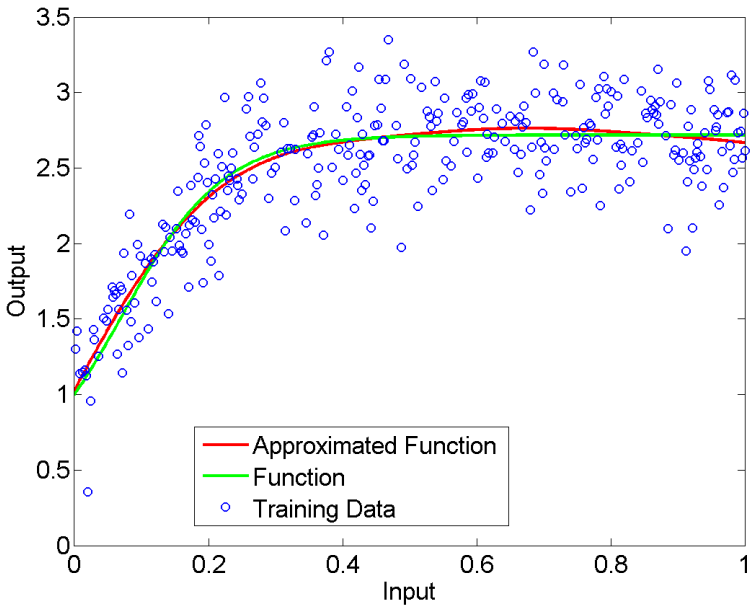
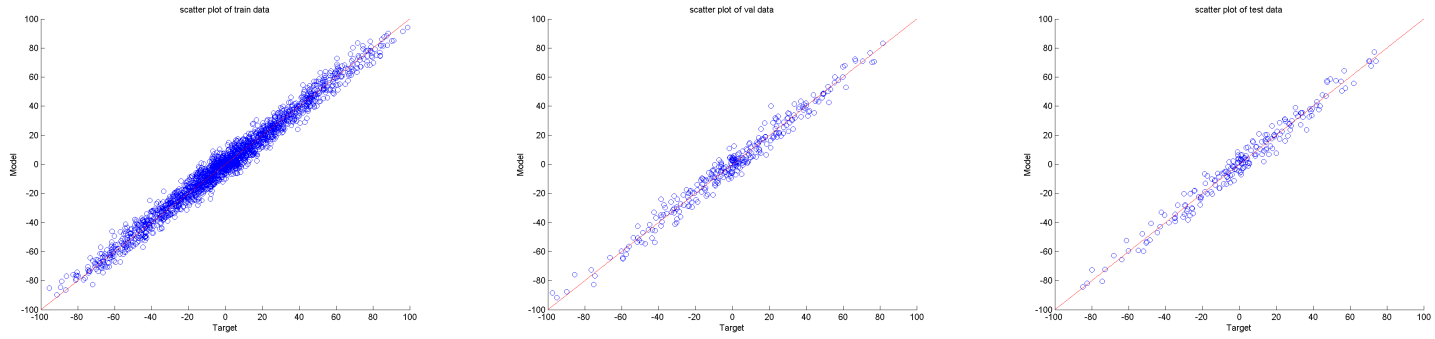


Figure 60: Plot of input vs output

2.3.6 Generalized RBF model for Datasets 2



(a) Training Data (b) Validation Data (c) Test Data

Figure 61: Scatter Plots

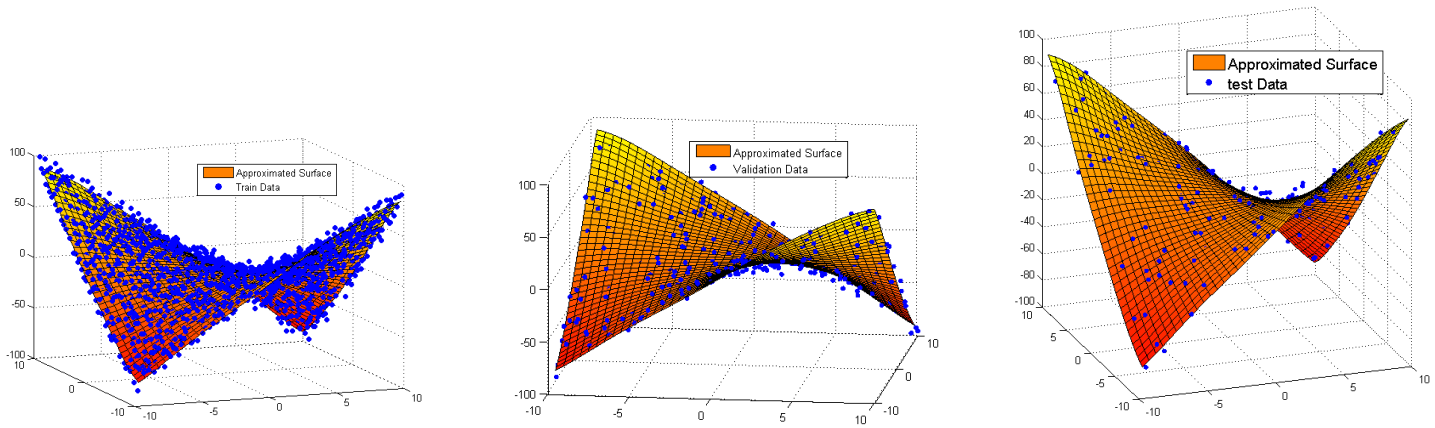


Figure 62: Plots showing the fit of Target values on the surface plotted using model outputs for train, validation and test data

2.4 Observations and Inferences

Polynomial curve fitting:

- For very small values of M, the approximation is bad. As the model complexity increases, the fit becomes more and more proper.
- When the model complexity is very high, it results in overfitting.
- Overfitting increases beyond a certain model complexity.
- We can observe that the weight values gets very high values when overfitting occurs.
- Regularisation controls this overfitting problem.
- Once we apply the regularisation, the weight values comes down to the normal range.

- For a particular model complexity the fit becomes better as we increase the dataset size.
- By reducing the choice of λ the E_{rms} decreases and suddenly shoots up.

Linear model:

- The overfitting occurs mainly because the model is trying to approximate a simple function with a complex function.
- The RMS error value decreases when large number of data points are used for training.
- The value of s is kept same for all the experiments as the varying of s results in different performance.
- With fewer data points N=20 and N=100 the model fits poorly for small model complexities but for higher values it fits excellently as the number of training points increases the underlying function is approximated the better.

MLFFNN:

- As the number of points increases the model obtained is a better approximation of the function.
- For bivariate data, gradient descent method the range of error was very high. but LM method error was lower.
- The number of hidden nodes are varied from 1 to 10 and the minimum validation error occurred for 10 hidden nodes.

RBFNN:

- For univariate data the optimum cluster size is 6 and regularisation parameter is $ln(\lambda) = -25$.
- For a particular regularisation parameter, the error decreases as we increase the number of clusters.
- For a particular cluster size the error decreases as the $ln(\lambda)$ decreases.
- For bivariate data it gives a good approximation for the target data.
- For bivariate data optimal cluster size is 32 and $ln\lambda$ is -28.

Of all the regression methods for univariate data only polynomial curve fitting gave better fit for smaller dataset