KMPA: Assignment 1 Report

Group 1: CS13B053, CS13B060, CS13B061

November 25, 2017

1 Classification

1.1 Linearly Separable Classes

1.1.1 Neural Network Model

A feedforward neural network with 2 hidden layers, consisting of 4 and 3 hidden nodes in the 1^{st} and 2^{nd} hidden layers, was used to classify the given bivariate 3-class data. The output nodes were linear, while the hidden nodes used the tanh activation function. The test set classification accuracy achieved was 100%.

1.1.2 Plot of Decision Boundaries

The decision boundary obtained by the trained neural network is shown below:

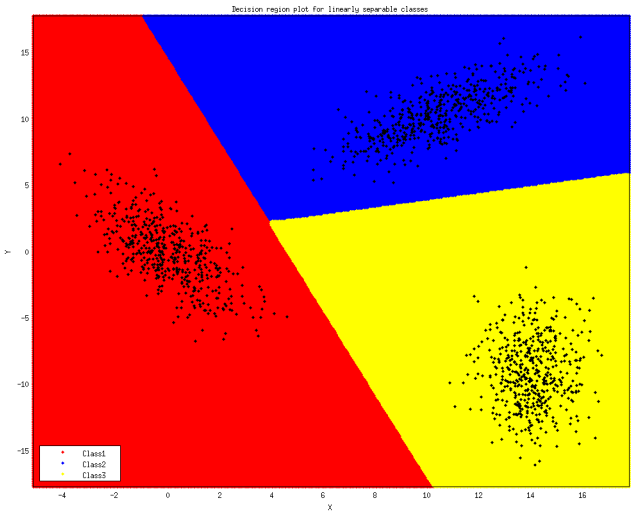


Figure 1: Decision boundary plot for linearly separable data.

1.1.3 Confusion Matrix

The confusion matrix is shown below:

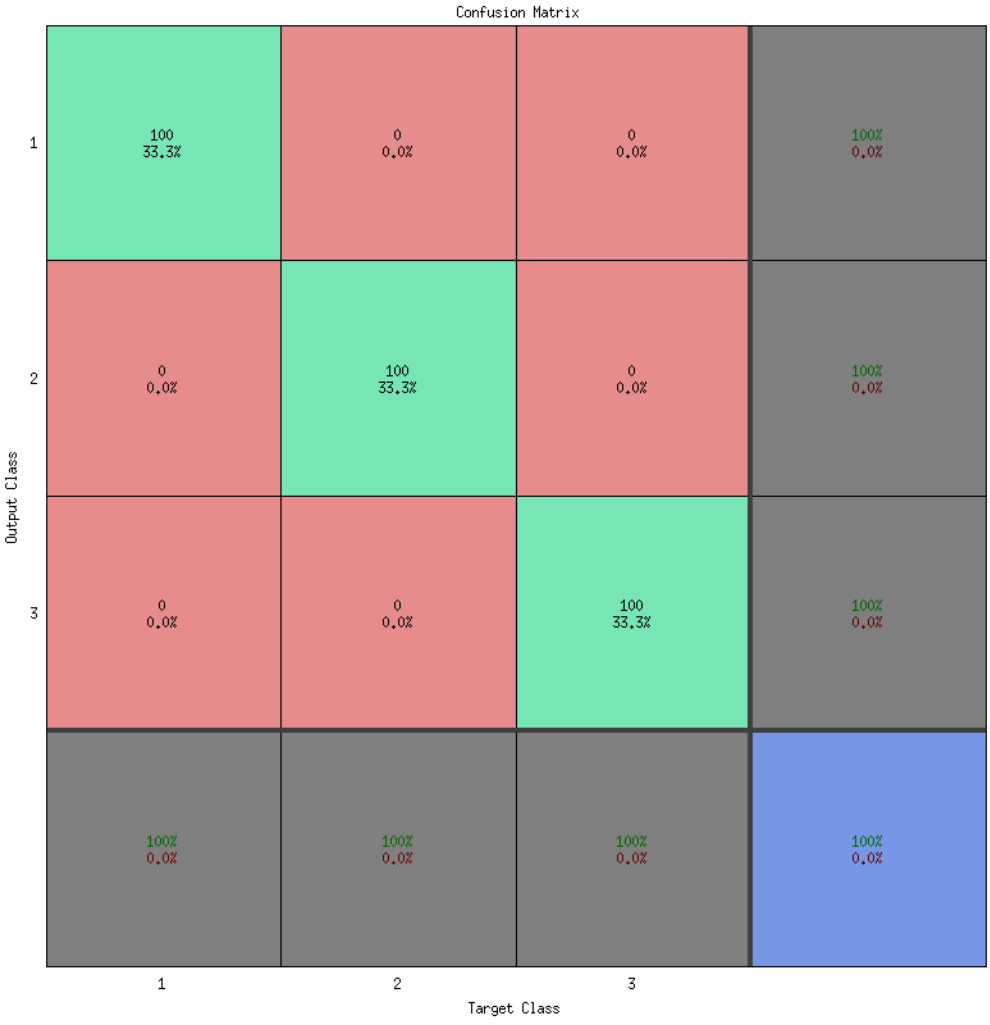


Figure 2: Confusion matrix for test data.

1.1.4 Observations & Inferences

It can be clearly seen that the linearly separable data is perfectly classified by the 2 layer neural network.

1.2 Nonlinearly Separable Classes

1.2.1 Neural Network Model

A feedforward neural network with 2 hidden layers, consisting of 6 and 5 hidden nodes in the 1^{st} and 2^{nd} hidden layers, was used to classify the given bivariate 3-class data. The output nodes were linear, while the hidden nodes used the tanh activation function. The test set classification accuracy achieved was 100%.

1.2.2 Plot of Decision Boundaries

The decision boundary obtained by the trained neural network is shown below:

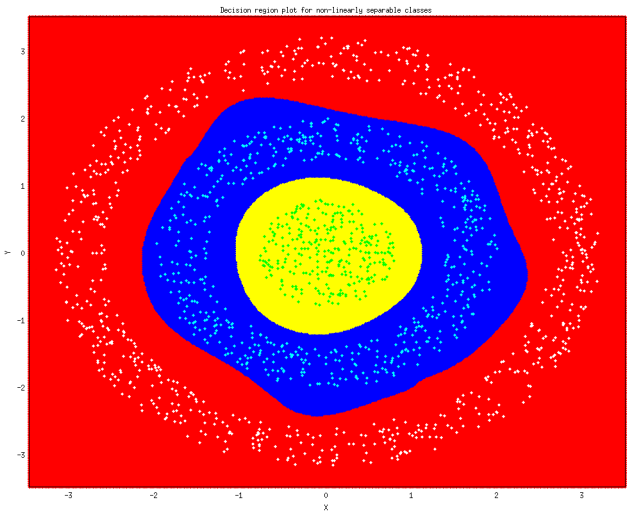


Figure 3: Decision boundary plot for nonlinearly separable data.

1.2.3 Confusion Matrix

The confusion matrix is shown below:

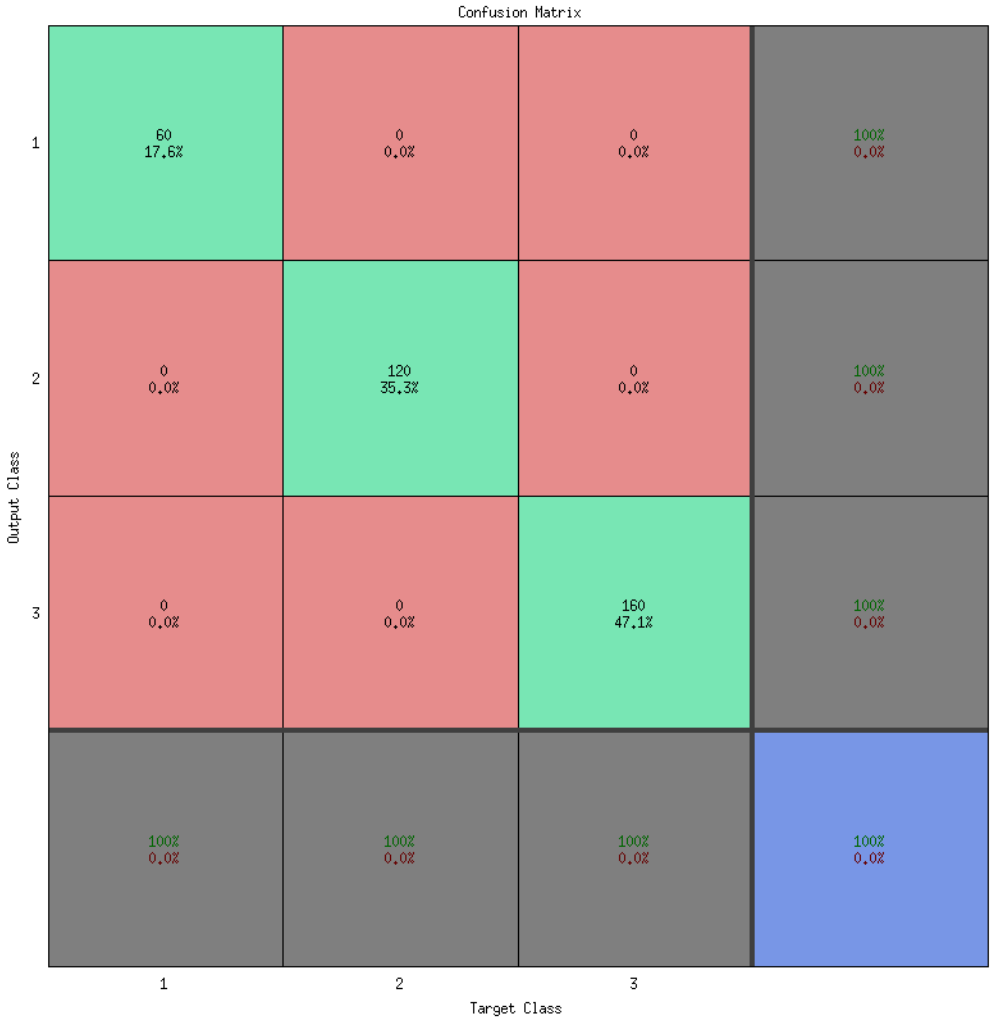
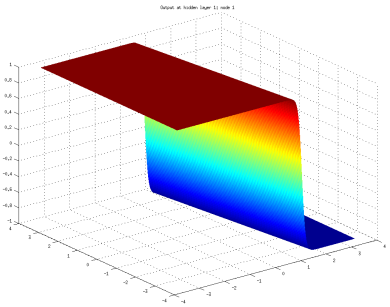


Figure 4: Confusion matrix for test data.

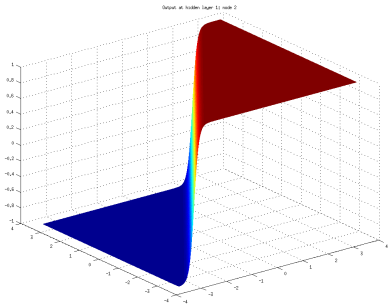
1.2.4 Output Plots for Hidden & Output Nodes

Given below are the plots for the outputs of the hidden nodes and output nodes during the training phase.

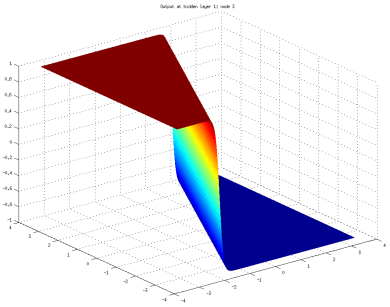
Epoch 1:



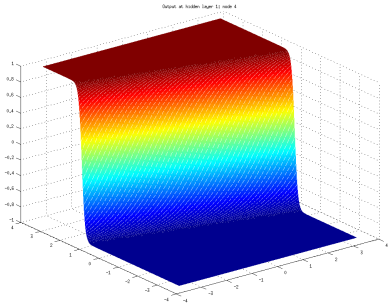
(a) Output plot for Node 1



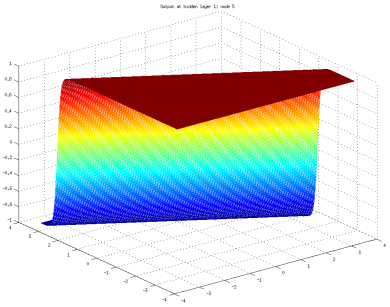
(b) Output plot for Node 2



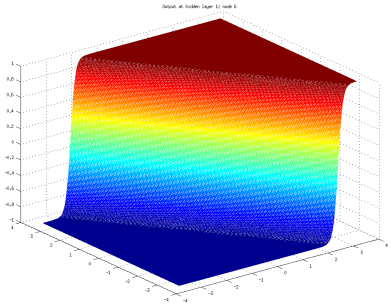
(c) Output plot for Node 3



(d) Output plot for Node 4

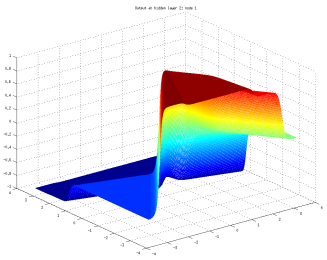


(e) Output plot for Node 5

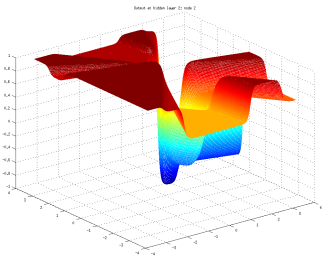


(f) Output plot for Node 6

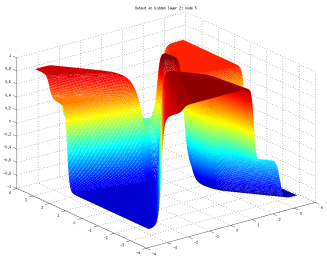
Figure 5: Output plots for hidden layer 1.



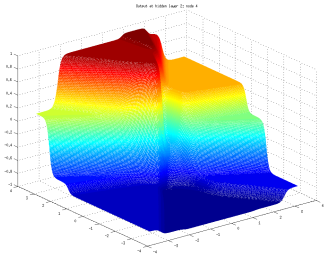
(a) Output plot for Node 1



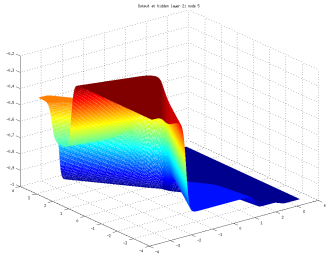
(b) Output plot for Node 2



(c) Output plot for Node 3

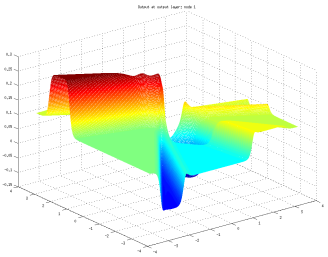


(d) Output plot for Node 4

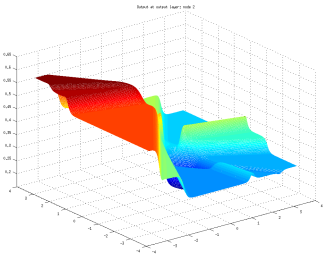


(e) Output plot for Node 5

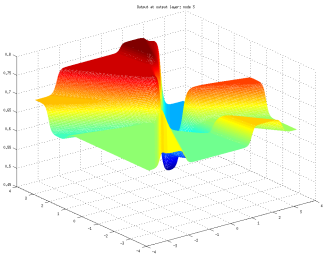
Figure 6: Output plots for hidden layer 2.



(a) Output plot for Node 1



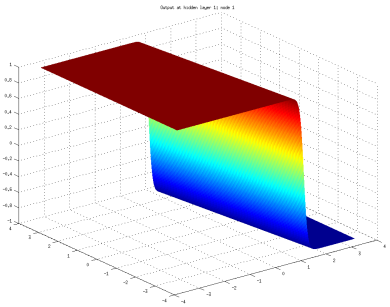
(b) Output plot for Node 2



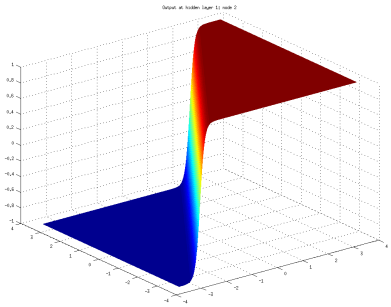
(c) Output plot for Node 3

Figure 7: Output plots for output layer.

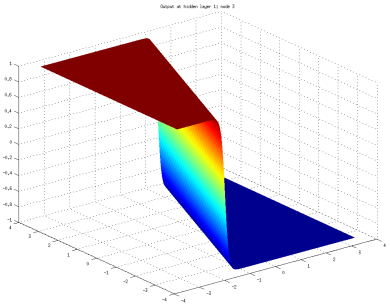
Epoch 2:



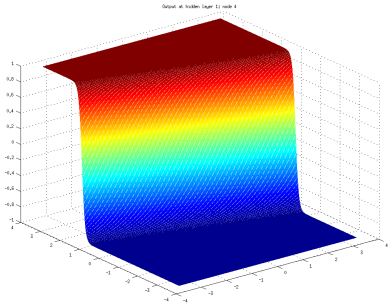
(a) Output plot for Node 1



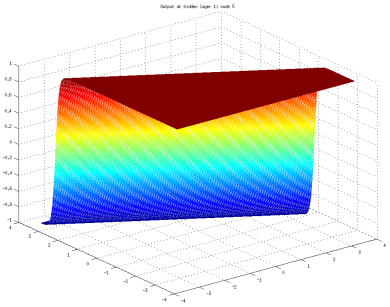
(b) Output plot for Node 2



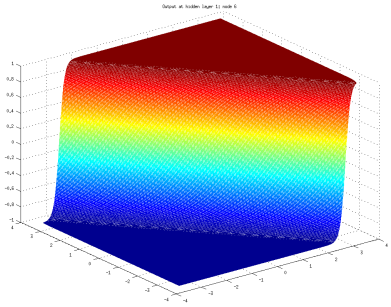
(c) Output plot for Node 3



(d) Output plot for Node 4

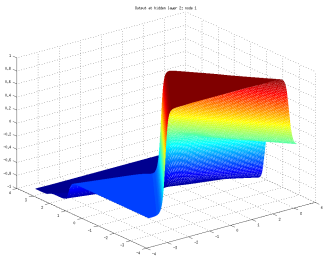


(e) Output plot for Node 5

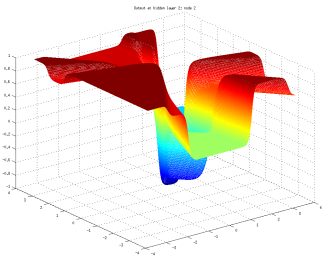


(f) Output plot for Node 6

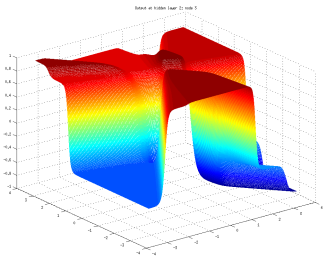
Figure 8: Output plots for hidden layer 1.



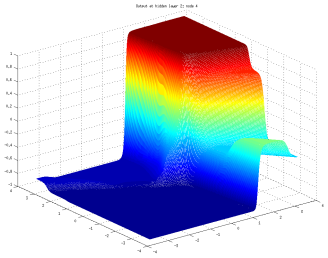
(a) Output plot for Node 1



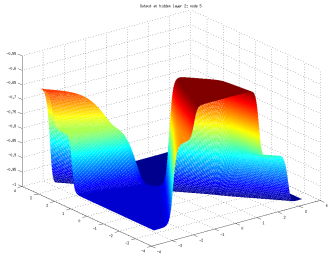
(b) Output plot for Node 2



(c) Output plot for Node 3

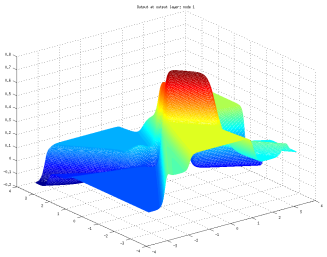


(d) Output plot for Node 4

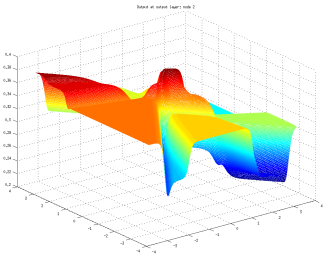


(e) Output plot for Node 5

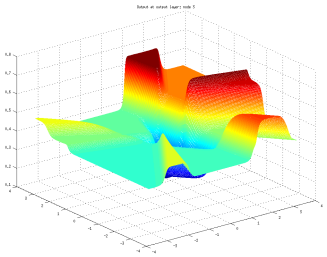
Figure 9: Output plots for hidden layer 2.



(a) Output plot for Node 1



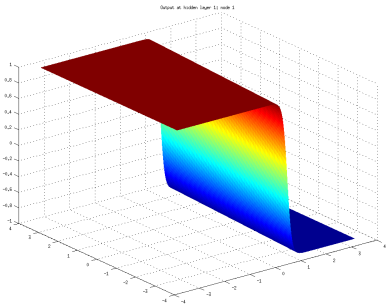
(b) Output plot for Node 2



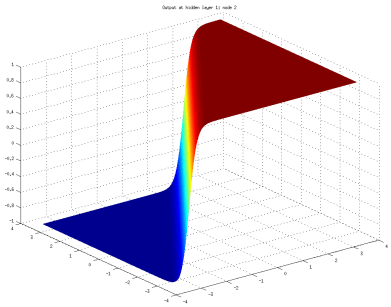
(c) Output plot for Node 3

Figure 10: Output plots for output layer.

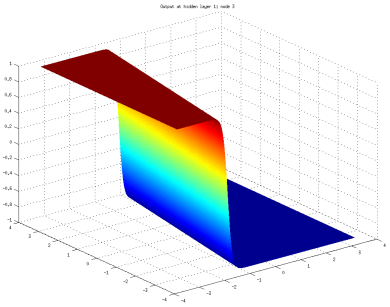
Epoch 10:



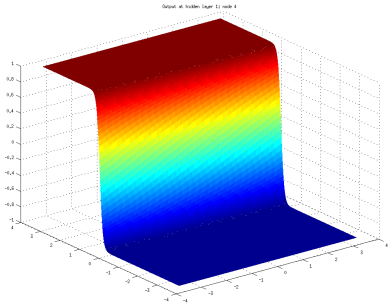
(a) Output plot for Node 1



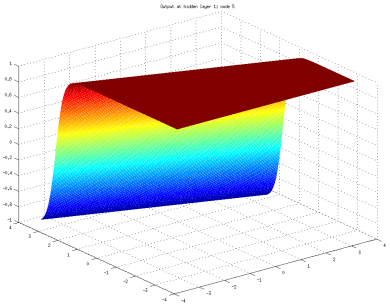
(b) Output plot for Node 2



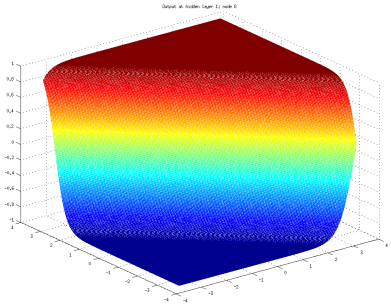
(c) Output plot for Node 3



(d) Output plot for Node 4

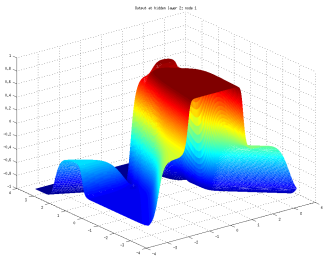


(e) Output plot for Node 5

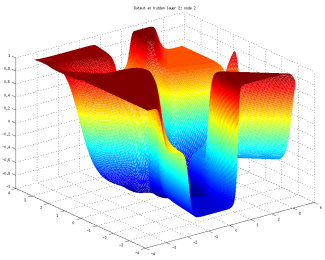


(f) Output plot for Node 6

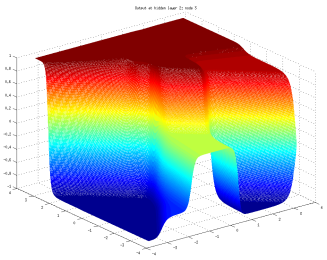
Figure 11: Output plots for hidden layer 1.



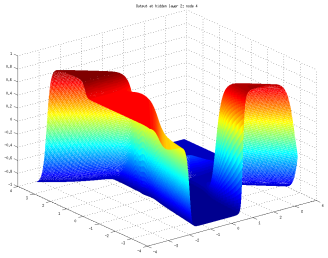
(a) Output plot for Node 1



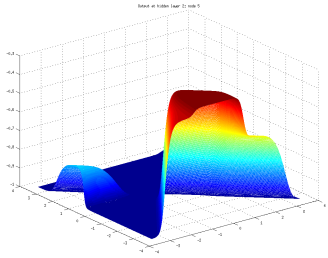
(b) Output plot for Node 2



(c) Output plot for Node 3

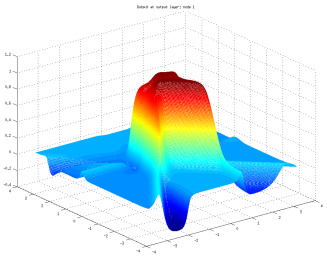


(d) Output plot for Node 4

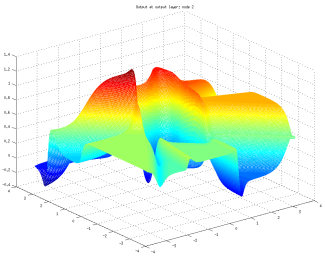


(e) Output plot for Node 5

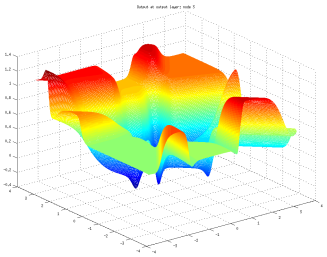
Figure 12: Output plots for hidden layer 2.



(a) Output plot for Node 1



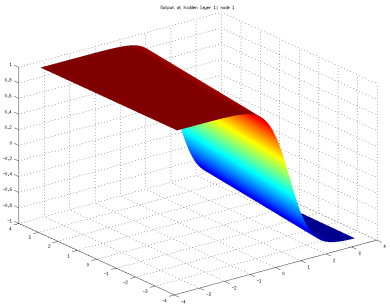
(b) Output plot for Node 2



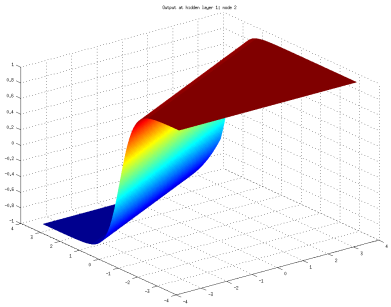
(c) Output plot for Node 3

Figure 13: Output plots for output layer.

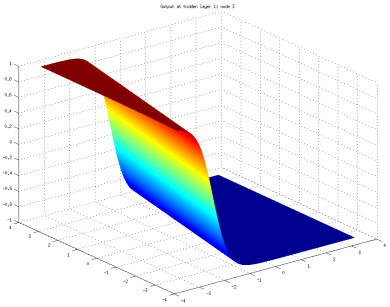
Epoch 50:



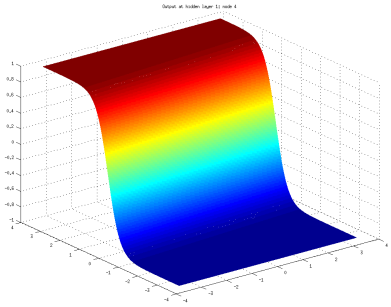
(a) Output plot for Node 1



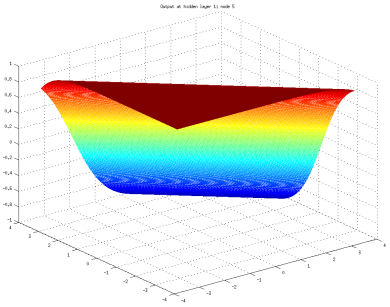
(b) Output plot for Node 2



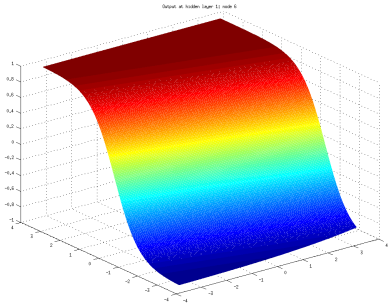
(c) Output plot for Node 3



(d) Output plot for Node 4

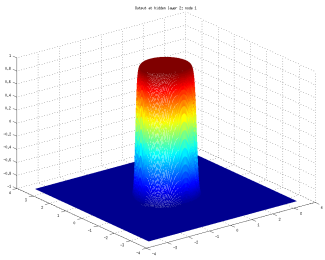


(e) Output plot for Node 5

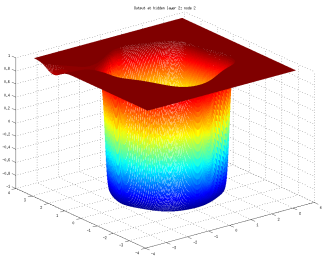


(f) Output plot for Node 6

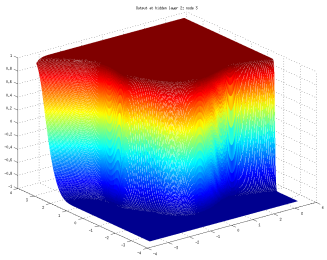
Figure 14: Output plots for hidden layer 1.



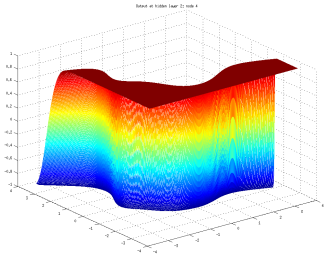
(a) Output plot for Node 1



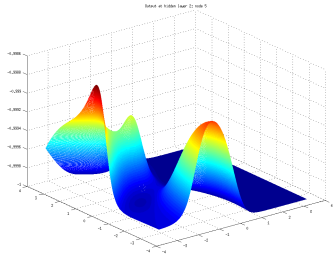
(b) Output plot for Node 2



(c) Output plot for Node 3

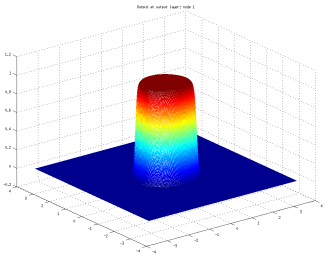


(d) Output plot for Node 4

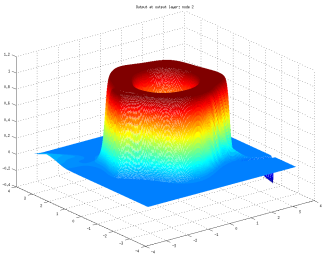


(e) Output plot for Node 5

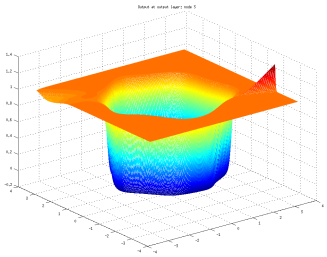
Figure 15: Output plots for hidden layer 2.



(a) Output plot for Node 1



(b) Output plot for Node 2



(c) Output plot for Node 3

Figure 16: Output plots for output layer.

The training stops before 50 epochs, hence the above plots are also the same plots obtained for 100 epochs, and post training plots.

1.2.5 Observations and Inferences

It can be seen that the neural network is able to perfectly classify the nonlinearly separable data. The learning progresses steadily as the number of epochs

increase by introducing increasing nonlinearity in the outputs of the nodes. This can be seen from the plots of the node outputs.

1.3 Overlapping Classes

1.3.1 Neural Network Model

A feedforward neural network with 2 hidden layers, consisting of 8 and 4 hidden nodes in the 1^{st} and 2^{nd} hidden layers, was used to classify the given bivariate 4-class data. The output nodes were linear, while the hidden nodes used the tanh activation function. The test set classification accuracy achieved was 90.2%.

1.3.2 Plot of Decision Boundaries

The decision boundary obtained by the trained neural network is shown below:

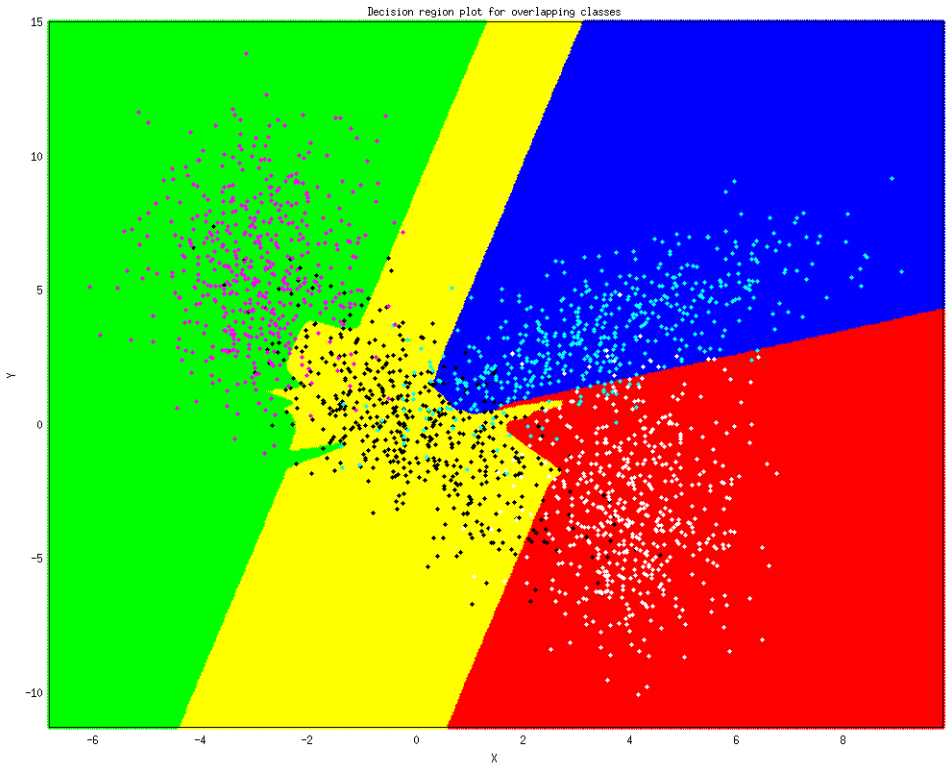


Figure 17: Decision boundary plot for overlapping data.

1.3.3 Confusion Matrix

The confusion matrix is shown below:

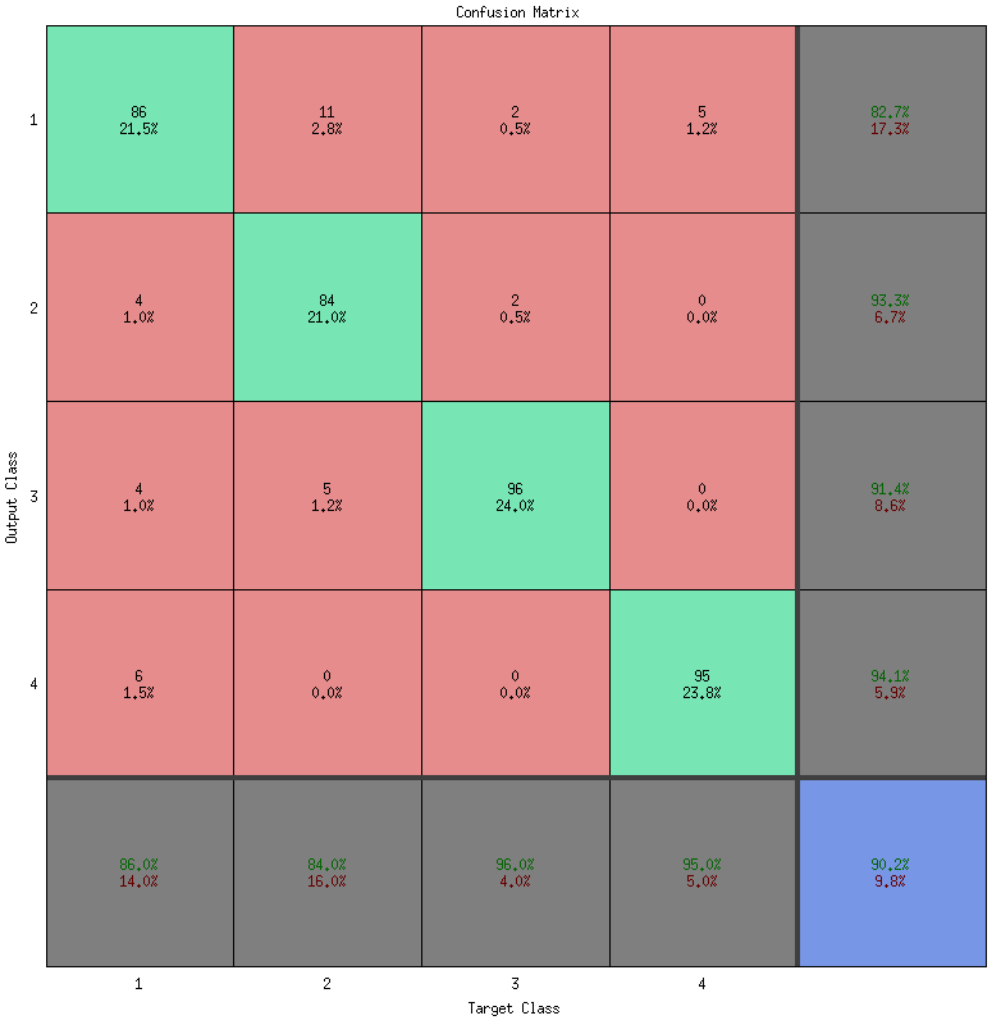


Figure 18: Confusion matrix for test data.

1.3.4 Observations and Inferences

The neural network does a good job of coming up with the decision boundaries for the overlapping classes, but doesn't do perfect classification. The classes are probably too mixed up to be distinguishable by a shallow network like this one.

1.4 Image Data

1.4.1 Neural Network Model

A feedforward neural network with 2 hidden layers, consisting of 40 and 20 hidden nodes in the 1^{st} and 2^{nd} hidden layers, was used to classify the given image data. The output nodes were linear, while the hidden nodes used the tanh activation function. The test set classification accuracy achieved was 63.8%.

1.4.2 Confusion Matrix

The confusion matrix is shown below:

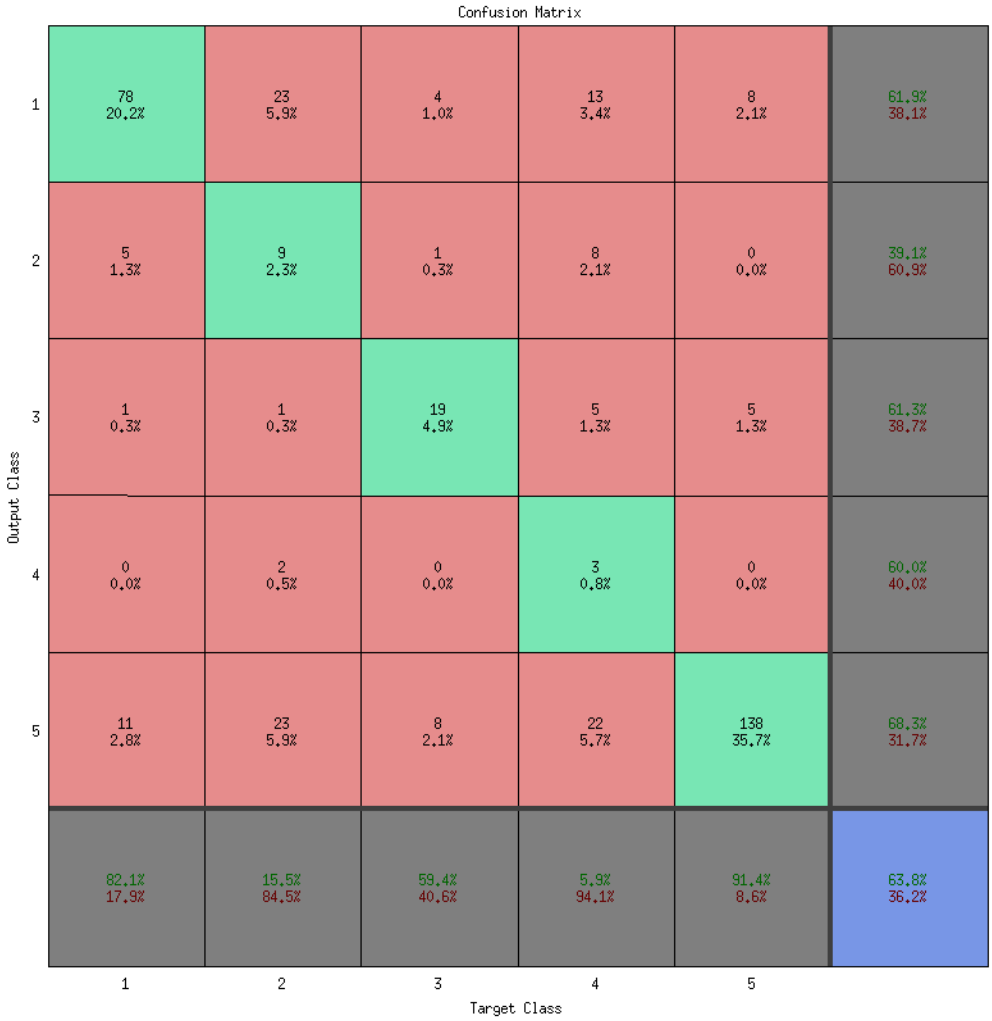


Figure 19: Confusion matrix for test data.

1.4.3 Observations and Inferences

It can be clearly seen that the best performing neural network performs considerably worse on the image dataset than the previous datasets. This could be because the image dataset is inherently more complex and higher dimensional than the simpler datasets we've seen so far. If we use a deeper network, then we might have better classification than the current best results. Ideally, we should

use a convolutional neural network to tackle the image classification problem. Another possibility is that the features provided for the image dataset aren't very "good", and we lose some information while using these features.

2 Regression

2.1 Polynomial Curve Fitting

2.1.1 Dataset

The dataset used for polynomial curve fitting is a univariate dataset obtained by sampling the function $f(x) = e^{cos(2\pi x)+sin(2\pi x)}$ and adding Gaussian noise with $\mu = 0$ and $\sigma = 0.08$.

2.1.2 Regression using Polynomial Basis Function

If we use polynomial basis functions, along with a squared loss with L2 regularization, we obtain the following optimal weights:

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

Here, Φ is the $N \times (D + 1)$ design matrix, with the n^{th} row containing the monomial terms $1, x_n^1, ..., x_n^D$.
t is the vector of desired outputs, i.e., $t_n = f(x_n) + \epsilon$, where ϵ is the added Gaussian noise.
λ is the regularization hyperparameter.

2.1.3 Plots of Approximated Function

The following are the plots of the approximated function, obtained for varying model complexities (degree of polynomial), training dataset sizes, and regularization parameter λ.

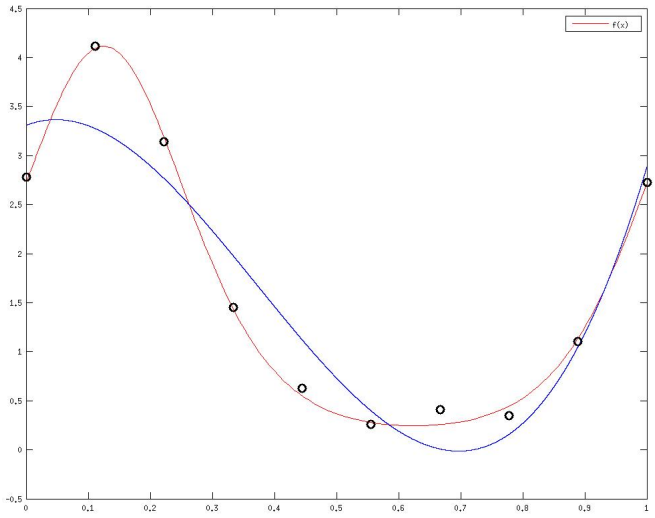


Figure 20: Output function for degree 3, training set size 10, and $\lambda = 0$.

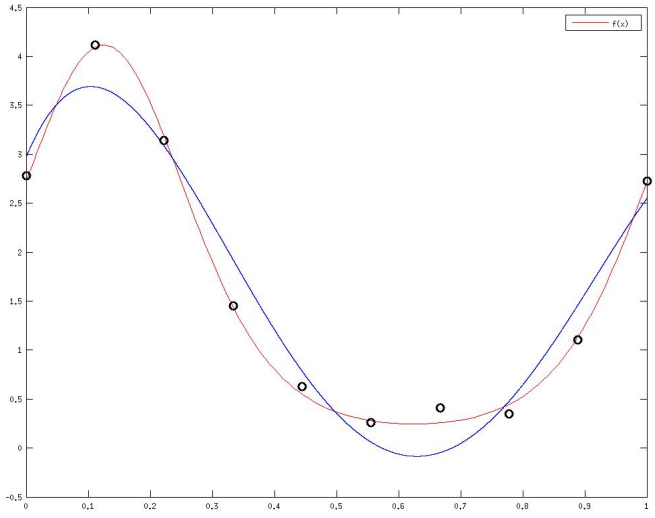


Figure 21: Output function for degree 4, training set size 10, and $\lambda = 0$.

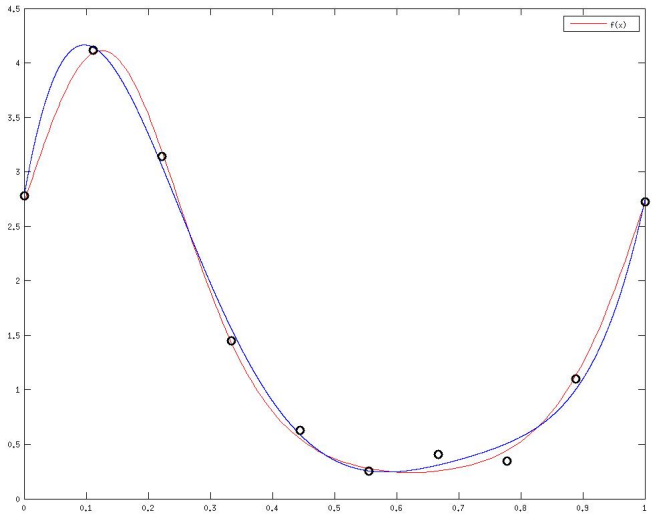


Figure 22: Output function for degree 5, training set size 10, and $\lambda = 0$.

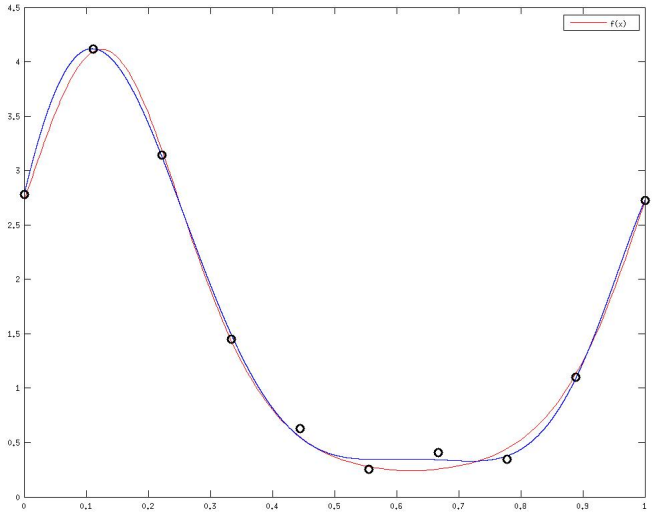


Figure 23: Output function for degree 7, training set size 10, and $\lambda = 0$.

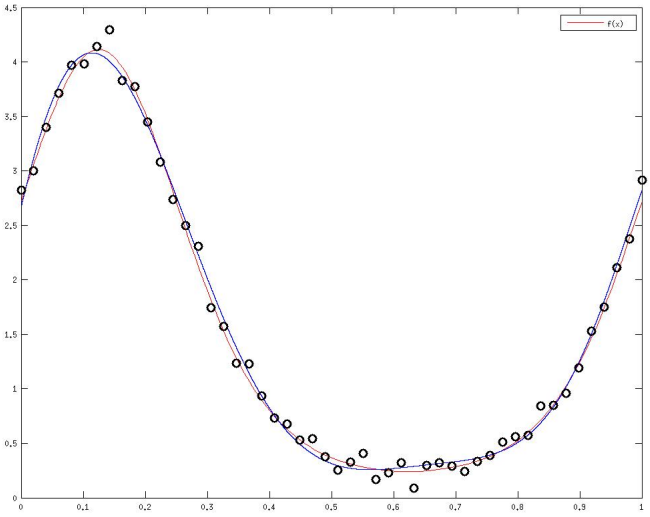


Figure 24: Output function for degree 7, training set size 50, and $\lambda = 0$.

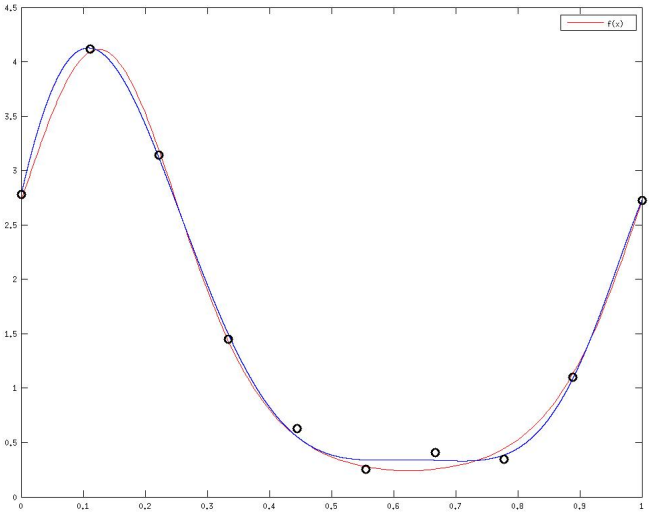


Figure 25: Output function for degree 7, training set size 10, and $\lambda = 10^{-10}$.

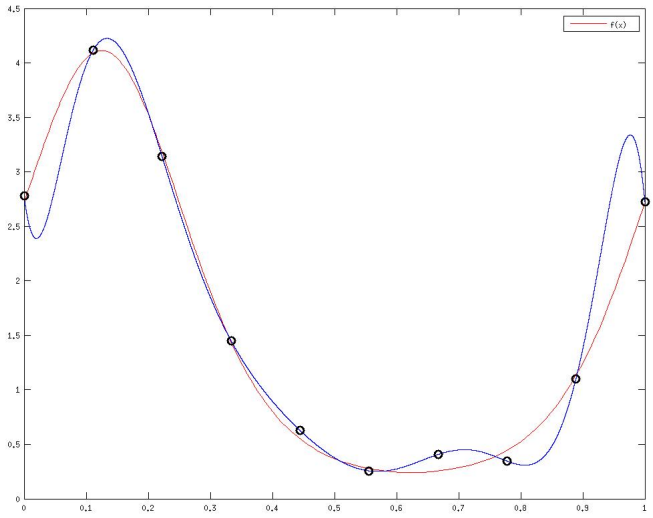


Figure 26: Output function for degree 9, training set size 10, and $\lambda = 0$.

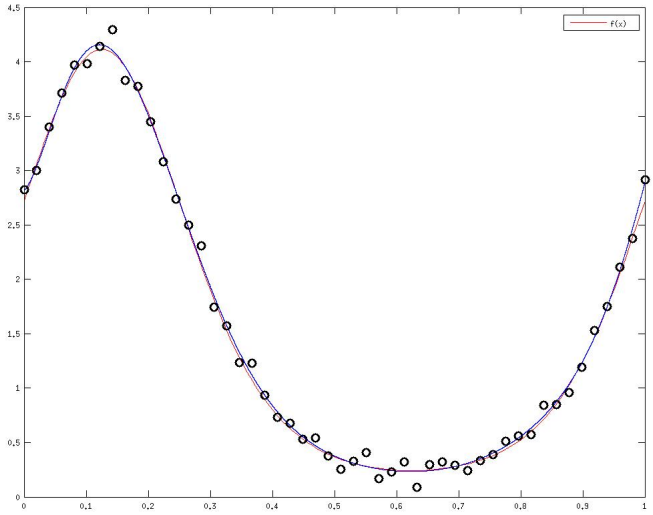


Figure 27: Output function for degree 9, training set size 50, and $\lambda = 0$.

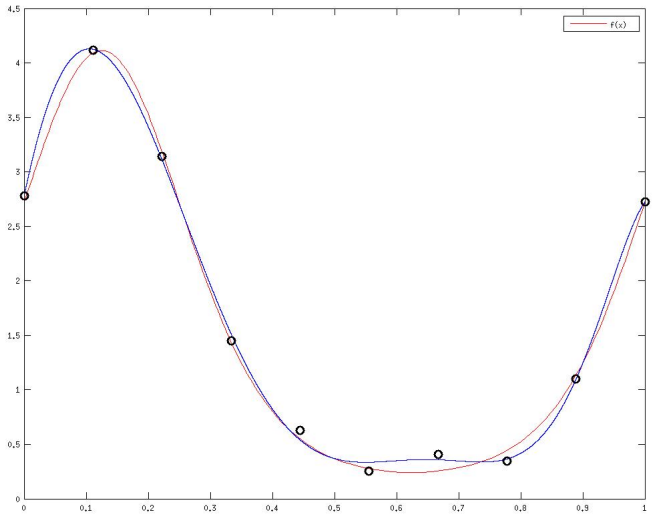


Figure 28: Output function for degree 9, training set size 10, and $\lambda = 10^{-10}$.

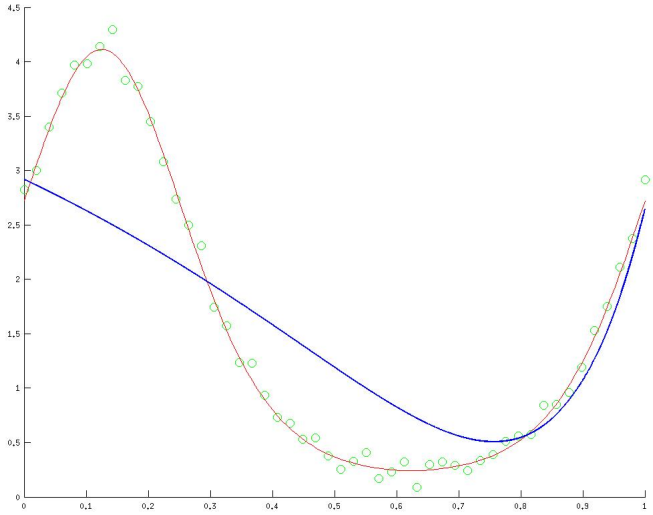


Figure 29: Output function for degree 9, training set size 50, and $\lambda = 1$.

2.1.4 Model Output and Target Output

Below are the plots for model output and the respective target output for the training dataset, validation dataset and test dataset for our best fitted model. The least test error was obtained for degree 7 and $\lambda = 10^{-15}$. The blue points are the model outputs and the red points are the target outputs

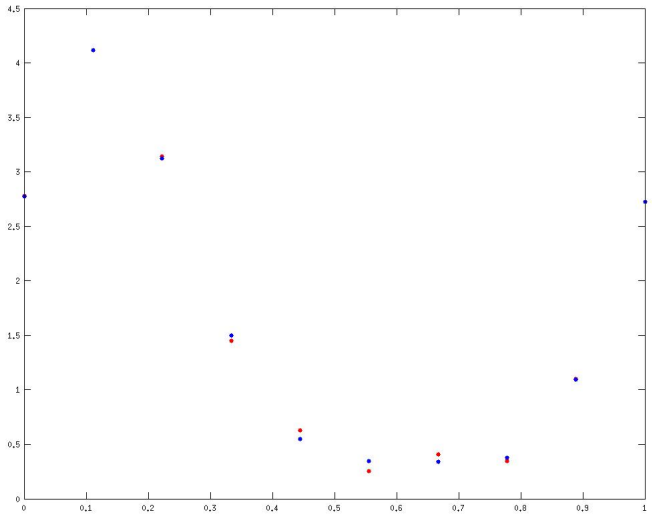


Figure 30: Model output and Target output for training set.

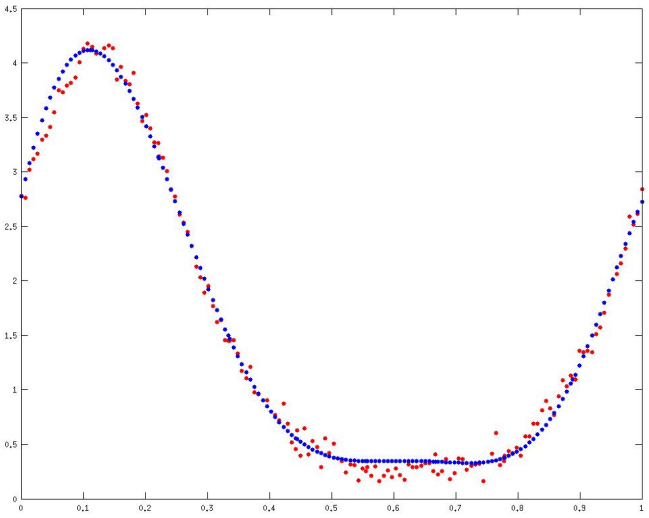


Figure 31: Model output and Target output for validation set.

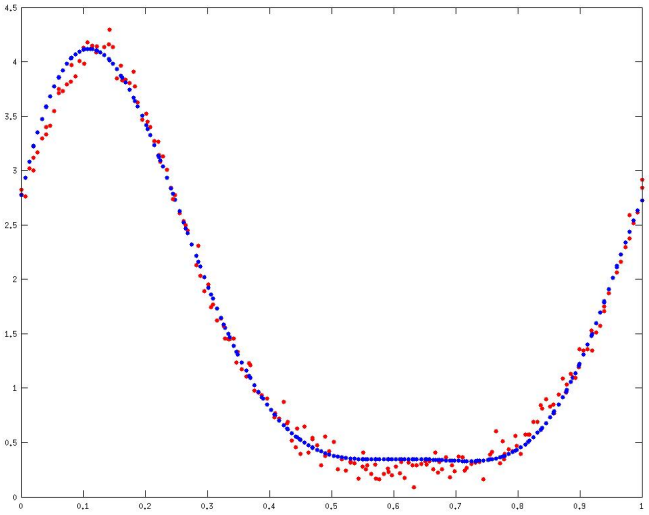


Figure 32: Model output and Target output for test set.

2.1.5 Model Output vs. Target Output

Below are the plots for model output on the y-axis vs. the respective target outputs on the x-axis for the training dataset, validation dataset and test dataset for our best fitted model (degree 7 and $\lambda = 10^{-15}$).

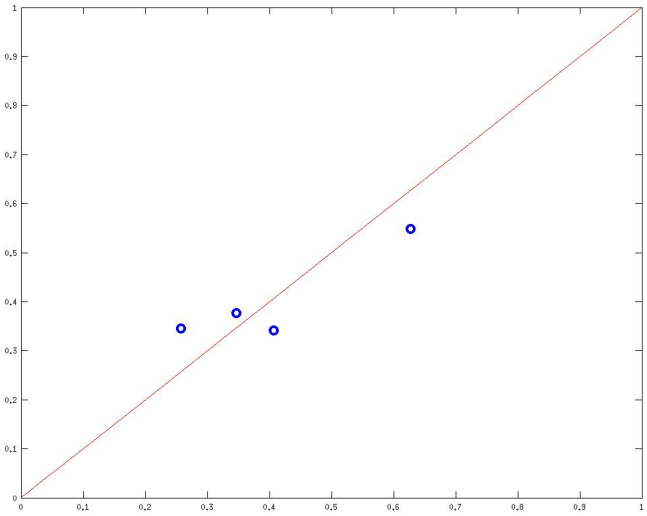


Figure 33: Model output vs. Target output for training set.

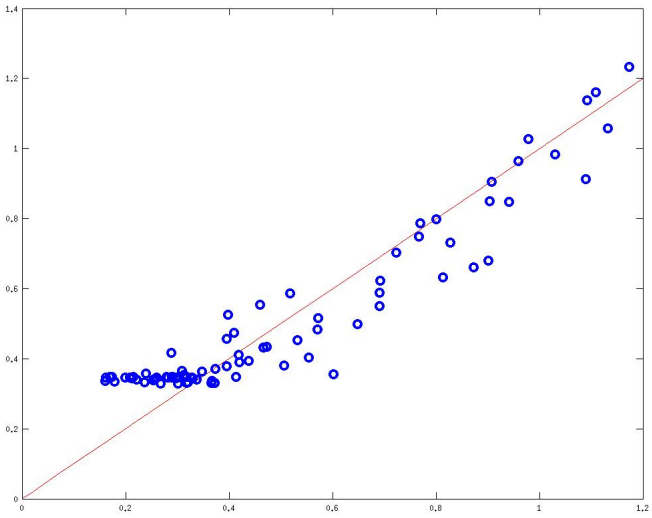


Figure 34: Model output vs. Target output for validation set.

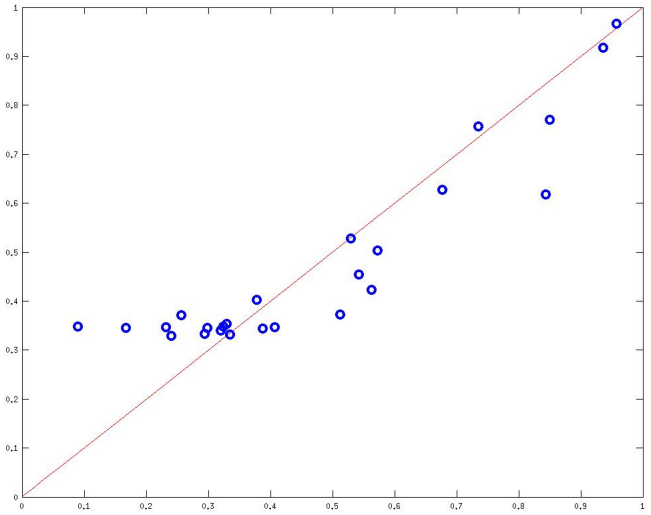


Figure 35: Model output vs. Target output for test set.

2.1.6 Observations and Inferences

We make the following observations from the above plots:

- Low degree polynomials, like 3 and 4, are unable to capture the general function underlying the training points, since they do not have enough degrees of freedom.
- Medium degree polynomials, like 5 and 7, do considerably better at approximating the underlying function than the low degree polynomials, even when trained on small datasets of size 10.
- For medium degree polynomials, like 7, when the training dataset size is increased, the polynomial becomes better at approximating the underlying function.
- For medium degree polynomials, like 7, regularization doesn't really affect the approximation obtained, since these polynomials aren't flexible enough to overfit the training data.
- High degree polynomials, like 9, when trained on size 10 datasets are able to exactly fit through all the training points. Thus they overfit, and begin to model even the noise present in the data. Hence it doesn't generalise well to the underlying function.
- High degree polynomials, like 9, when trained on larger datasets of size 50, are able to avoid overfitting, and thus give an accurate representation of the underlying function.
- When the 9 degree polynomial is trained on size 10 datasets, but with moderate regularization, it avoids overfitting and generalizes well to the underlying function.
- When the regularization parameter is increased to a very large value like 1, the polynomial obtained is almost like a straight line, and doesn't represent any of the characteristics of the training data.

2.2 Gaussian Basis Functions

2.2.1 Regression using Gaussian Basis functions

The general form of a Gaussian basis function is:

$$\varphi(x) = e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

where μ is the center, and σ is the spread of the Gaussian basis function. If we use Gaussian basis functions, along with a squared loss with L2 regularization, we obtain the following optimal weights:

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

Here, Φ is the $N \times D$ design matrix, with the n^{th} row containing the Gaussian basis terms $\varphi_1(x_n), \varphi_2(x_n), ...\varphi_D(x_n)$.
t is the vector of desired outputs, i.e., $t_n = f(x_n) + \epsilon$, where ϵ is the added Gaussian noise.
λ is the regularization hyperparameter.

2.2.2 Model Output and Target Output

Below are the plots for model output and target output for different model complexities and values of λ.

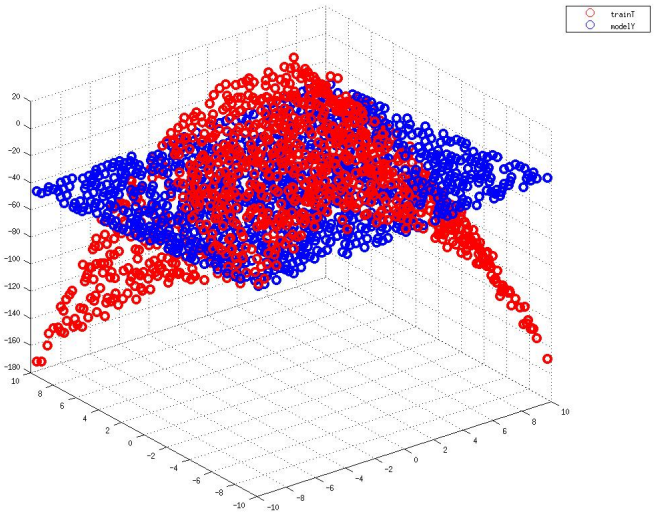


Figure 36: Model output and Target output for 3 basis functions with $\lambda = 0$.

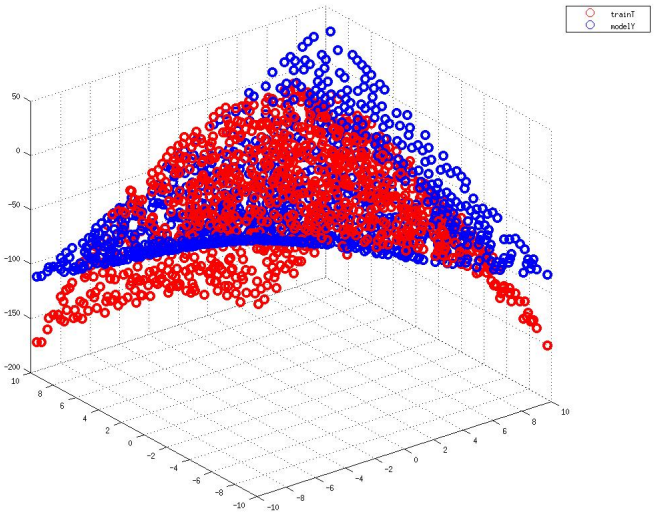


Figure 37: Model output and Target output for 4 basis functions with $\lambda = 0$.

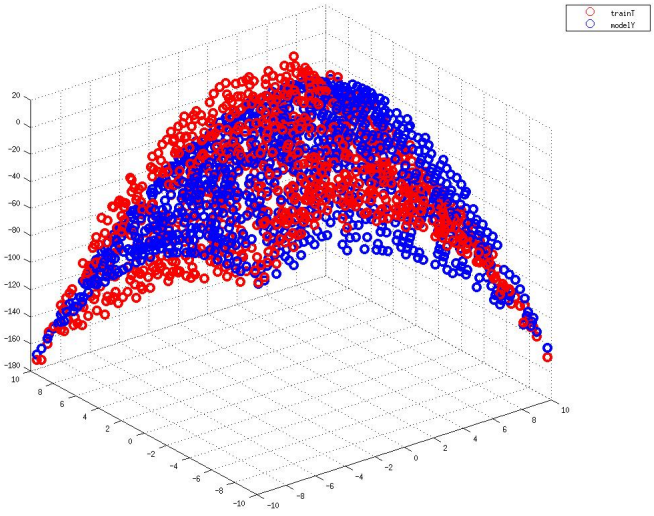


Figure 38: Model output and Target output for 5 basis functions with $\lambda = 0$.

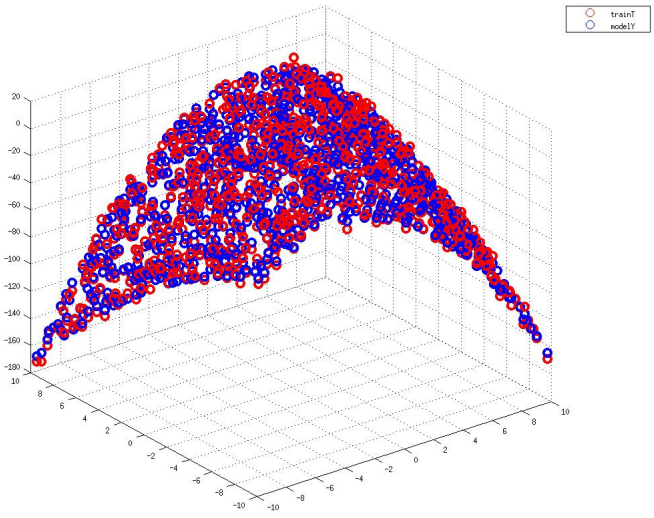


Figure 39: Model output and Target output for 6 basis functions with $\lambda = 0$.

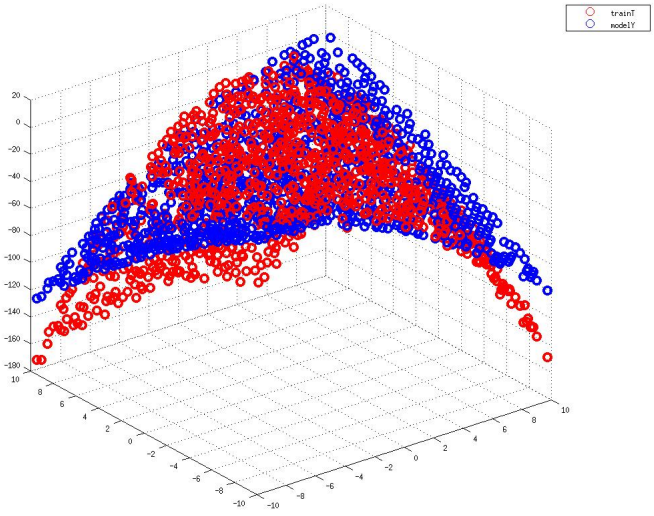


Figure 40: Model output and Target output for 6 basis functions with $\lambda = 10^{-5}$.

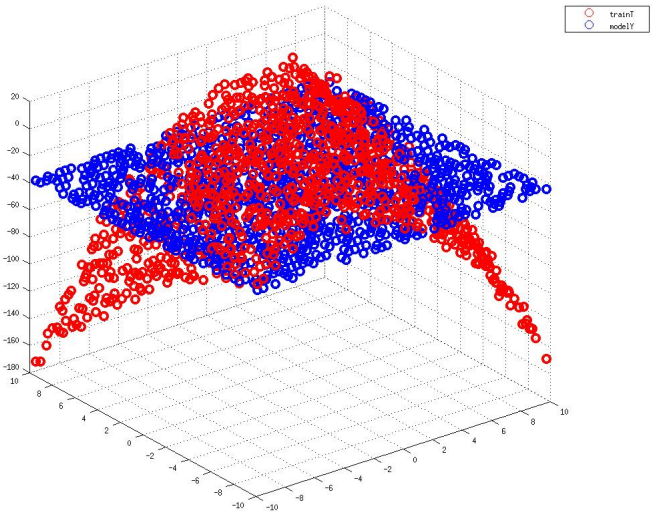


Figure 41: Model output and Target output for 9 basis functions with $\lambda = 1$.

2.2.3 Model Output and Target Output for Best Model

We found that the model using 8 GBFs, with $\lambda = 10^{-13}$ gives the best result on the validation data. Below are the plots for model output and target output for the train, test and validation data for this model.

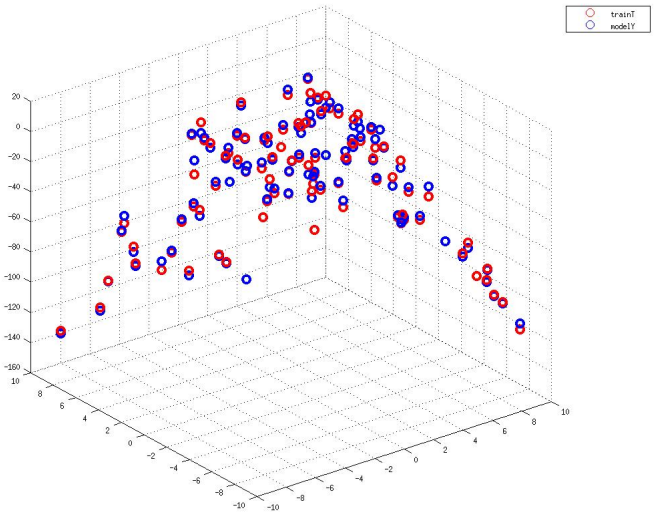


Figure 42: Model output and Target output for training data.

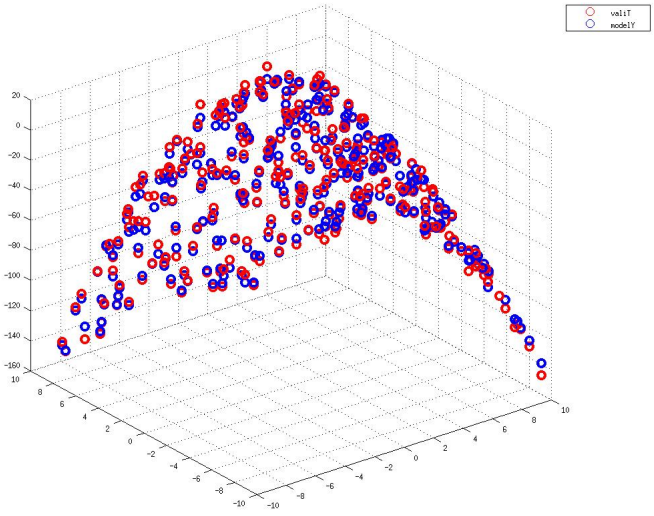


Figure 43: Model output and Target output for validation data.

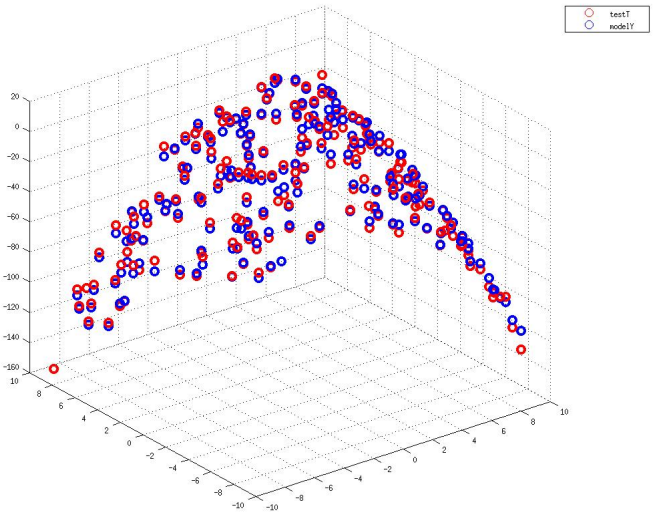


Figure 44: Model output and Target output for test data.

2.2.4 Model Output vs. Target Output

The model output was plotted on the y-axis and the target output was plotted on the x-axis for each of the train, validation, and test data for the best model. The plots are given below.

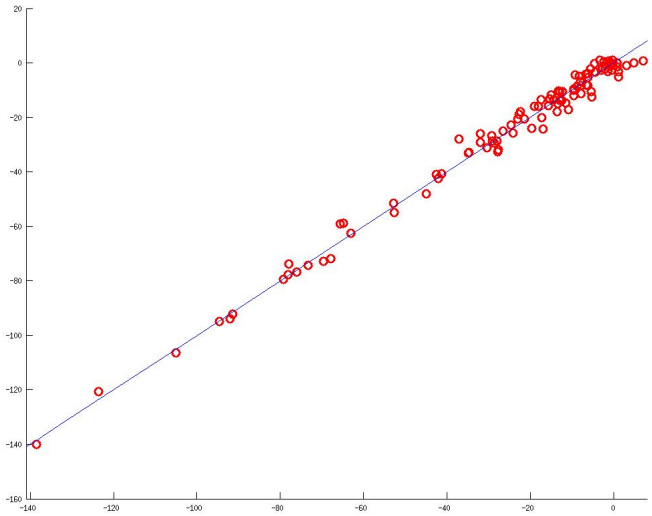


Figure 45: Model output vs Target output for training data.

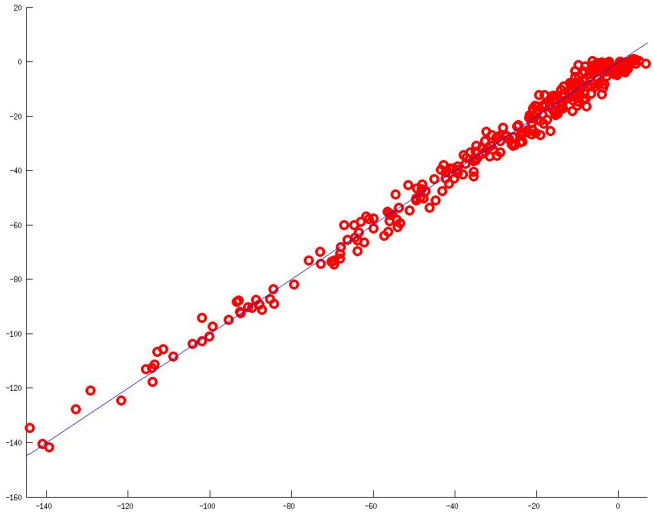


Figure 46: Model output vs Target output for validation data.

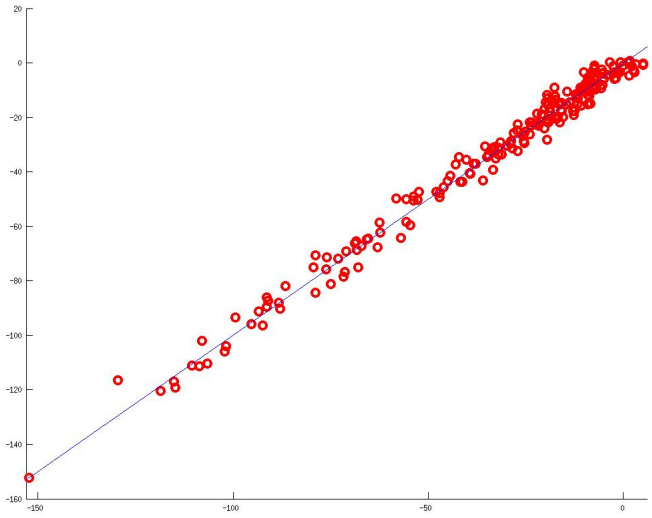


Figure 47: Model output vs Target output for test data.

2.2.5 Observations and Inferences

We draw the following observations and inferences from the above plots:

- Low complexity models, like 3 GBFs, are unable to capture the shape of the training dataset.
- Medium complexity models, like 4 and 5 GBFs, perform better than low complexity models at fitting to the training dataset. The fit is still not very good.
- High complexity models, like 6 GBFs, almost fit perfectly to the training set. The model might be overfitting in order to get such a close approximation of the training dataset.
- When the high complexity models are regularized, their fit on the training dataset deteriorates, since regularization prevents overfitting. The effect of regularization is not seen evidently on low complexity models, since they do not overfit to begin with.
- Extremely large values of λ, like 1, lead to almost invariant models due to heavy penalization of weights.

2.3 MLFFNN for Univariate Data

2.3.1 Neural Network Model

The neural network model used had 1 hidden layer, with 3 hidden nodes. The output node was linear, and the hidden nodes used the sigmoid tanh activation function. The network was trained on a 150 point training set.

2.3.2 Model Output and Target Output

Given below are the plots for the model output and target output for the training, validation, and test datasets.

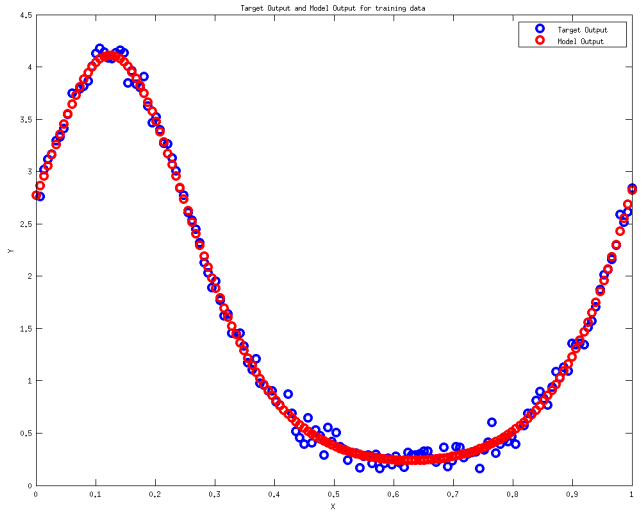


Figure 48: Model output and Target output for training data.

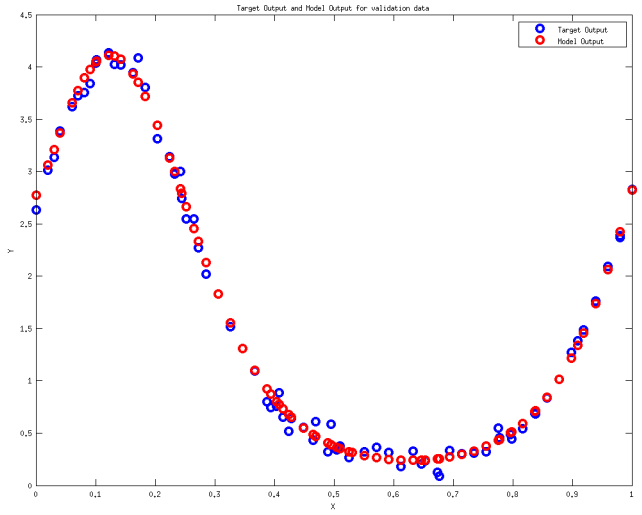


Figure 49: Model output and Target output for validation data.

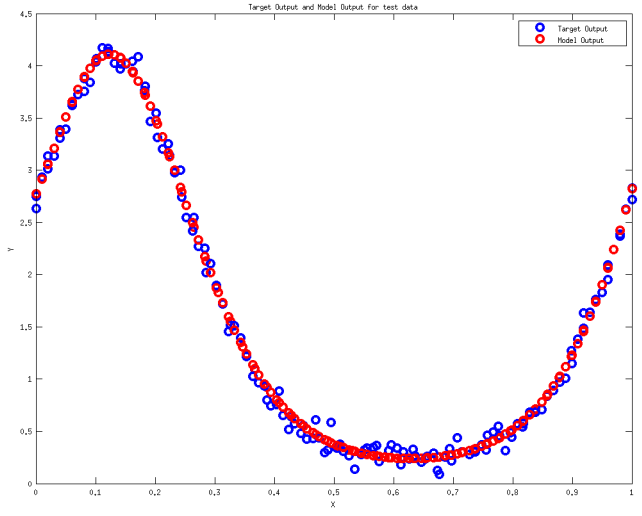


Figure 50: Model output and Target output for test data.

2.3.3 Model Output vs. Target Output

The target output was plotted on the x-axis and the model output was plotted on the y-axis for the training, validation, and the test data. The plots are shown below.

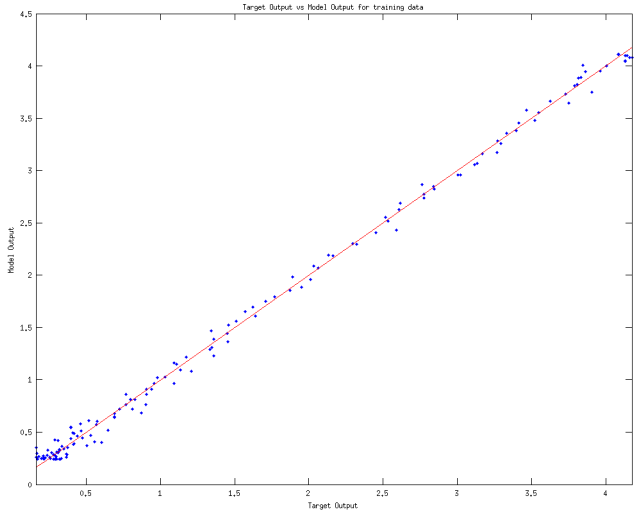


Figure 51: Model output vs. Target output for training data.

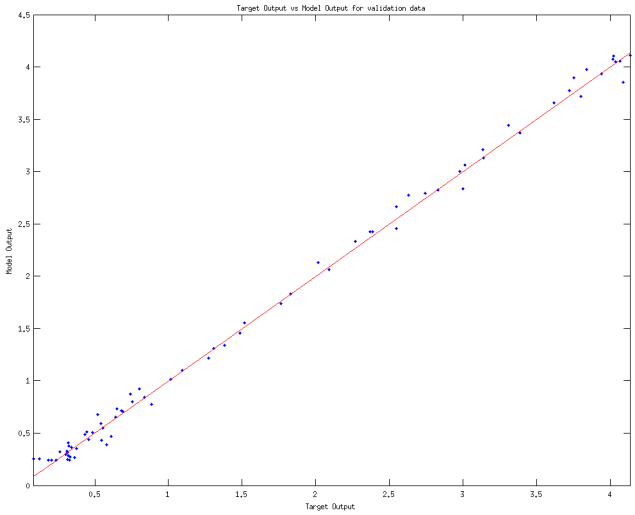


Figure 52: Model output vs. Target output for validation data.

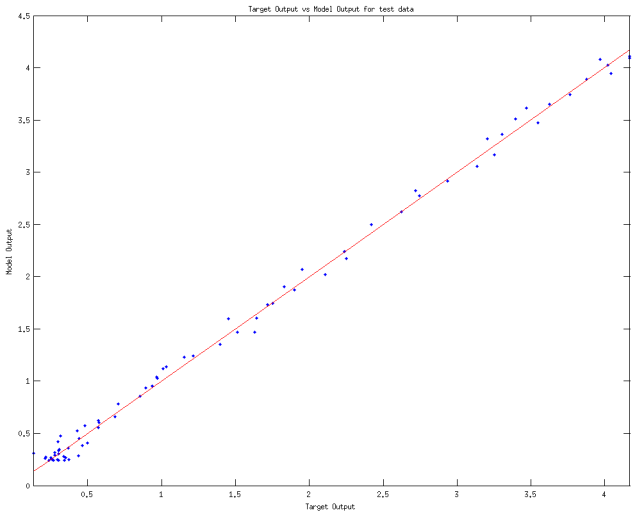


Figure 53: Model output vs. Target output for test data.

2.3.4 Observations and Inferences

From the plots it's clear that the single hidden layer neural network performs exceedingly well at approximating the underlying univariate function. This is expected, since the univariate data is simple enough to be approximated by low degree polynomials, and hence a powerful model like a neural network would be able to extract the underlying function easily from the training data.

2.4 MLFFNN for Bivariate Data

2.4.1 Neural Network Model

The neural network model used had 2 hidden layers, with 5 and 3 hidden nodes in the first and second hidden layers respectively. The output node was linear, and the hidden nodes used the sigmoid tanh activation function. The network was trained on the 2000 point training set.

2.4.2 Model Output and Target Output

Given below are the plots for the model output and target output for the training, validation, and test datasets.

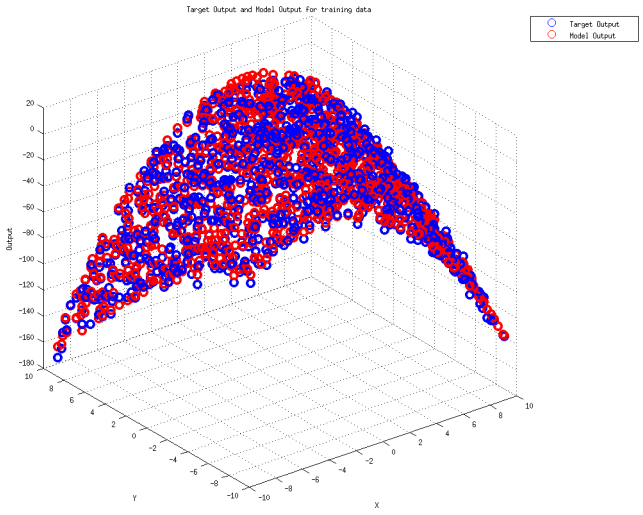


Figure 54: Model output and Target output for training data.

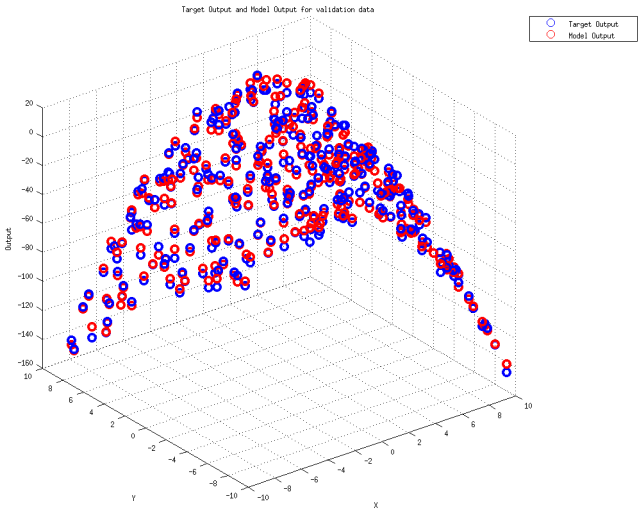


Figure 55: Model output and Target output for validation data.

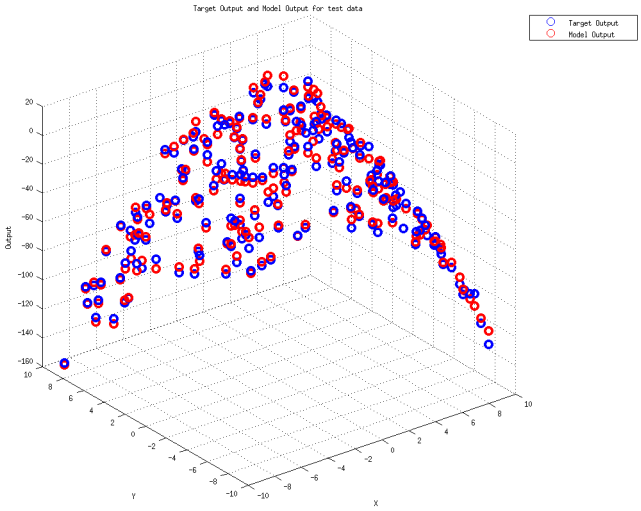


Figure 56: Model output and Target output for test data.

2.4.3 Model Output vs. Target Output

The target output was plotted on the x-axis and the model output was plotted on the y-axis for the training, validation, and the test data. The plots are shown below.

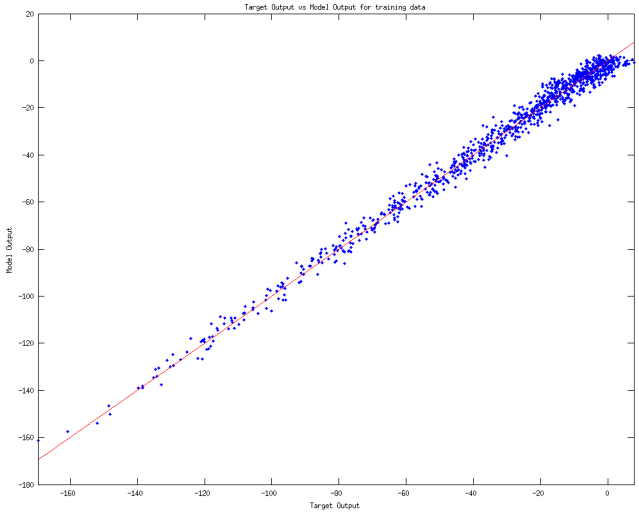


Figure 57: Model output vs. Target output for training data.

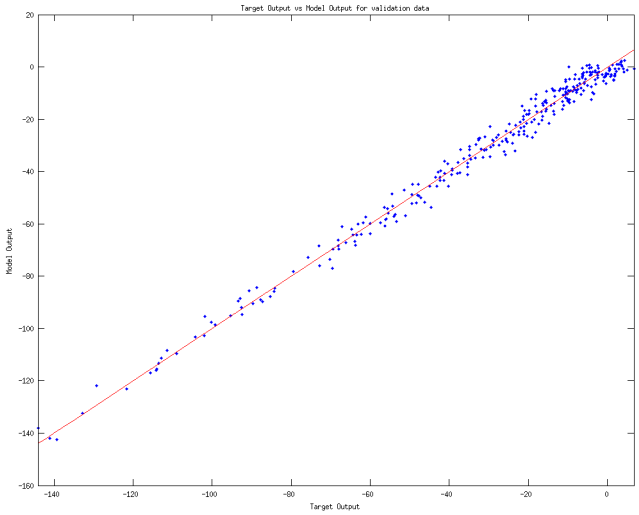


Figure 58: Model output vs. Target output for validation data.

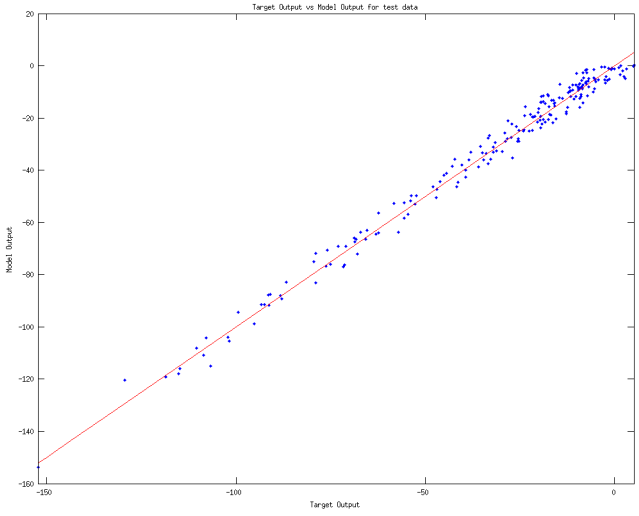
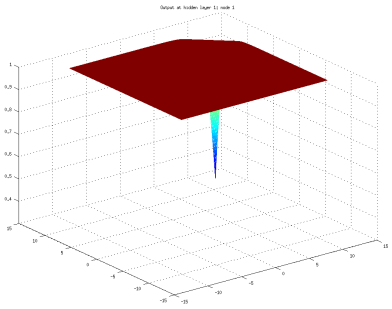


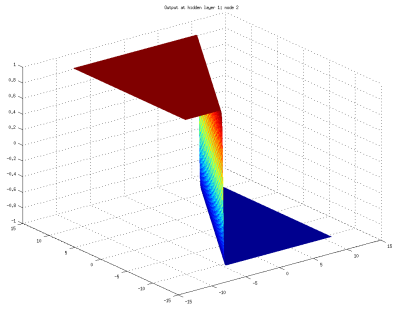
Figure 59: Model output vs. Target output for test data.

2.4.4 Output Plots for Hidden & Output Nodes

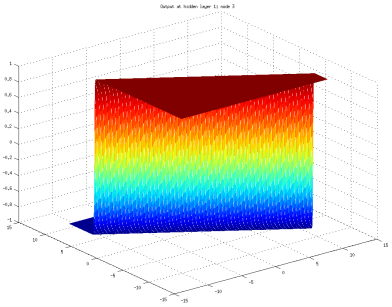
Epoch 1:



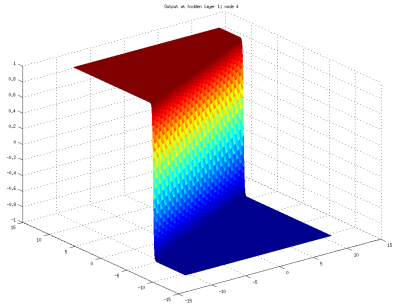
(a) Output plot for Node 1



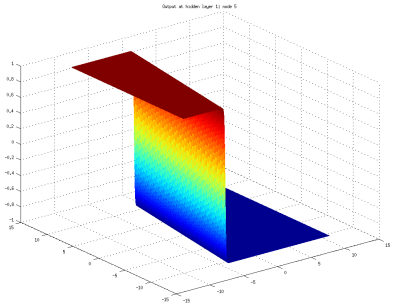
(b) Output plot for Node 2



(c) Output plot for Node 3

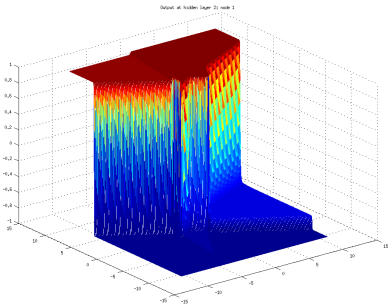


(d) Output plot for Node 4

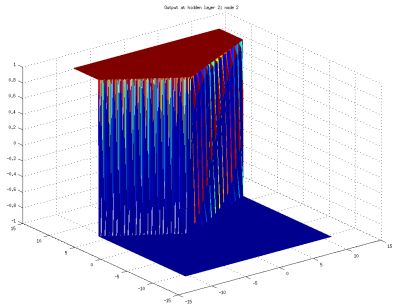


(e) Output plot for Node 5

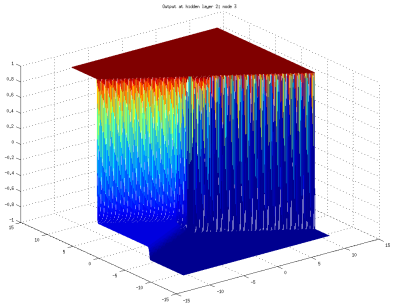
Figure 60: Output plots for hidden layer 1.



(a) Output plot for Node 1



(b) Output plot for Node 2



(c) Output plot for Node 3

Figure 61: Output plots for hidden layer 2.

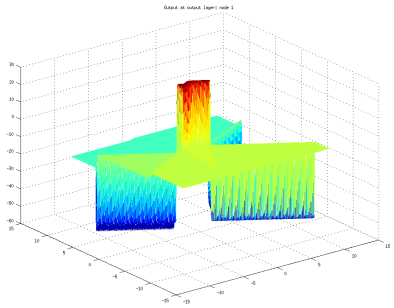
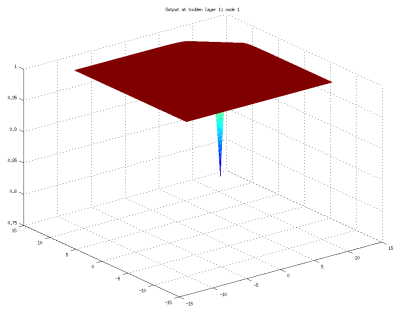
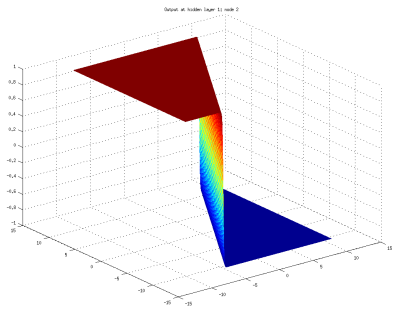


Figure 62: Output plot for output node

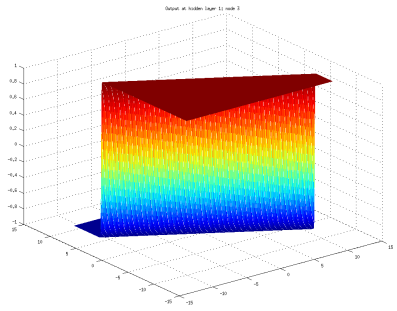
Epoch 2:



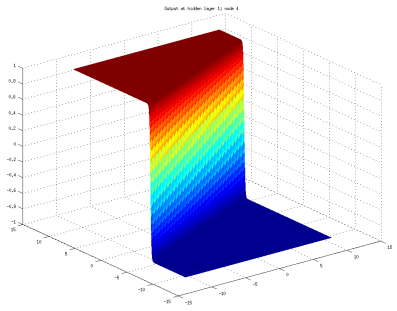
(a) Output plot for Node 1



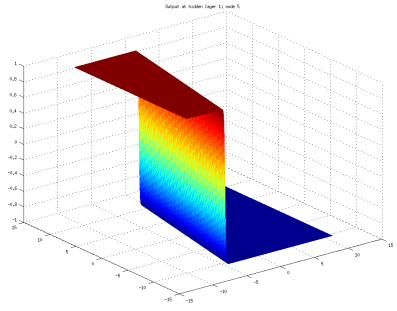
(b) Output plot for Node 2



(c) Output plot for Node 3

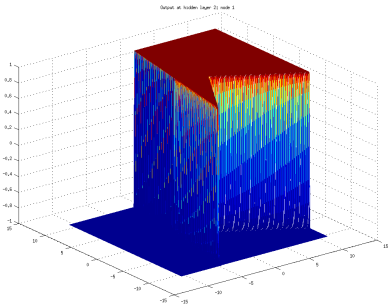


(d) Output plot for Node 4

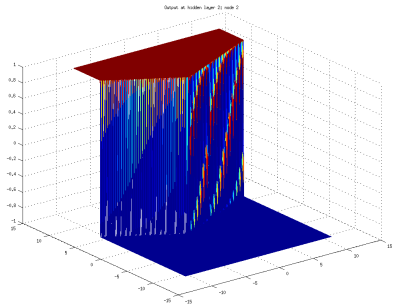


(e) Output plot for Node 5

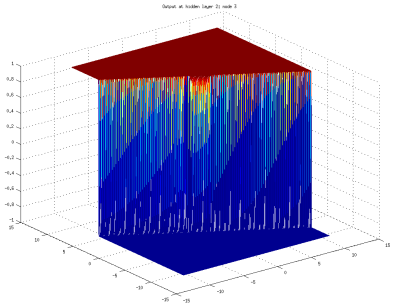
Figure 63: Output plots for hidden layer 1.



(a) Output plot for Node 1



(b) Output plot for Node 2



(c) Output plot for Node 3

Figure 64: Output plots for hidden layer 2.

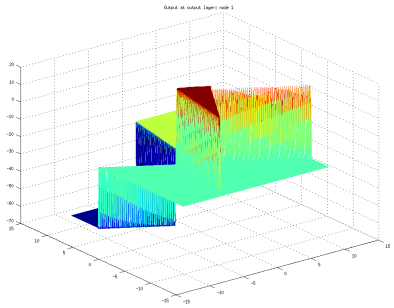
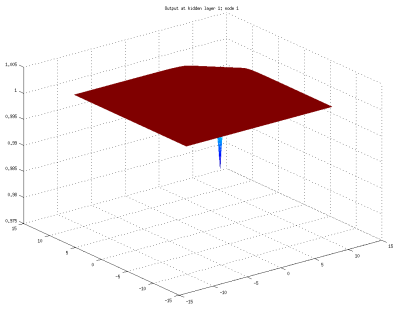
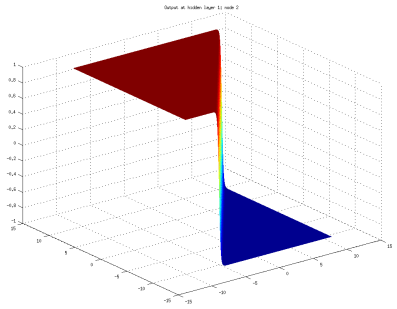


Figure 65: Output plot for output node

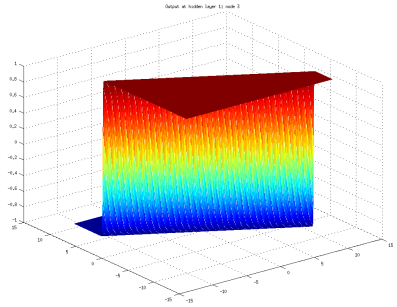
Epoch 10:



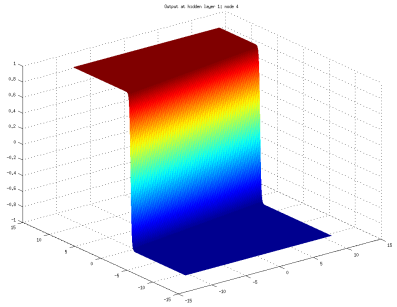
(a) Output plot for Node 1



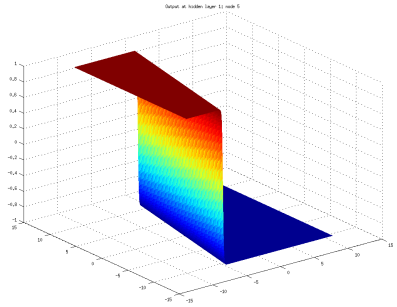
(b) Output plot for Node 2



(c) Output plot for Node 3

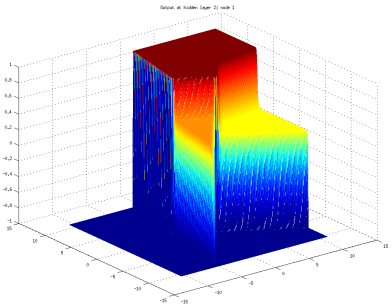


(d) Output plot for Node 4

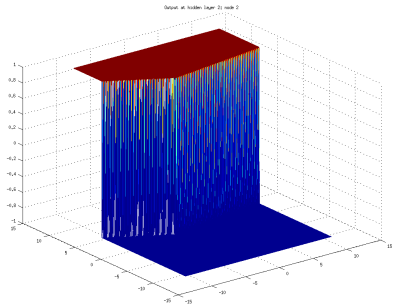


(e) Output plot for Node 5

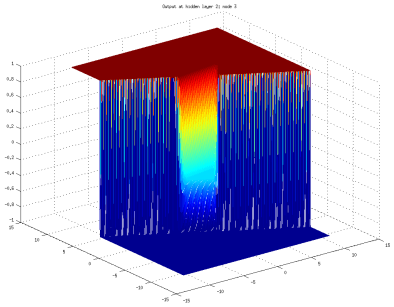
Figure 66: Output plots for hidden layer 1.



(a) Output plot for Node 1



(b) Output plot for Node 2



(c) Output plot for Node 3

Figure 67: Output plots for hidden layer 2.

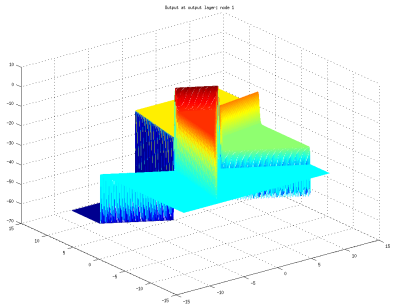
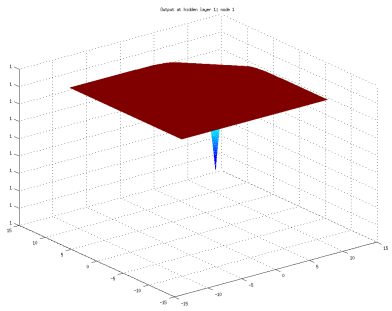
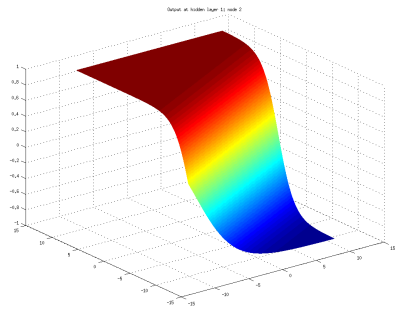


Figure 68: Output plot for output node

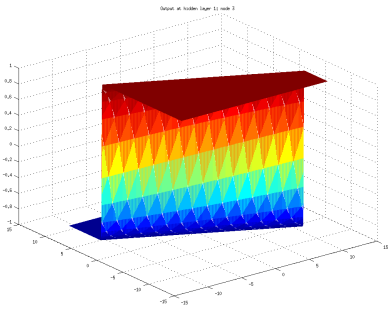
Epoch 50:



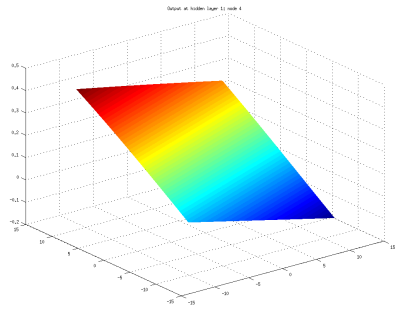
(a) Output plot for Node 1



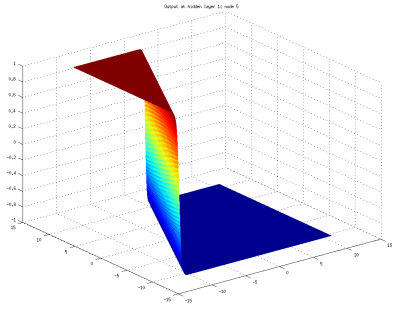
(b) Output plot for Node 2



(c) Output plot for Node 3

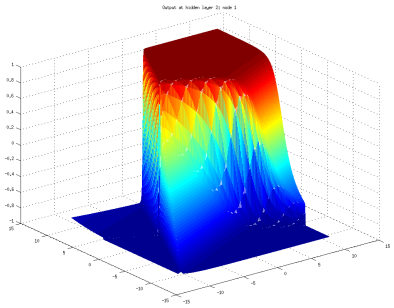


(d) Output plot for Node 4

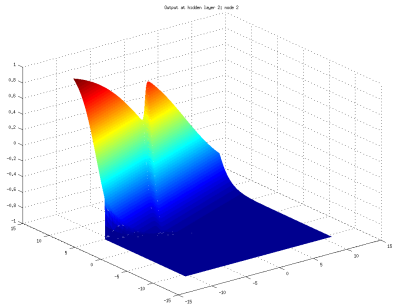


(e) Output plot for Node 5

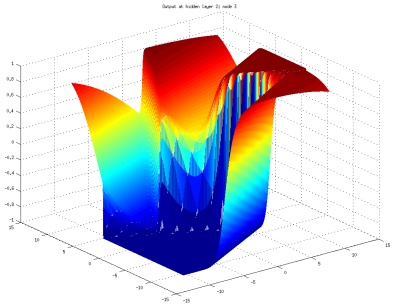
Figure 69: Output plots for hidden layer 1.



(a) Output plot for Node 1



(b) Output plot for Node 2



(c) Output plot for Node 3

Figure 70: Output plots for hidden layer 2.

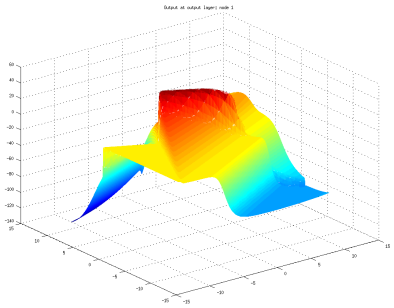
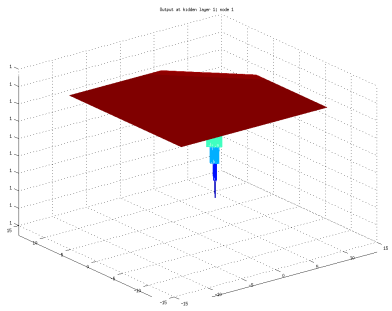
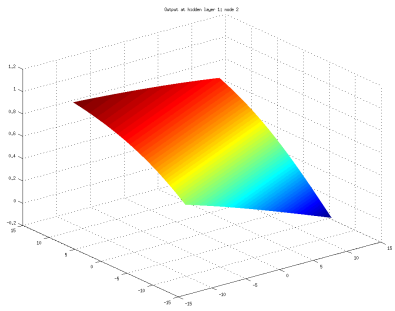


Figure 71: Output plot for output node

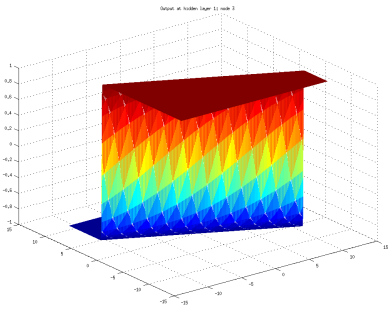
Epoch 100:



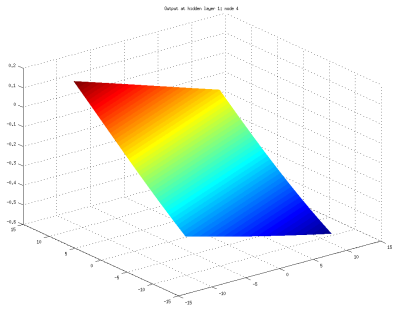
(a) Output plot for Node 1



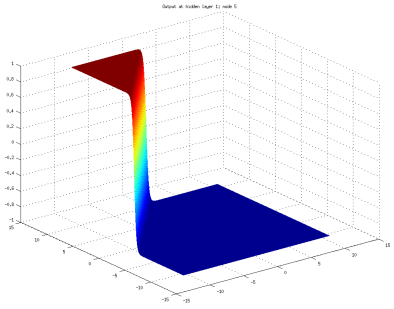
(b) Output plot for Node 2



(c) Output plot for Node 3

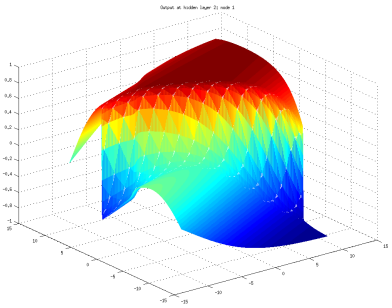


(d) Output plot for Node 4

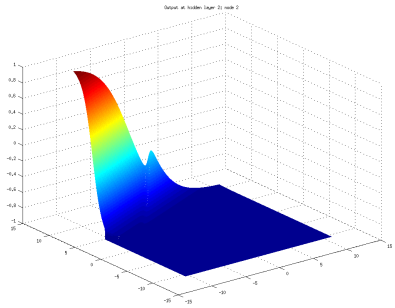


(e) Output plot for Node 5

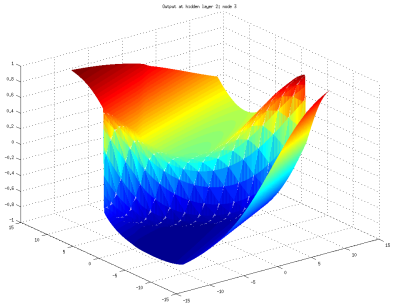
Figure 72: Output plots for hidden layer 1.



(a) Output plot for Node 1



(b) Output plot for Node 2



(c) Output plot for Node 3

Figure 73: Output plots for hidden layer 2.

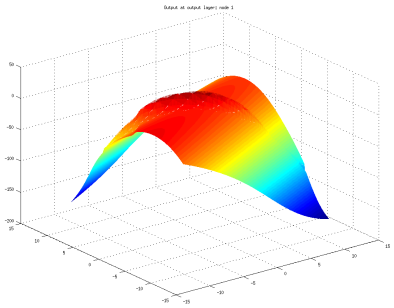
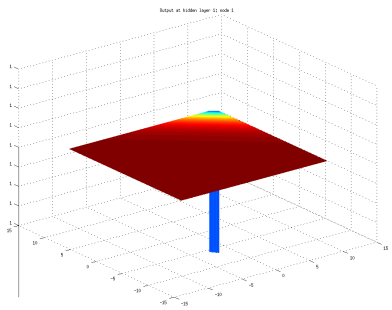
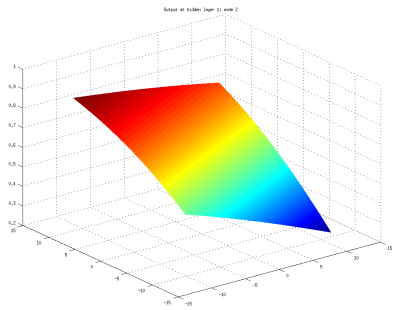


Figure 74: Output plot for output node

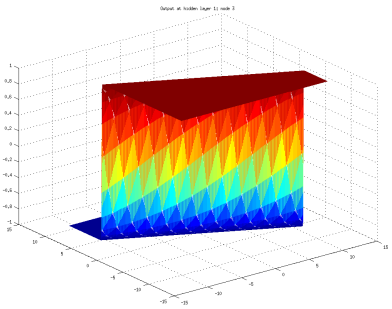
Post Training:



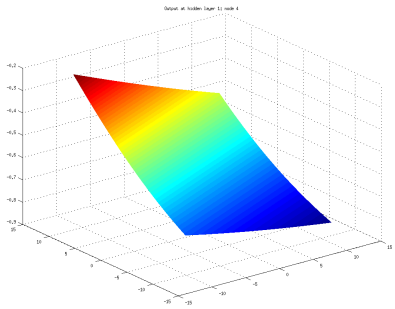
(a) Output plot for Node 1



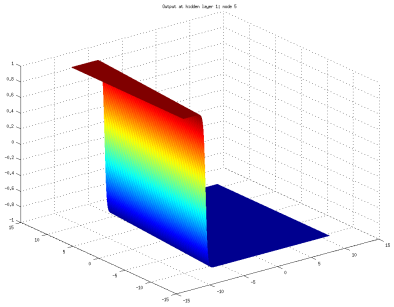
(b) Output plot for Node 2



(c) Output plot for Node 3

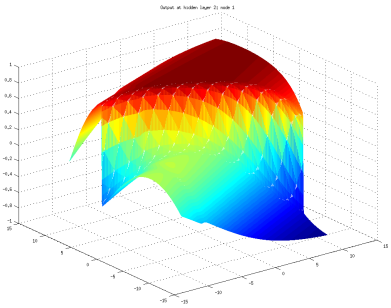


(d) Output plot for Node 4

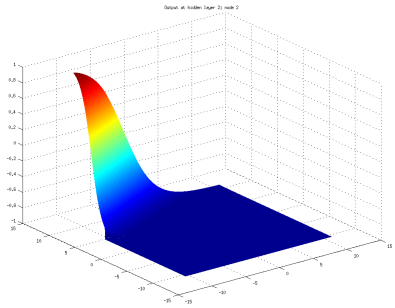


(e) Output plot for Node 5

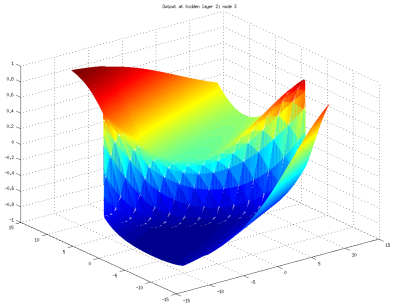
Figure 75: Output plots for hidden layer 1.



(a) Output plot for Node 1



(b) Output plot for Node 2



(c) Output plot for Node 3

Figure 76: Output plots for hidden layer 2.

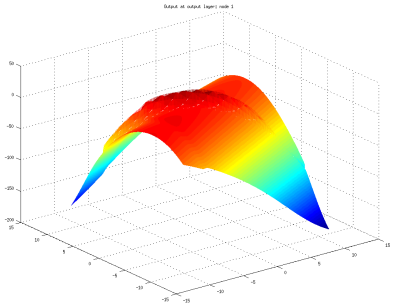


Figure 77: Output plot for output node

2.4.5 Observations and Inferences

We draw the following observations and inferences from the above plots:

- The neural network performs very well on the bivariate data.

- If we observe the training phase through the plots of the outputs of the hidden nodes and the output node, with some effort, we can see how the output node is slowly modified as the training progresses to fit the underlying function of the bivariate data.

2.5 Generalized Radial Basis Functions (bivariate)

2.5.1 Regression using Generalized RBFs

If we use Generalized RBFs for regression, we obtain the following optimal weights:

$$w^* = (\Phi^T\Phi + \lambda\widetilde{\Phi})^{-1}\Phi^T t$$

Here, Φ is the $N \times D$ design matrix, with the n^{th} row containing the Gaussian basis terms $\varphi_1(x_n), \varphi_2(x_n), ...\varphi_D(x_n)$.
t is the vector of desired outputs, i.e., $t_n = f(x_n) + \epsilon$, where ϵ is the added Gaussian noise.
λ is the regularization hyperparameter. $\widetilde{\Phi}_{ij} = e^{\frac{-(\mu_i - \mu_j)^2}{2\sigma^2}}$ Here $\sigma = \frac{\alpha}{2m}$, where $\alpha = max_{i,j}||\mu_i - \mu_j||$

2.5.2 Model Output and Target Output

The plots of model output and target output were created for the best model obtained. The best model was achieved for 6 RBFs, with $\lambda = 10^{-11}$, trained on 1000 data points. The plots are given below:

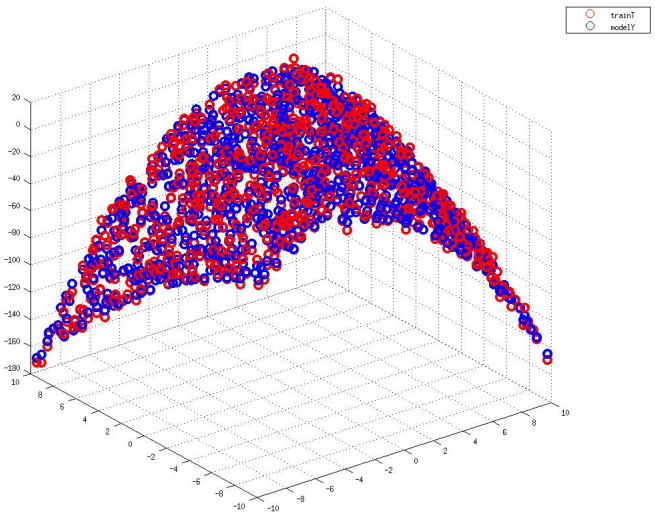


Figure 78: Model output and Target output for training data.

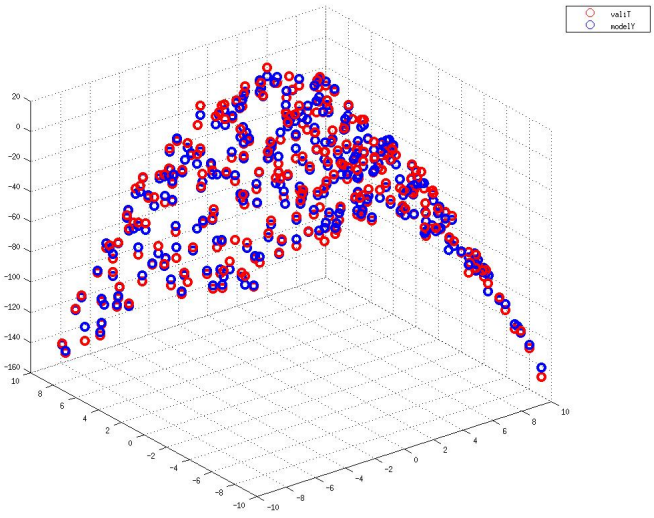


Figure 79: Model output and Target output for validation data.

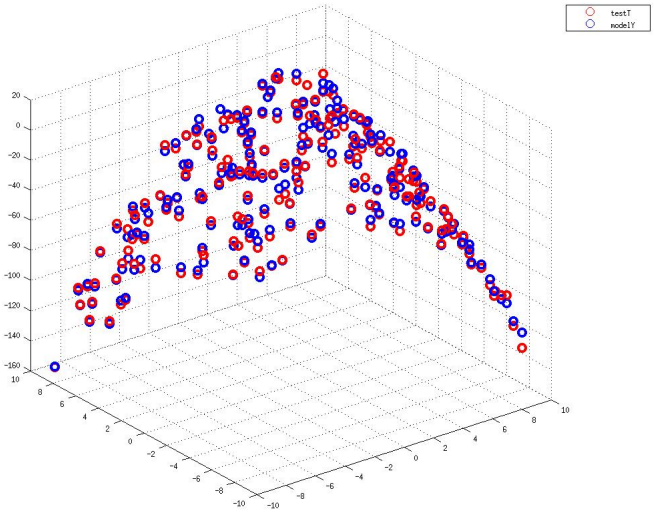


Figure 80: Model output and Target output for test data.

2.5.3 Model Output vs. Target Output

The model output was plotted on the y-axis and the target output was plotted on the x-axis for each of the train, validation, and test data for the best model. The plots are given below:

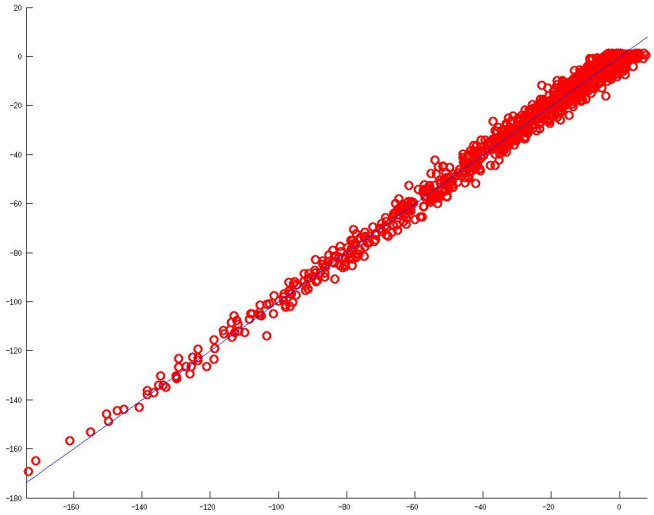


Figure 81: Model output vs Target output for train data.

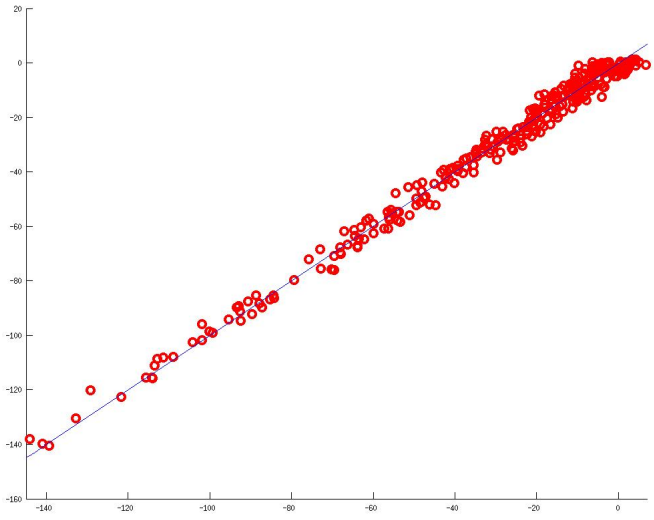


Figure 82: Model output vs Target output for validation data.

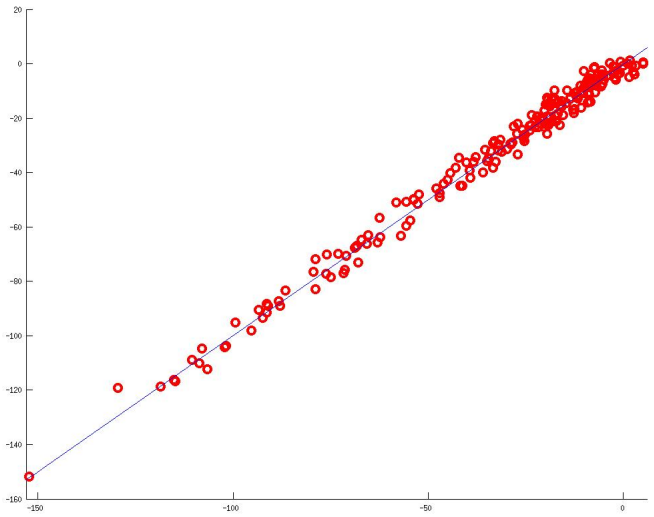


Figure 83: Model output vs Target output for test data.

2.6 Generalized Radial Basis Functions (univariate)

2.6.1 Model Output and Target Output

The plots of model output and target output were created for the best model obtained. The best model was achieved for 30 RBFs, with $\lambda = 0$, trained on 50 data points. The plots are given below:

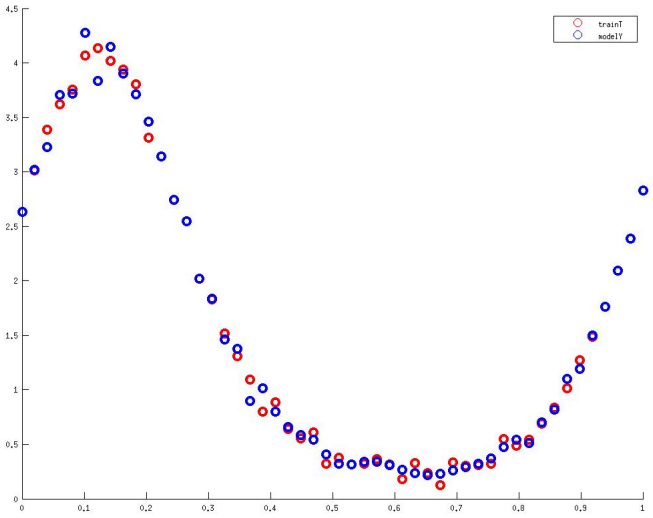


Figure 84: Model output and Target output for training data.

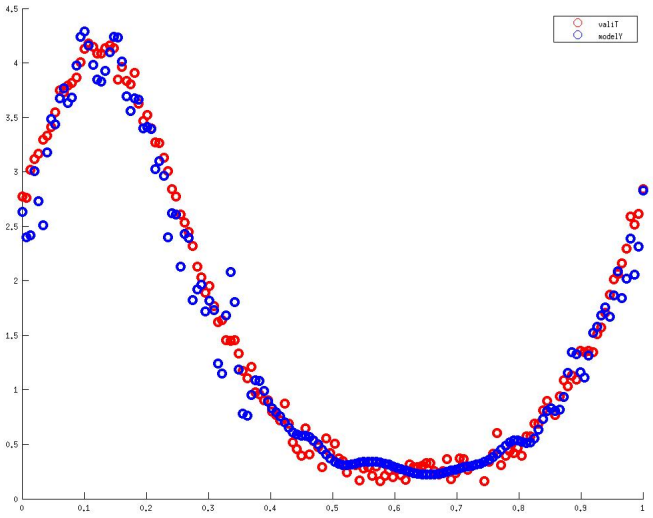


Figure 85: Model output and Target output for validation data.

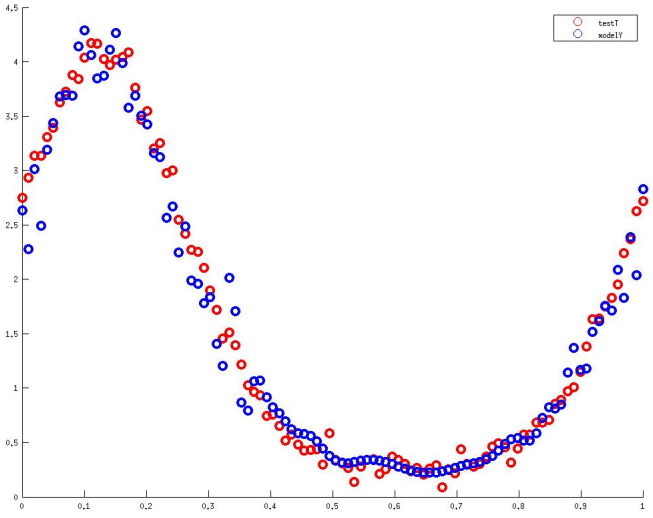


Figure 86: Model output and Target output for test data.

2.6.2 Model Output vs. Target Output

The model output was plotted on the y-axis and the target output was plotted on the x-axis for each of the train, validation, and test data for the best model. The plots are given below:

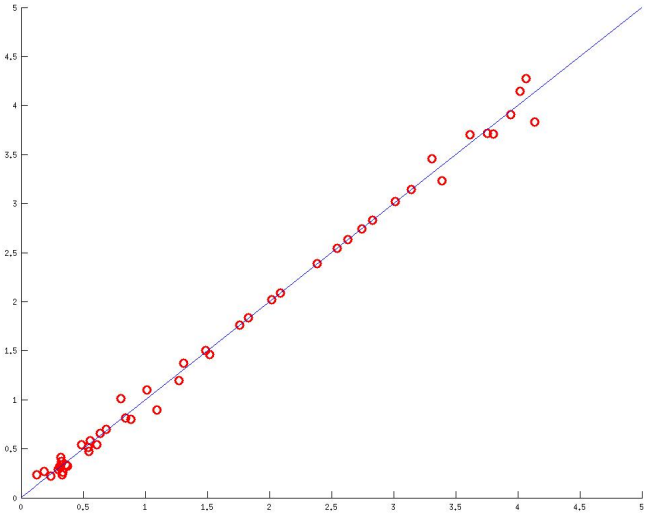


Figure 87: Model output vs Target output for train data.

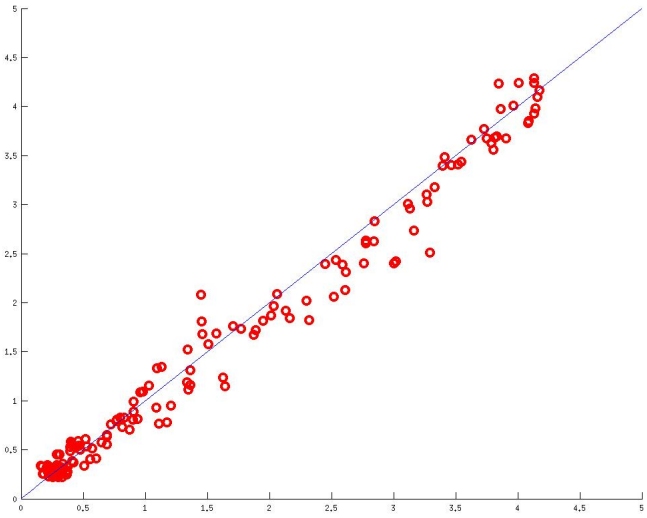


Figure 88: Model output vs Target output for validation data.

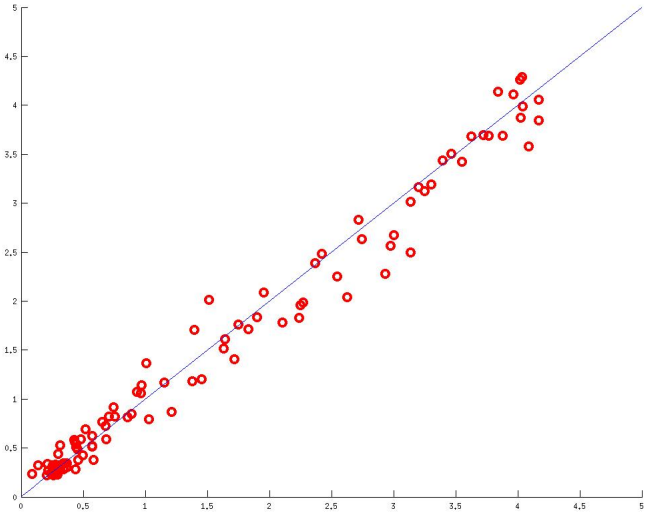


Figure 89: Model output vs Target output for test data.

2.6.3 Observations & Inferences

The generalized RBF gives very similar results when compared to the GBF model. The advantage produced by the Tikhonov regularization doesn't seem to be pronounced for this dataset.