CS671: Deep Learning and Applications Programming Assignment-2

Rohit Kumar
SCENE, IIT Mandi
D22078
D22078@students.iitmandi.ac.in

Nikhil Mahar
SCENE, IIT Mandi
D22123
D22123@students.iitmandi.ac.in

Sudhir Kumar Choudhary
SMME, IIT Mandi
(Project)

Abstract

Neural networks are known as universal approximators. Unlike previous report, where only single neuron was used for classification and regression, in this report we adopted multiple neurons to classify multiple classes and also to fit the univariate and bivariate data. A single and double-layered neural network is developed from scratch. Cross-entropy loss function is used to classify 3 classes and mean squared loss function is used to fit univariate and bivariate data. It was observed that multiple neurons in either single or multiple layers performed well.

1 Introduction

A multilayer perceptron (MLP) is a type of artificial neural network that is commonly used for classification and regression tasks. It is composed of multiple layers of interconnected processing units called neurons, with each layer being fully connected to the previous and following layers.

The input layer receives the input data, and each neuron in the input layer corresponds to one input feature. The output layer produces the final output, which could be a classification label or a predicted value. The hidden layers are intermediate layers between the input and output layers, and they perform transformations on the input data to extract relevant features.

Each neuron in the MLP is a mathematical function that takes a weighted sum of its inputs, applies an activation function to the sum, and produces an output. The weights and biases of the neurons are learned during the training process using backpropagation, which is an iterative algorithm that adjusts the weights to minimize the error between the predicted output and the actual output.

MLPs are powerful and flexible models that can learn complex nonlinear relationships in data. They have been successfully applied to a wide range of problems, including image classification, natural language processing, and speech recognition. However, they can be computationally expensive to train and require a large amount of labeled data. The learning rule can be summarized as follows:

1. Initialize the weights and bias terms to small random values.
2. For each training example, compute the weighted sum of the inputs.
3. Apply the activation function to the weighted sum.
4. Update the weights and bias term if the predicted output is not equal to the true output.
5. Repeat steps 2-4 for a fixed number of epochs or until the model converges.

This report is broadly divided into two parts: (1) Classification, (2) Regression to measure the capabilities of MLP.

2 Classification

2.1 Classification of linearly separable datasets

In this section a MLP is used to classify the linearly separable and non-linearly separable multi-classes. Schematic representation of single hidden layer neural network is shown in Figure 1 Three

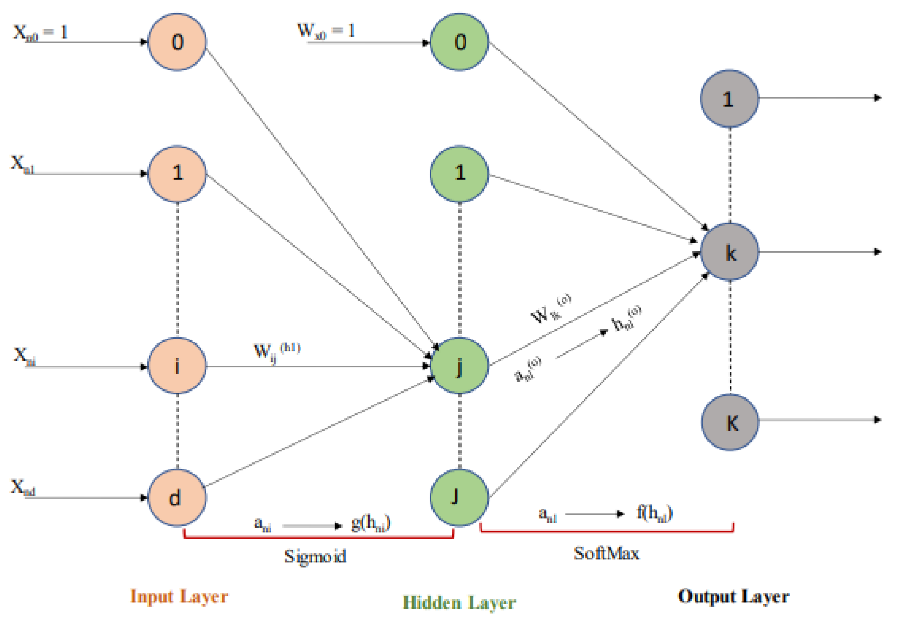


Figure 1: Neural Network with single hidden layer

types of dataset are provided which are linearly seprable. Each dataset is further split into 60% traning, 20% validation, and 20% test samples as shown in Table 1. Three classes of data can be seen in Figure 2.

	Total Samples	Training Samples	Validation Samples	Test Samples
Class 1	500	300	100	100
Class 2	500	300	100	100
Class 3	500	300	100	100

Table 1: Data division for training, testing and validation

A neural network with one hidden layer, one input layer and one output layer is used to classify the 3 classes. The learning rate used is 0.010. In between input and hidden layer, logistic activation function is used, while softmax activation function is used in between hidden and output layer. The number of neurons in input layer are 2, which are equivalent to number of features, the number of neurons in output layer are 3. One hot encoding is used to classfiy 3 classes in output layer. For hidden layer cross validation is done by changing number of neurons. As can be seen from table 2, network with 3 hidden nodes performed well. The validation error starts to increase after 3 neurons, which signifies overfitting of data. Cross entropy loss function with backpropagation algorithm utilizing gradient descent method is used to update weights and minimize errors. The algorithm used to train the network is as follows:

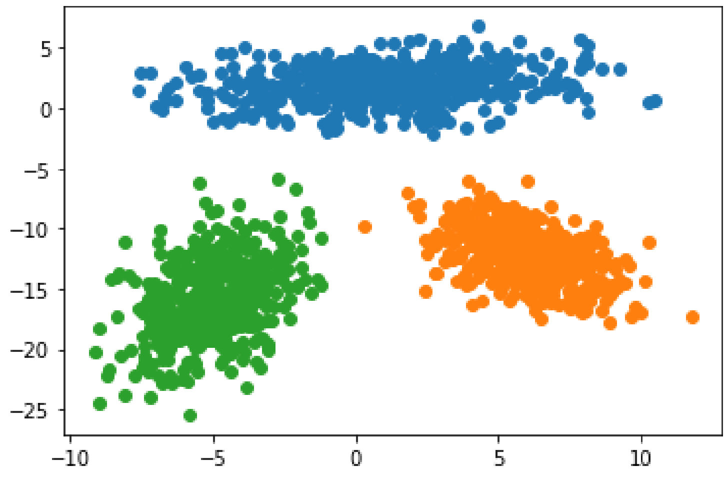


Figure 2: Datasets belonging to 3 classes, blue, orange, green as class 1, class2 and class3 respectively

1. Given-training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathrm{R}^d$, and $\mathbf{y}_n \in \mathrm{R}^K$, Initialize the W_{ij} and W_{jk} with random values.
2. Randomly choose a training example x_n
3. Compute output of all output neurons $(k = 1, 2, \dots K)$: s_{nk}
4. Compute the instantaneous error: $E_n = -log(s_{nk})$
5. Update weights between hidden and output layer $(j = 1, 2, \dots J)$ and $(k = 1, 2, \dots K)$:
   - $\Delta w_{jk} = -\eta \frac{\partial E_n}{\partial w_{jk}} = \eta \delta_{nk} h_{nj} = \eta(1 - s_{nk}) h_{nj}$
   - $w_{jk} = w_{jk} + \Delta w_{jk}$
6. Update weights between hidden and input layer $(i = 1, 2, \dots d)$ and $(j = 1, 2, \dots J)$:
   - $\Delta w_{ij} = -\eta \frac{\partial E_n}{\partial w_{ij}} = \eta \delta_{nj} h_{ni} = \eta(1 - s_{nk}) w_{jk} g(a_{nj})(1 - g(a_{nj})) x_{ni}$
   - $w_{ij} = w_{ij} + \Delta w_{ij}$
7. Repeat steps 2 to 6 till all the training examples are presented once (Epoch)
8. Compute the average error: $E_{avg} = \frac{1}{N} \sum E_n$
9. Repeat steps 2 to 8 till the convergence criterion is satisfied

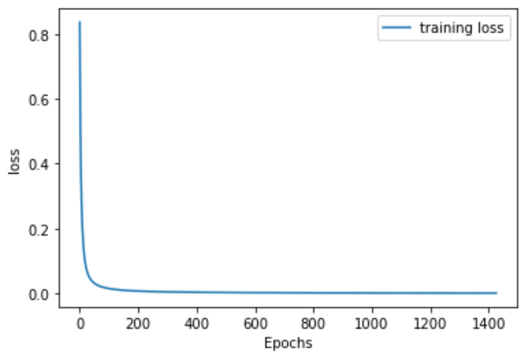
Confusion matrix of test data for best network is given in table **??**, it can be observed that accuracy is 100%. Complete derivation of Δw_{ij}, Δw_{jk} is available in appendix. Figure 3 depicts the variation of MSE loss with epochs for training and validation samples of best architecture. Stopping criteria was set as to either 10000 epochs are reached or $MSE < 10^{-3}$. Figure 4 shows the output of individual neurons on training, validation, and testing data. It can be observed that individual neuron is trying to learn pieces of function from the dataset. Figure 5 represents the output of each neuron in output layer, it can be observed that neurons are able to classify difference between 3 classes, as individual neuron is acting as binary classifier, which classify one class from other two classes. On plotting decision region plot superimposed by training data, and also observing confusion matrix of test samples, it can be observed from Figure 6(a) that neural network classified all the 3 classes with 100% accuracy. On comparing with single linear perceptron output in 6(b), it can be seen that boundaries are linear as single linear perceptron was used. Although single linear perceptron also provided 100% accuracy, it should be noted that 3 different single neurons were used as binary classifier which increased the overall computation requirments.

Network (Input-Hidden-Output)	**Train loss**	**Valid loss**	**Confusion Matrix (validation)**	**epoch**
2-1-3	0.543506	0.474592	$\begin{bmatrix} \mathbf{100} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{99} & \mathbf{0} \\ \mathbf{2} & \mathbf{0} & \mathbf{98} \end{bmatrix}$	10000
2-2-3	0.0009994	0.0009991	$\begin{bmatrix} \mathbf{100} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{99} & \mathbf{0} \\ \mathbf{2} & \mathbf{0} & \mathbf{98} \end{bmatrix}$	1774
2-3-3	0.0009992	0.0009980	$\begin{bmatrix} \mathbf{100} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{100} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{100} \end{bmatrix}$	1495
2-4-3	0.0009996	0.0009992	$\begin{bmatrix} \mathbf{100} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{100} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{100} \end{bmatrix}$	1225
2-5-3	0.0009998	0.0009994	$\begin{bmatrix} \mathbf{100} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{100} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{100} \end{bmatrix}$	917

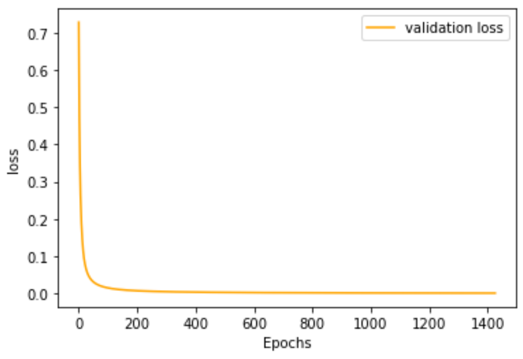
Table 2: Experiments performed to find best architecture

		Actual Class		
		Class 1	Class 2	Class 3
Predicted class	Class 1	100	0	0
	Class 2	0	100	0
	Class 3	0	0	100

Table 3: Confusion matrix for test data

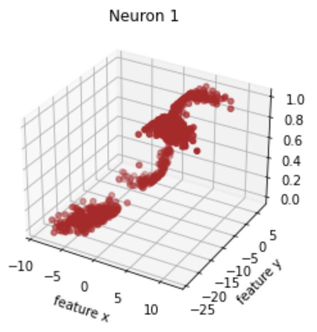


(a)

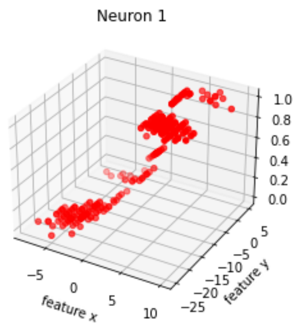


(b)

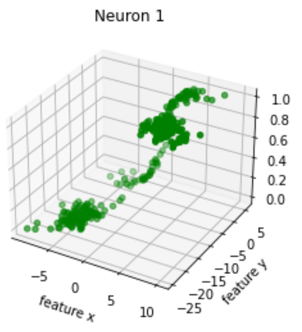
Figure 3: Plot of mean squared error vs epochs for (a) training samples, (b) validation samples



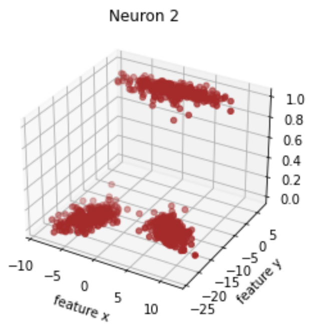
(a) Train



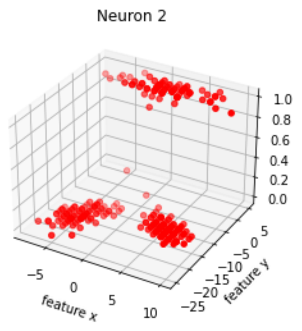
(b) Validation



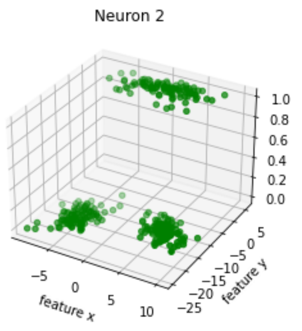
(c) Test



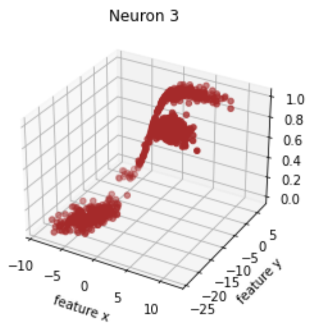
(d) Train



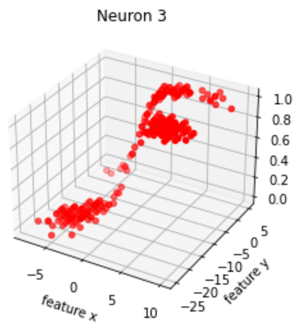
(e) Validation



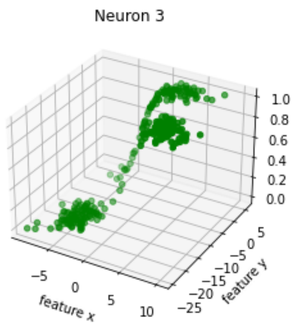
(f) Test



(g) Train

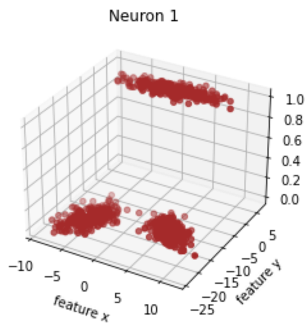


(h) Validation

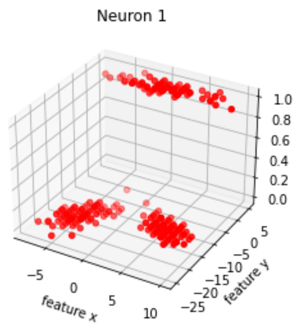


(i) Test

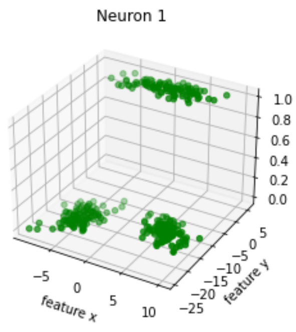
Figure 4: Plots of neurons in hidden layer for training, validation and testing dataset.



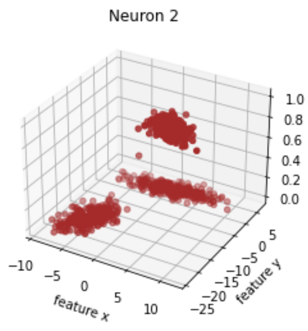
(a) Train



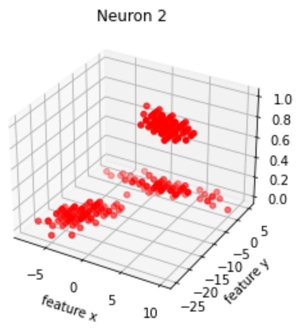
(b) Validation



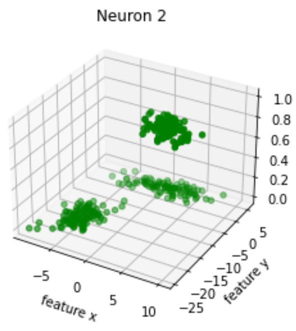
(c) Test



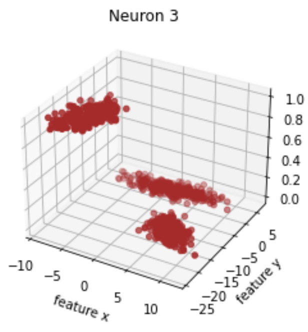
(d) Train



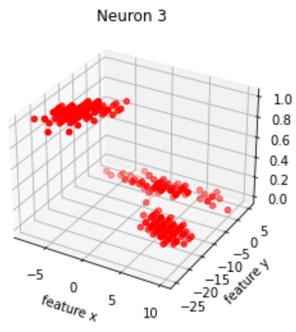
(e) Validation



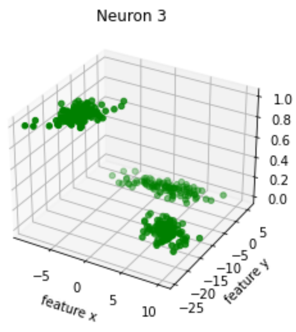
(f) Test



(g) Train

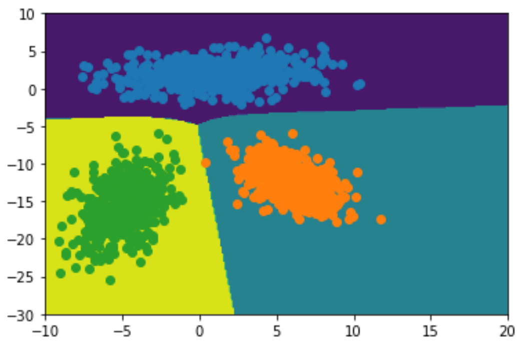


(h) Validation



(i) Test

Figure 5: Plots of neurons in output layer for training, validation and testing dataset.


(a)

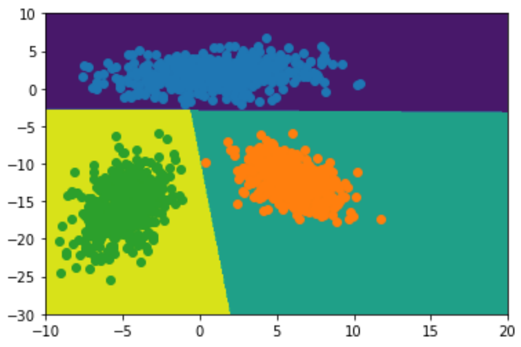

(b)

Figure 6: Decision boundary plots (a) MLP (b) Single linear perceptron

2.2 Classification of non-linearly separable datasets

In this section neural network with 2 hidden layers is used to classify 3 classes. Schematic representation of network is shown in Figure 7 Three types of dataset are provided which are linearly

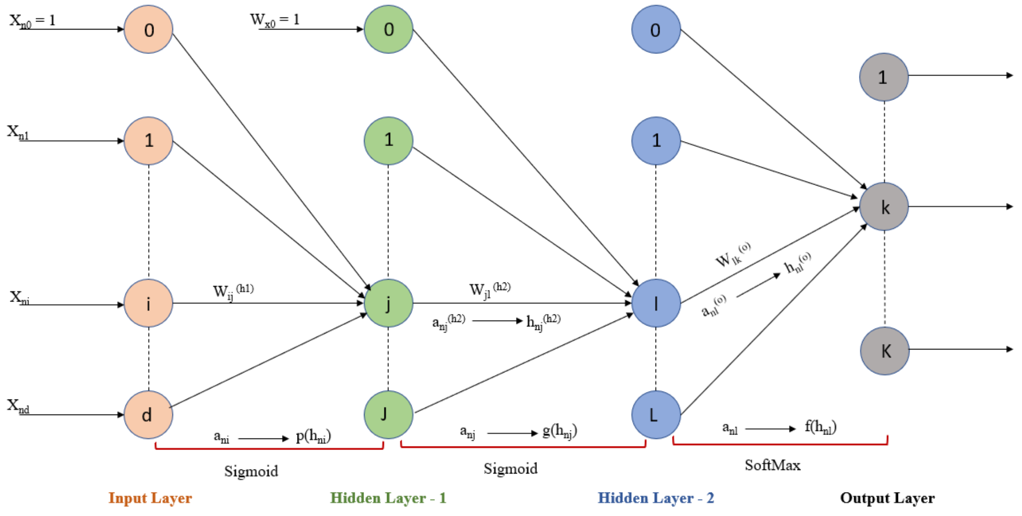

$X_{n0} = 1$
X_{n1}
X_{ni}
X_{nd}
$W_{x0} = 1$
$W_{ij}^{(h1)}$
$W_{jl}^{(h2)}$
$W_{lk}^{(o)}$
$a_{nj}^{(h2)} \rightarrow h_{nj}^{(h2)}$
$a_{nl}^{(o)} \rightarrow h_{nl}^{(o)}$
$a_{ni} \rightarrow p(h_{ni})$
$a_{nj} \rightarrow g(h_{nj})$
$a_{nl} \rightarrow f(h_{nl})$
Sigmoid
Sigmoid
SoftMax
Input Layer
Hidden Layer - 1
Hidden Layer - 2
Output Layer

Figure 7: Neural Network with two hidden layer

seprable. Each dataset is further split into 60% traning, 20% validation, and 20% test samples as shown in Table 4. Three classes of data can be seen in Figure 8. A neural network with two hidden

	Total Samples	Training Samples	Validation Samples	Test Samples
Class 1	500	300	100	100
Class 2	500	300	100	100
Class 3	1000	600	200	200

Table 4: Data division for training, testing and validation

layer, one input layer and one output layer is used to classify the 3 classes. The learning rate used is 0.010. In between input and hidden layer 1 and in between hidden layer 1 and hidden layer 2, logistic activation function is used, while softmax activation function is used in between hidden layer 2 and output layer. The number of neurons in input layer are 2, which are equivalent to number

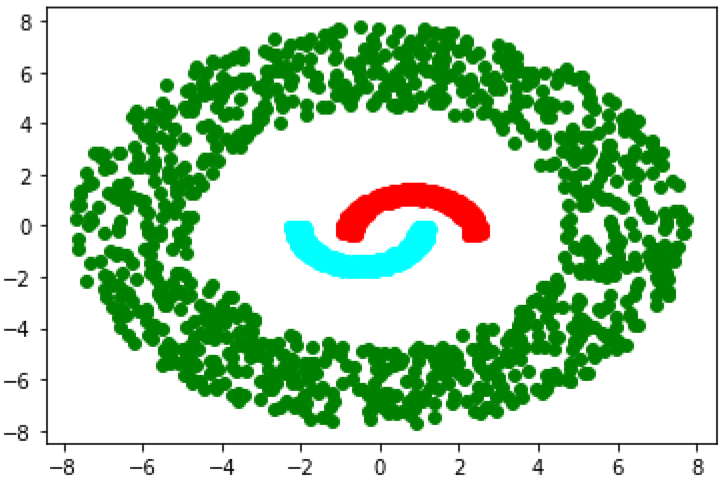


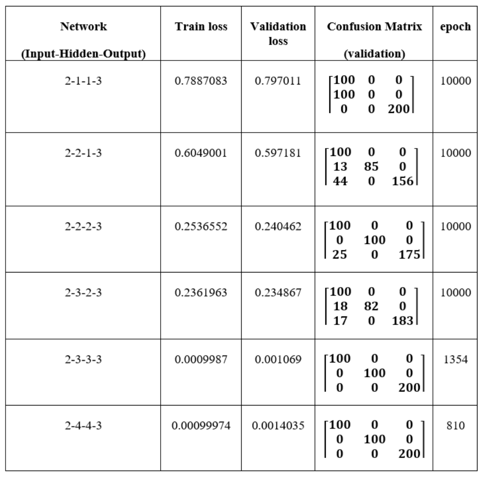
Figure 8: Datasets belonging to 3 classes, red, cyan, green as class 1, class2 and class3 respectively

of features, the number of neurons in output layer are 3. One hot encoding is used to classfiy 3 classes in output layer. For hidden layer cross validation is done by changing number of neurons. As can be seen from table 5, network with 3 hidden nodes in each hidden layer performed well. The validation error starts to increase after 3 neurons, which signifies overfitting of data. Cross entropy loss function with backpropagation algorithm utilizing gradient descent method is used to update weights and minimize errors. The algorithm used to train the network is as follows:

1. Given-training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathrm{R}^d$, and $\mathbf{y}_n \in \mathrm{R}^K$, Initialize the W_{ij} and W_{jk} with random values.
2. Randomly choose a training example x_n
3. Compute output of all output neurons $(k = 1, 2, \dots K)$: s_{nk}
4. Compute the instantaneous error: $E_n = -log(s_{nk})$
5. Update weights between hidden and output layer $(l = 1, 2, \dots L)$ and $(k = 1, 2, \dots K)$:
   - $\Delta w_{lk} = -\eta \frac{\partial E_n}{\partial w_{lk}} = \eta \delta_{nk} h_{nl} = \eta(1 - s_{nk}) h_{nl}$
   - $w_{lk} = w_{lk} + \Delta w_{lk}$
6. Update weights between hidden layer 1 and hidden layer 2 $(j = 1, 2, \dots J)$ and $(l = 1, 2, \dots L)$:
   - $\Delta w_{jl} = -\eta \frac{\partial E_n}{\partial w_{jl}} = \eta \delta_{nl} h_{ni} = \eta(1 - s_{nk}) w_{lk} g(a_{nl})(1 - g(a_{nl})) h_{nj}$
   - $w_{jl} = w_{jl} + \Delta w_{jl}$
7. Update weights between hidden and input layer $(i = 1, 2, \dots d)$ and $(j = 1, 2, \dots J)$:
   - $\Delta w_{ij} = -\eta \frac{\partial E_n}{\partial w_{ij}} = \eta \delta_{nj} x_{ni} = \eta(1 - s_{nk}) w_{lk} g(a_{nl})(1 - g(a_{nl})) w_{jl} P(a_{nj})(1 - P(a_{nj}))(x_{ni}$
   - $w_{ij} = w_{ij} + \Delta w_{ij}$
8. Repeat steps 2 to 7 till all the training examples are presented once (Epoch)
9. Compute the average error: $E_{avg} = \frac{1}{N} \sum E_n$
10. Repeat steps 2 to 9 till the convergence criterion is satisfied

Confusion matrix of test data for best network is given in table 6, it can be observed that accuracy is 100%.

Complete derivation of Δw_{ij}, Δw_{jl}, and Δw_{lk} is available in appendix. Figure 9 depicts the variation of MSE loss with epochs for training and validation samples of best architecture. Stopping criteria was set as to either 10000 epochs are reached or $MSE < 10^{-3}$. Figure 10, Figure 11 and



Network (Input-Hidden-Output)	Train loss	Validation loss	Confusion Matrix (validation)	epoch
2-1-1-3	0.7887083	0.797011	$\begin{bmatrix} 100 & 0 & 0 \\ 100 & 0 & 0 \\ 0 & 0 & 200 \end{bmatrix}$	10000
2-2-1-3	0.6049001	0.597181	$\begin{bmatrix} 100 & 0 & 0 \\ 13 & 85 & 0 \\ 44 & 0 & 156 \end{bmatrix}$	10000
2-2-2-3	0.2536552	0.240462	$\begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 25 & 0 & 175 \end{bmatrix}$	10000
2-3-2-3	0.2361963	0.234867	$\begin{bmatrix} 100 & 0 & 0 \\ 18 & 82 & 0 \\ 17 & 0 & 183 \end{bmatrix}$	10000
2-3-3-3	0.0009987	0.001069	$\begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 200 \end{bmatrix}$	1354
2-4-4-3	0.00099974	0.0014035	$\begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 200 \end{bmatrix}$	810

Table 5: Experiments performed to find best architecture

Predicted class		**Actual Class**		
		Class 1	Class 2	Class 3
	Class 1	100	0	0
	Class 2	0	100	0
	Class 3	0	0	200

Table 6: Confusion matrix of test data

shows the output of individual neurons of hidden layer 1 and hidden layer 2 on training, validation, and testing data. It can be observed that individual neuron is trying to learn pieces of function from the dataset. Figure 12 represents the output of each neuron in output layer, it can be observed that neurons are able to classify difference between 3 classes, as individual neuron is acting as binary classifier, which classify one class from other two classes. On plotting decision region plot superimposed by training data, and also observing confusion matrix of test samples, it can be observed from Figure 13(a) that neural network classified all the 3 classes with 100% accuracy. On comparing with single nonlinear perceptron output in 13(b), it can be seen that single linear perceptron failed to classify non-linear separable data due to limited computation capacity of single linear neuron.

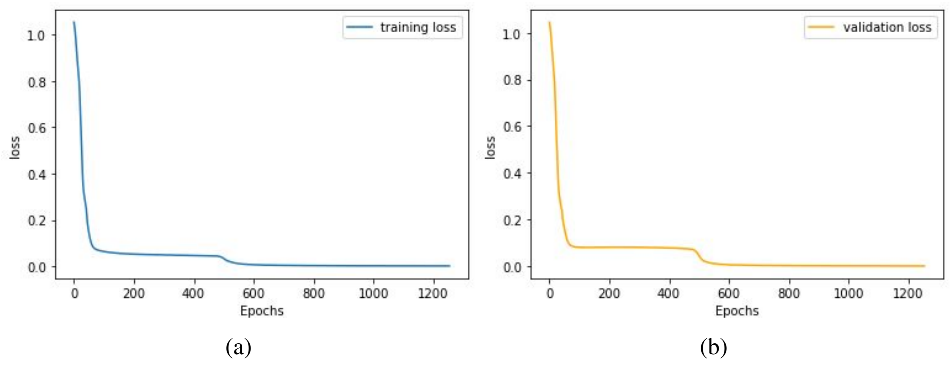
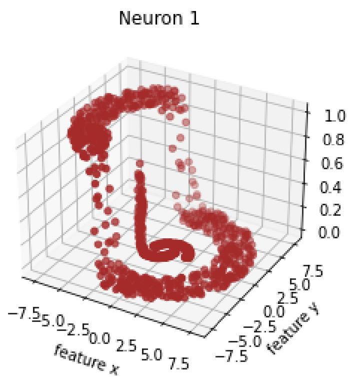
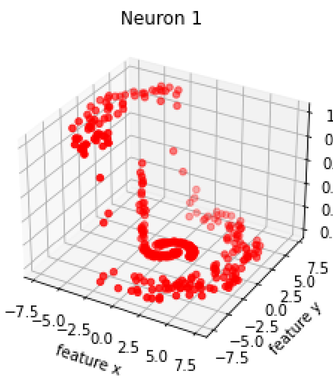


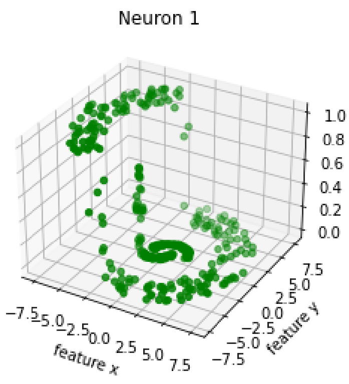
Figure 9: Plot of mean squared error vs epochs for (a) training samples, (b) validation samples



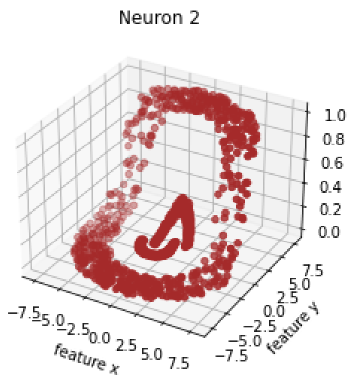
(a) Train



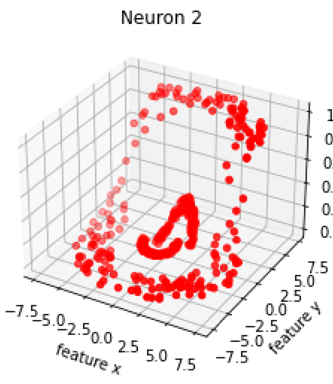
(b) Validation



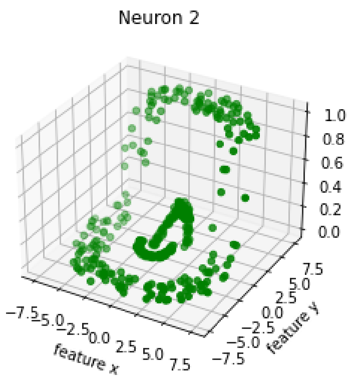
(c) Test



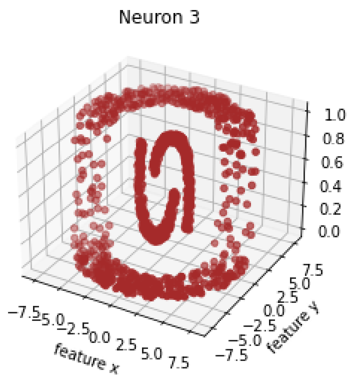
(d) Train



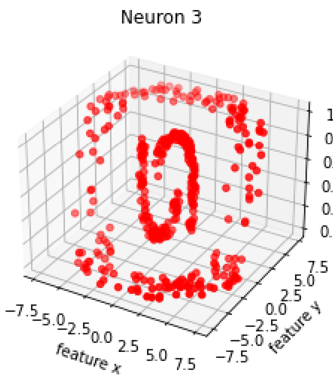
(e) Validation



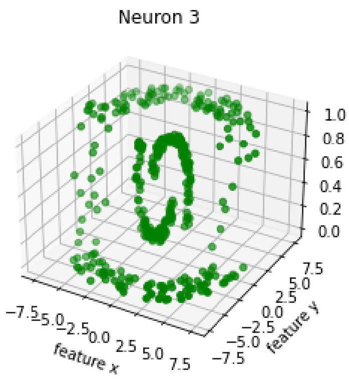
(f) Test



(g) Train

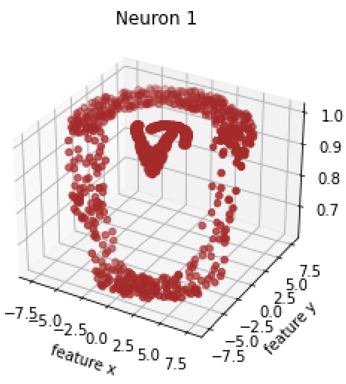


(h) Validation

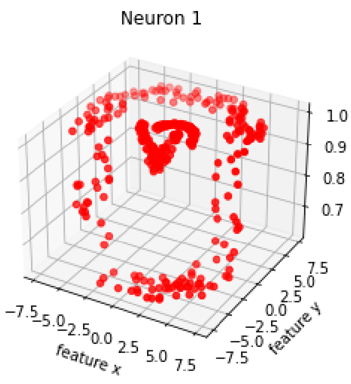


(i) Test

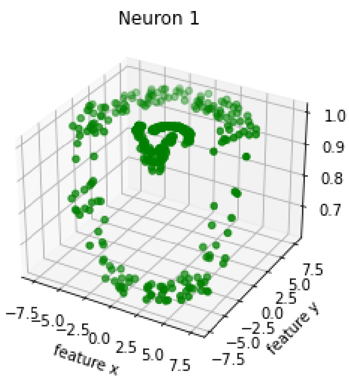
Figure 10: Plots of neurons in 1st hidden layer for training, validation, and testing dataset.



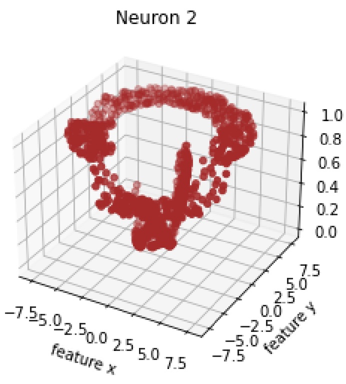
(a) Train



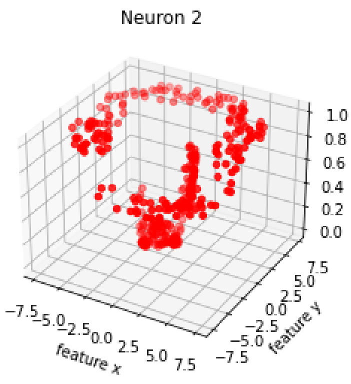
(b) Validation



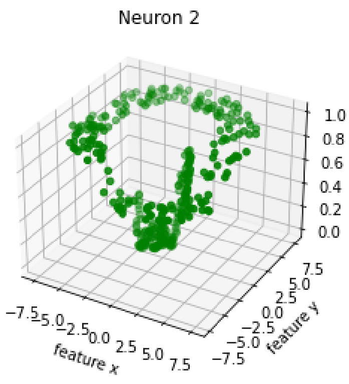
(c) Test



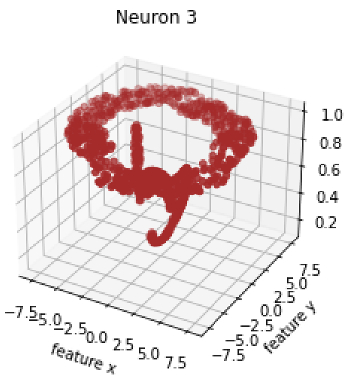
(d) Train



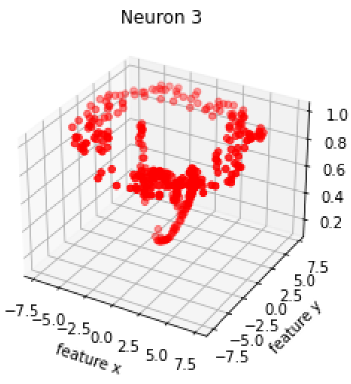
(e) Validation



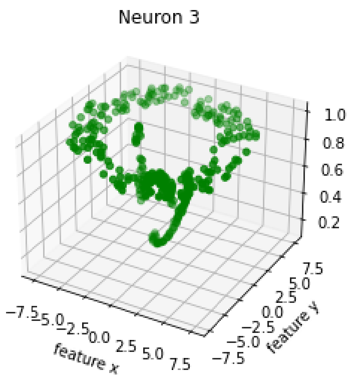
(f) Test



(g) Train

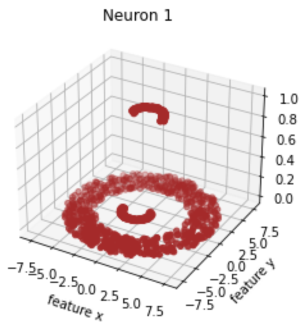


(h) Validation

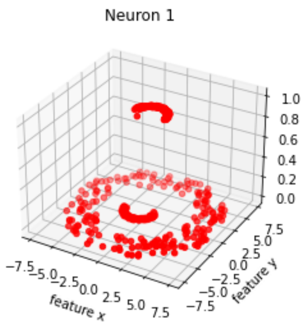


(i) Test

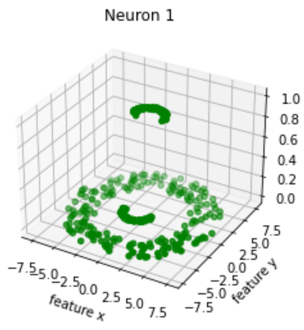
Figure 11: Plots of neurons in 2nd hidden layer for training, validation, and testing dataset.



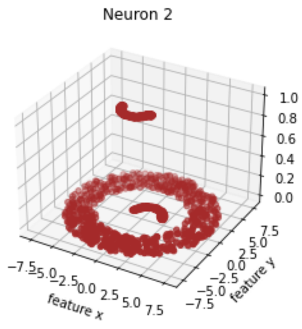
(a) Train



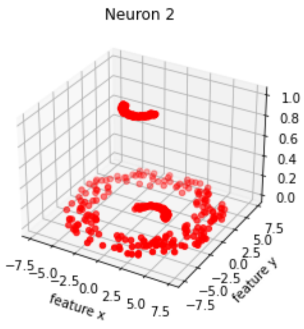
(b) Validation



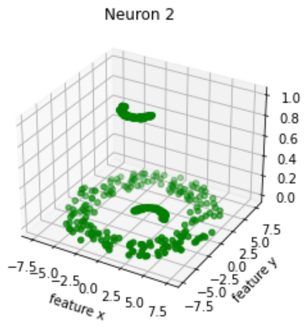
(c) Test



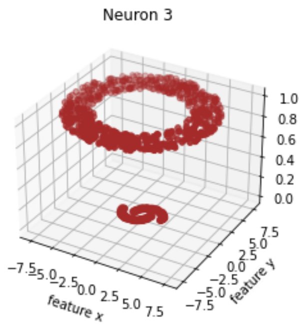
(d) Train



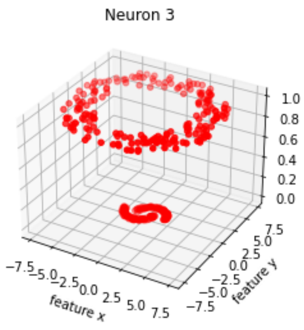
(e) Validation



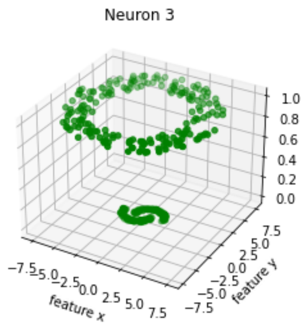
(f) Test



(g) Train

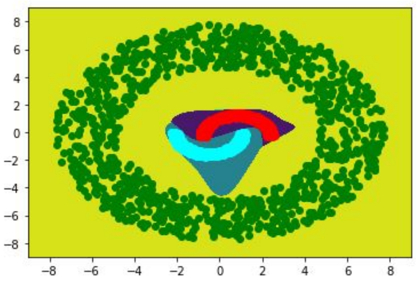


(h) Validation

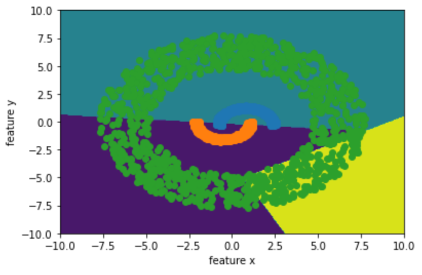


(i) Test

Figure 12: Plots of neurons in output layer for training, validation, and testing dataset.



(a)



(b)

Figure 13: Decision boundary plots (a) MLP (b) Single linear perceptron

3 Regression

3.1 Regression of univariate dataset

In this section MLP model is trained to fit the one dimensional univariate and bivariate data. Schematic representation of neural network is shown in Figure 14 Dataset comprised of 1000 sam-

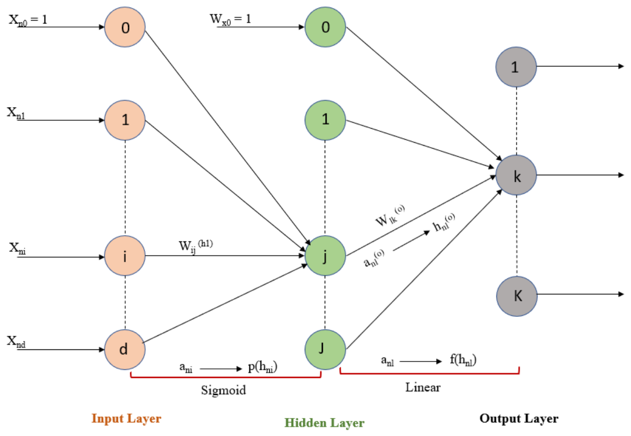


Figure 14: Neural Network with single hidden layer

ples, out of which 60% are used for training and the remaining 20% each for validation and testing as shown in Table 7 and Figure 15. For fitting the line around univariate data, a neural network with

Total Samples	Training Samples	Validation Samples	Test Samples
1000	600	200	200

Table 7: Data division for training, testing and validation

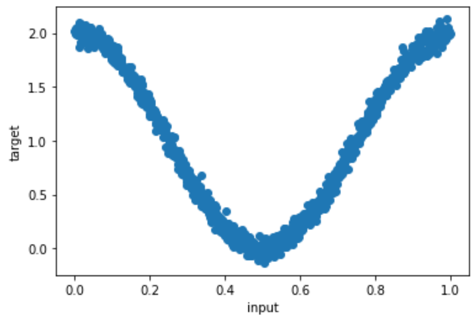
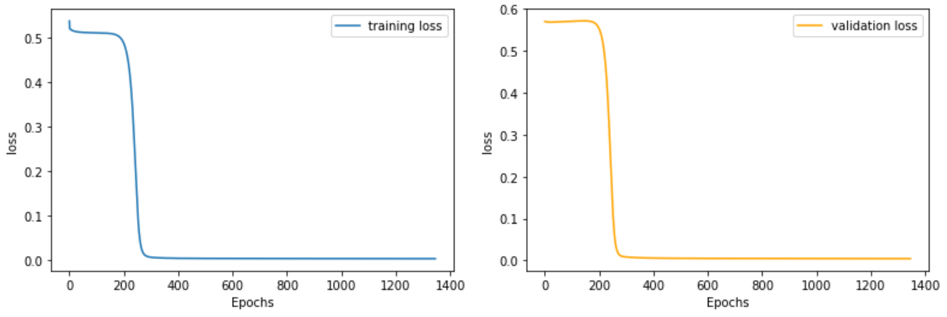


Figure 15: Univariate data required for training

one hidden layer, one input layer and one output layer is used. The learning rate used is 0.010. In between input and hidden layer, logistic activation function is used, while linear activation function is used in between hidden and output layer. The number of neurons in input layer are 1, which are equivalent to number of features, the number of neurons in output layer are 1. For hidden layer cross validation is done by changing number of neurons. As can be seen from table, network with 3 hidden nodes performed well. The validation error starts to increase after 3 neurons, which signifies overfitting of data. Mean squared loss function with backpropagation algorithm utilizing gradient descent method is used to update weights and minimize errors. The algorithm used to train the network is as follows:



(a) (b)

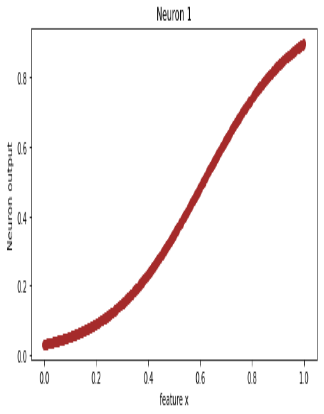
Figure 16: Plot of mean squared error vs epochs for (a) training samples, (b) validation samples

1. Given-training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathrm{R}^d$, and $\mathbf{y}_n \in \mathrm{R}^K$, Initialize the W_{ij} and W_{jk} with random values.
2. Randomly choose a training example x_n
3. Compute output of all output neurons $(k = 1, 2, \ldots K)$: s_{nk}
4. Compute the instantaneous error: $E_n = \frac{1}{2N} \sum (y_{nk} - s_{nk})^2$
5. Update weights between hidden and output layer $(j = 1, 2, \ldots J)$ and $(k = 1, 2, \ldots K)$:
   - $\Delta w_{jk} = -\eta \frac{\partial E_n}{\partial w_{jk}} = \eta \delta_{nk} h_{nj} = \eta (y_{nk} - s_{nk}) h_{nj}$
   - $w_{jk} = w_{jk} + \Delta w_{jk}$
6. Update weights between hidden and input layer $(i = 1, 2, \ldots d)$ and $(j = 1, 2, \ldots J)$:
   - $\Delta w_{ij} = -\eta \frac{\partial E_n}{\partial w_{ij}} = \eta \delta_{nj} h_{ni} = \eta (y_{nk} - s_{nk}) w_{jk} g(a_{nj})(1 - g(a_{nj})) x_{ni}$
   - $w_{ij} = w_{ij} + \Delta w_{ij}$
7. Repeat steps 2 to 6 till all the training examples are presented once (Epoch)
8. Compute the average error: $E_{avg} = \frac{1}{N} \sum E_n$
9. Repeat steps 2 to 8 till the convergence criterion is satisfied

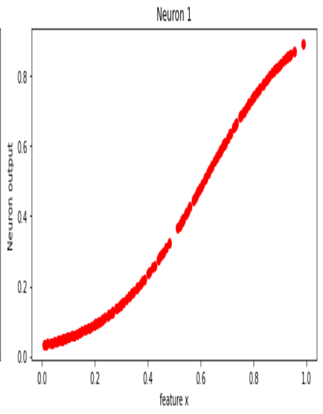
Network	Train loss	Valid loss	Epochs
$1-1-1$	0.21556879	0.2058568	10000
$1-2-1$	0.00640416	0.0072134	10000
$1-3-1$	0.00384584	0.0038458	1385
$1-4-1$	0.00399622	0.0039483	269
$1-5-1$	0.00399956	0.0041764	284

Table 8: Variation of loss with variation in network architecture

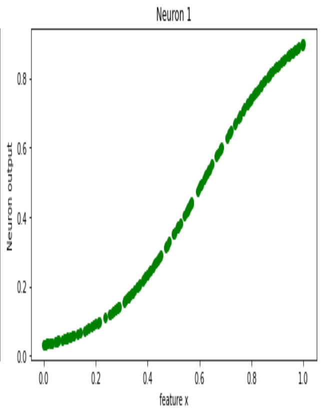
Figure 16 depicts the variation of MSE loss with epochs for training and validation samples of best architecture. Stopping criteria was set as to either 10000 epochs are reached or $MSE < 0.004$.It can be observed from table 8 that network with 3 nodes performs well. Figure 17 shows the output of individual neurons on training, validation, and testing data. It can be observed that individual neuron is trying to learn pieces of function from the dataset. Figure 18 represents the output of neuron in output layer, it can be observed output neuron is able to, combine all the peicewise information obtaned from hidden layer to yield desired output. Figure 19 shows the plots of model output and target output for training data, validation data and test data for best architecture i.e. network with 3 hidden neurons. Figure 20 shows the how much good our model is fitting the data, it can be seen that for training, validation and testing data is forming a straight line with slope of 45 degrees, which represents a good fit.



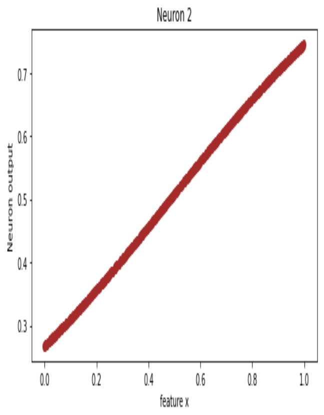
(a) Train



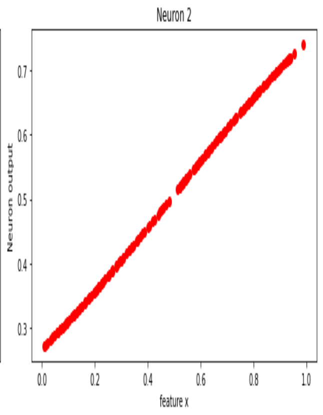
(b) Validation



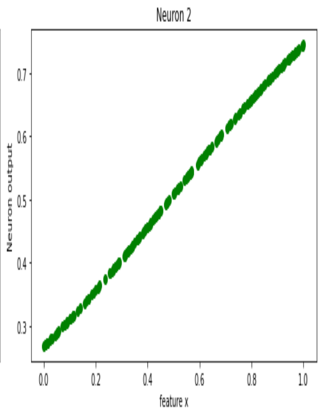
(c) Test



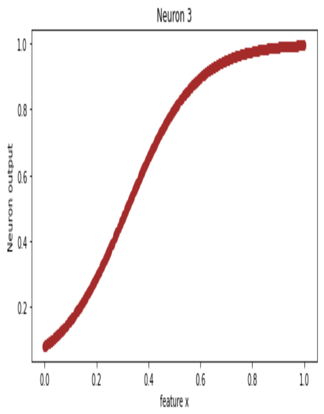
(d) Train



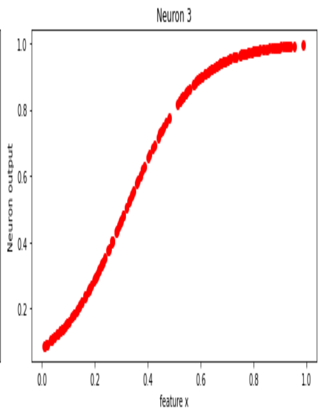
(e) Validation



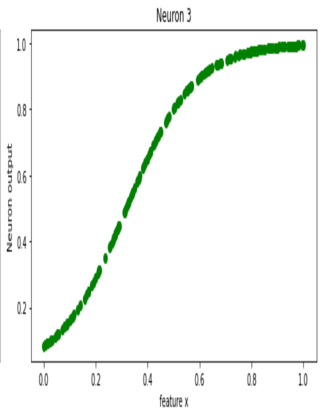
(f) Test



(g) Train

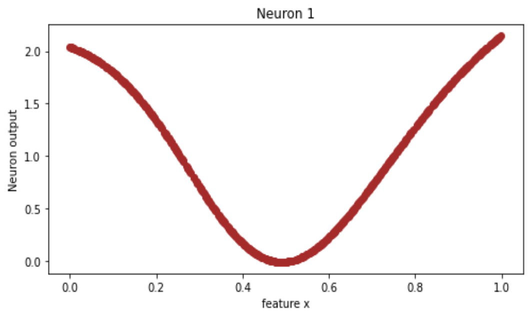


(h) Validation

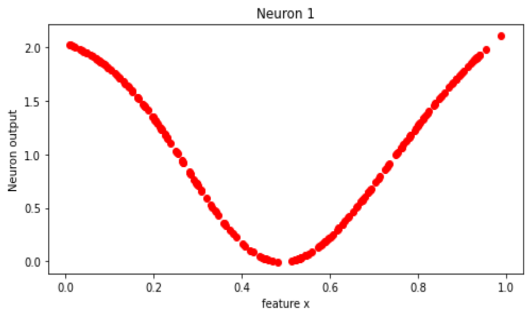


(i) Test

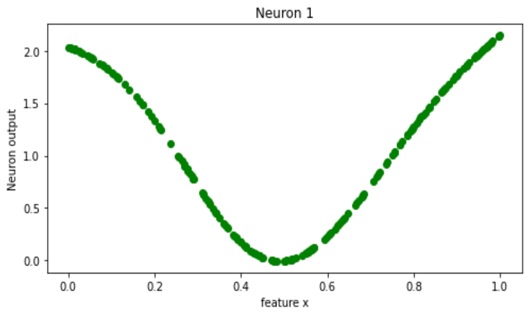
Figure 17: Plots of neurons in hidden layer for training, validation, and testing dataset.



(a) Train

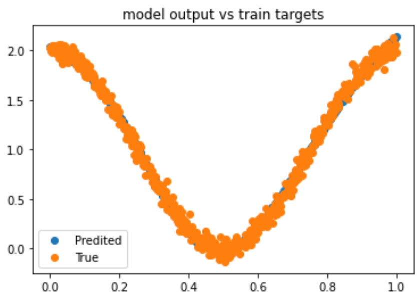


(b) Validation

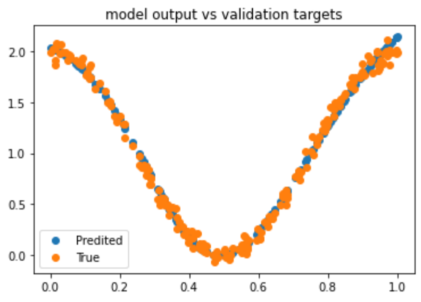


(c) Test

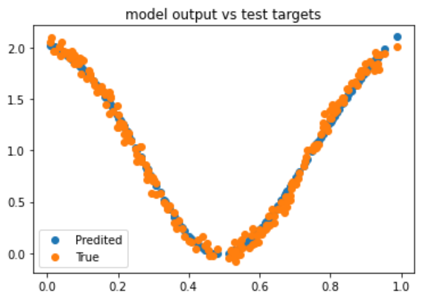
Figure 18: Plots of neurons in output layer for training, validation, and testing dataset.



(a)

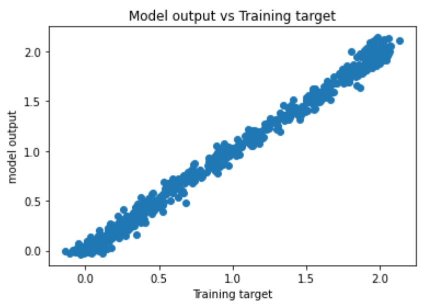


(b)

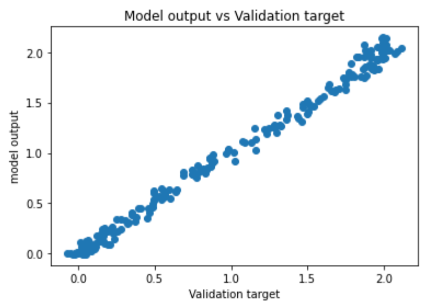


(c)

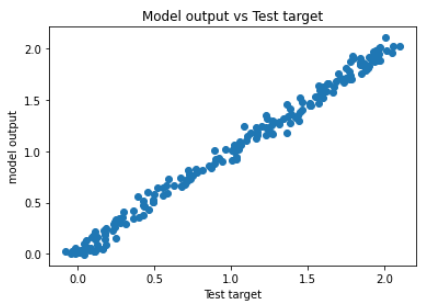
Figure 19: (a) Model output vs train targets, (b) Model output vs validation targets, (c) Model output vs test targets



(a)



(b)



(c)

Figure 20: Scatter plot showing goodness of fit for (a) training, (b) validation, and (c) test dataset

3.2 Regression of bivariate dataset

Neural network with 2 hidden layer is used to fit nonlinear bivariate data. Dataset comprised of 10200 samples with two features, out of which 60% are used for training and the remaining 20% each for validation and testing as shown in Table 9 and Figure 22. Schematic representation of neural network is shown in Figure 21 For fitting the surface around bivariate data, first cross validation

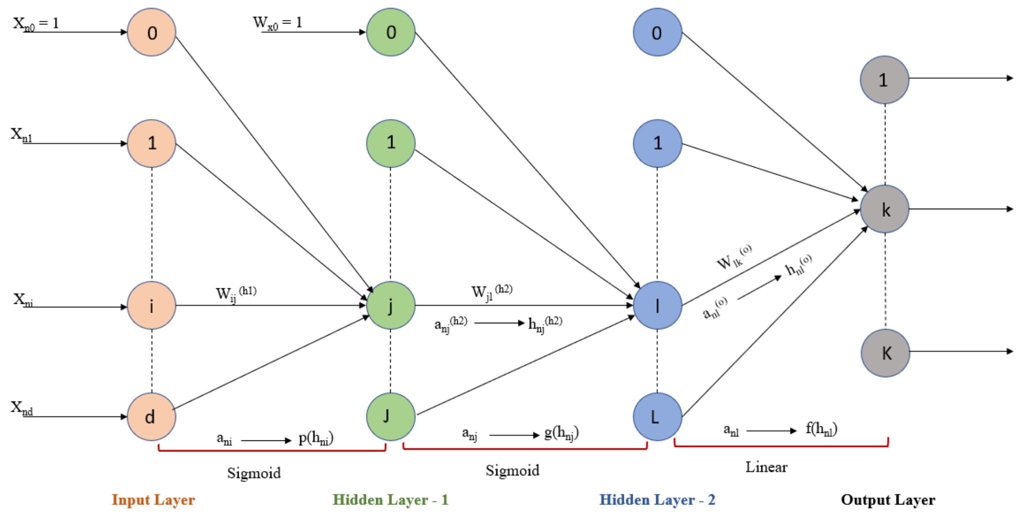

$X_{n0} = 1$
X_{n1}
X_{ni}
X_{nd}
$W_{x0} = 1$
$W_{ij}^{(h1)}$
$W_{jl}^{(h2)}$
$a_{nj}^{(h2)} \rightarrow h_{nj}^{(h2)}$
$W_{lk}^{(o)}$
$a_{nl}^{(o)} \rightarrow h_{nl}^{(o)}$
$a_{ni} \rightarrow p(h_{ni})$
$a_{nj} \rightarrow g(h_{nj})$
$a_{nl} \rightarrow f(h_{nl})$
Sigmoid
Sigmoid
Linear
Input Layer
Hidden Layer - 1
Hidden Layer - 2
Output Layer

Figure 21: Neural Network with 2 hidden layers

is done to decide the best architecture. It can be observed from table 10, neural network with

Total Samples	**Training Samples**	**Validation Samples**	**Test Samples**
10200	6120	2040	2040

Table 9: Data division for training, testing and validation

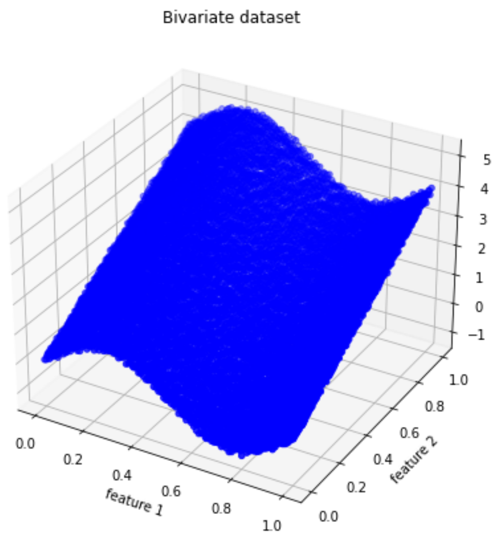


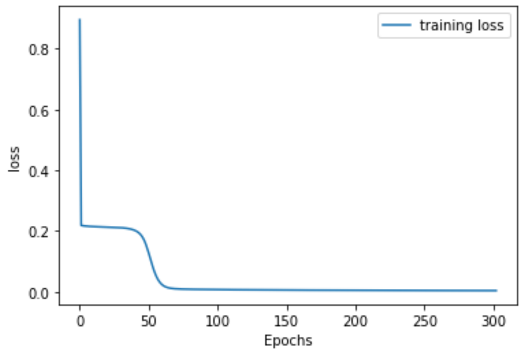
Figure 22: Bivariate data required for training

twp hidden layer, 4 neurons each performed best. The learning rate used is 0.010. In between input and hidden layer 1, and also between hidden layer 1 and hidden layer 2 logistic activation function is used, while linear activation function is used in between hidden and output layer. The number of neurons in input layer are 2, which are equivalent to number of features, the number of neurons in output layer are 1. The validation error starts to increase after 4 neurons, which signifies overfitting of data. Mean squared loss function with backpropagation algorithm utilizing gradient descent method is used to update weights and minimize errors. The algorithm used to train the network is as follows:

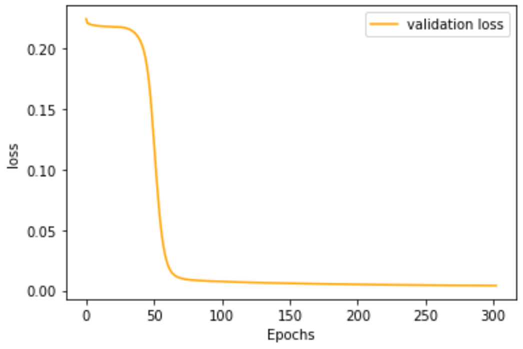
1. Given-training data: $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$, $\mathbf{x}_n \in \mathrm{R}^d$, and $\mathbf{y}_n \in \mathrm{R}^K$, Initialize the W_{ij} and W_{jk} with random values.
2. Randomly choose a training example x_n
3. Compute output of all output neurons ($k = 1, 2, \dots K$): s_{nk}
4. Compute the instantaneous error: $E_n = \frac{1}{2N} \sum (y_{nk} - s_{nk})^2$
5. Update weights between hidden and output layer ($l = 1, 2, \dots L$) and ($k = 1, 2, \dots K$):
   - $\Delta w_{lk} = -\eta \frac{\partial E_n}{\partial w_{lk}} = \eta \delta_{nk} h_{nl} = \eta(y_{nk} - s_{nk}) h_{nl}$
   - $w_{lk} = w_{lk} + \Delta w_{lk}$
6. Update weights between hidden layer 1 and hidden layer 2 ($j = 1, 2, \dots J$) and ($l = 1, 2, \dots L$):
   - $\Delta w_{jl} = -\eta \frac{\partial E_n}{\partial w_{jl}} = \eta \delta_{nl} h_{ni} = \eta(y_{nk} - s_{nk}) w_{lk} g(a_{nl})(1 - g(a_{nl})) h_{nj}$
   - $w_{jl} = w_{jl} + \Delta w_{jl}$
7. Update weights between hidden and input layer ($i = 1, 2, \dots d$) and ($j = 1, 2, \dots J$):
   - $\Delta w_{ij} = -\eta \frac{\partial E_n}{\partial w_{ij}} = \eta \delta_{nj} x_{ni} = \eta(y_{nk} - s_{nk}) w_{lk} g(a_{nl})(1 - g(a_{nl})) w_{jl} P(a_{nj})(1 - P(a_{nj}))(x_{ni}$
   - $w_{ij} = w_{ij} + \Delta w_{ij}$
8. Repeat steps 2 to 7 till all the training examples are presented once (Epoch)

Network	Train loss	Valid loss	epoch
$2-1-1$	0.21556879	0.2058568	10000
$2-2-1$	0.00640416	0.0072134	10000
$2-3-1$	0.00384584	0.0038458	950
$2-4-1$	0.00399622	0.0039483	269
$2-5-1$	0.00399956	0.0041764	284
$2-1-1-1$	0.2260894	0.2149359	10000
$2-2-2-1$	0.00511609	0.0045612	10000
$2-3-3-1$	0.00399973	0.0046650	2019
$2-3-4-1$	0.00399973	0.00459896	849
$2-4-4-1$	0.00399969	0.00350031	310
$2-4-5-1$	0.00399996	0.00379057	270

Table 10: Variation of loss with change in network



(a)

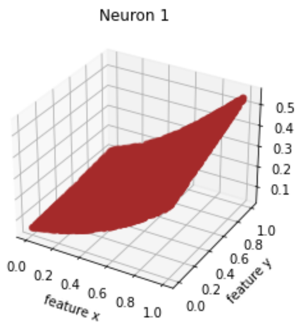


(b)

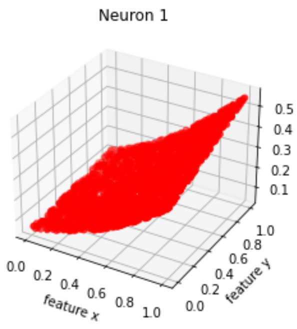
Figure 23: Plot of mean squared error vs epochs for (a) training samples, (b) validation samples

9. Compute the average error: $E_{avg} = \frac{1}{N} \sum E_n$
10. Repeat steps 2 to 9 till the convergence criterion is satisfied

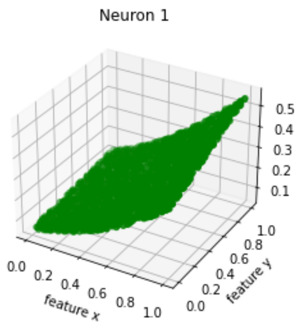
Figure 23 depicts the variation of MSE loss with epochs for training and validation samples of best architecture. Stopping criteria was set as to either 10000 epochs are reached or $MSE < 0.004$. Figure 24 shows the output of individual neurons on training, validation, and testing data. It can be observed from Figure 24 and Figure 25 that individual neuron is trying to learn pieces of function from the dataset. Figure 26 represents the output of neuron in output layer, it can be observed output neuron is able to, combine all the peicewise information obtaned from hidden layer to yield desired output. Figure 27 shows the plots of model output and target output for training data, validation data and test data for best architecture i.e. network with 2 hidden layers having 4 neurons each. Figure 28 shows the how much good our model is fitting the data, it can be seen that for training, validation and testing data is forming a straight line with slope of 45 degrees, which represents a good fit.



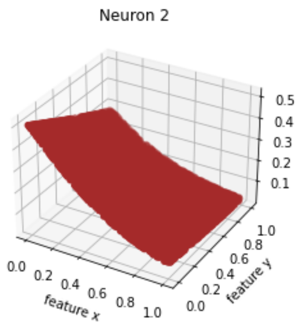
(a) Train



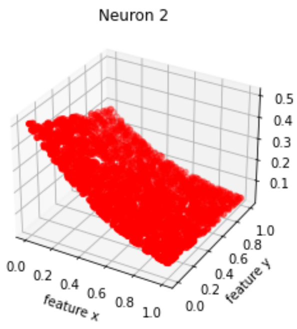
(b) Validation



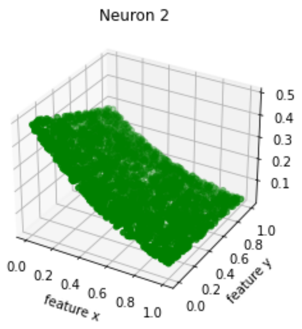
(c) Test



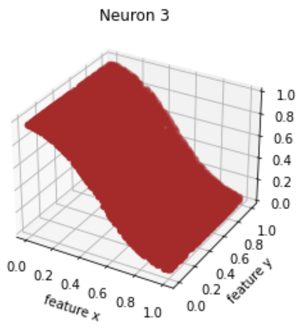
(d) Train



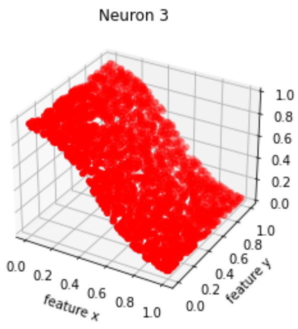
(e) Validation



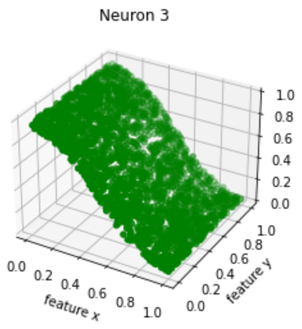
(f) Test



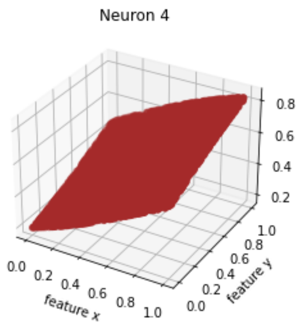
(g) Train



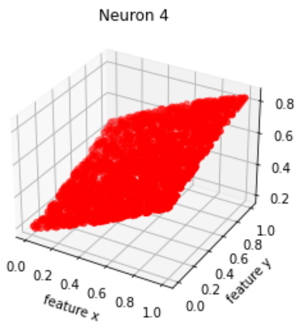
(h) Validation



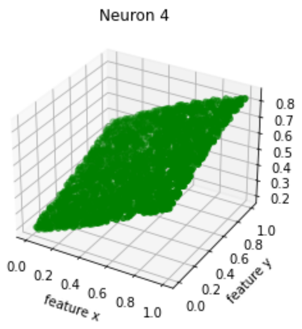
(i) Test



(j) Train

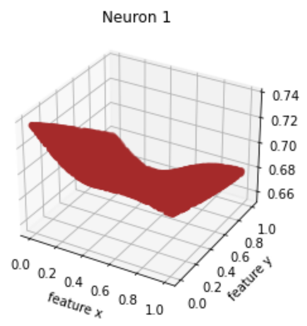


(k) Validation

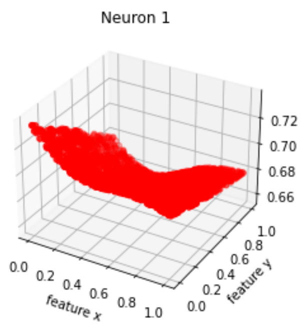


(l) Test

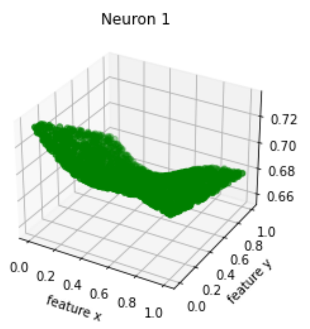
Figure 24: Plots of neurons in 1st hidden layer for training, validation, and testing dataset.



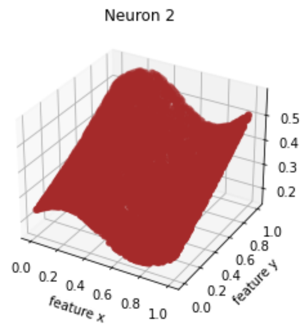
(a) Train



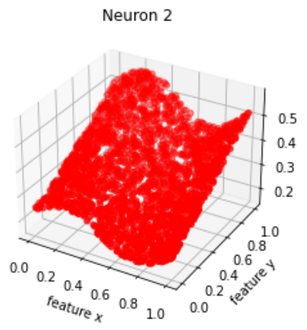
(b) Validation



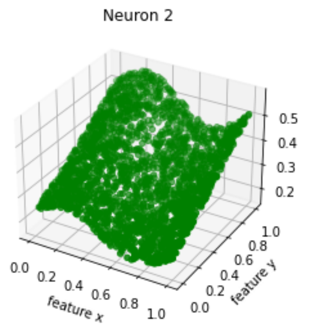
(c) Test



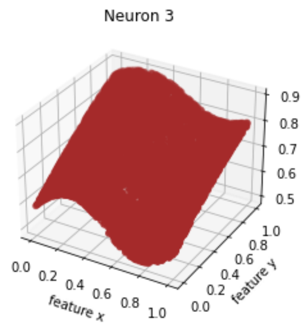
(d) Train



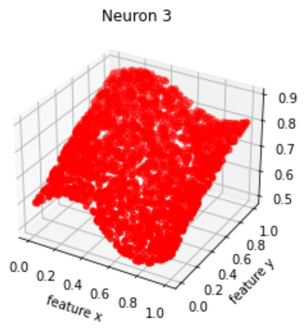
(e) Validation



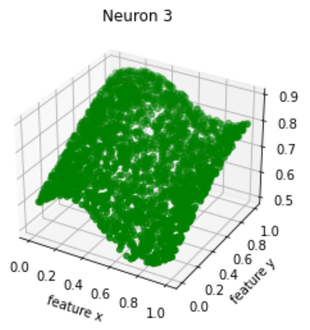
(f) Test



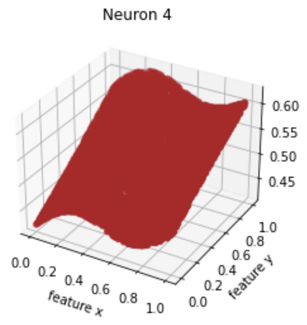
(g) Train



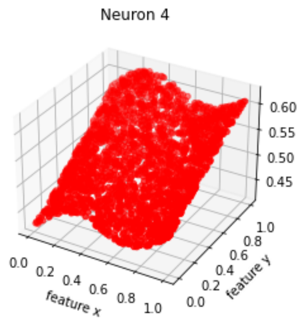
(h) Validation



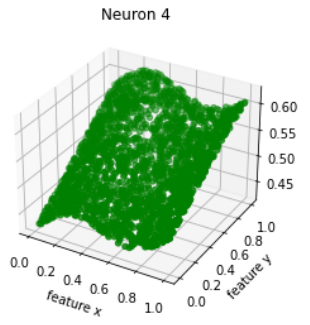
(i) Test



(j) Train

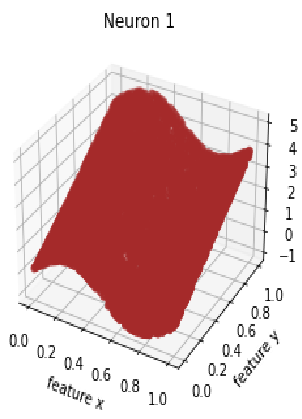


(k) Validation

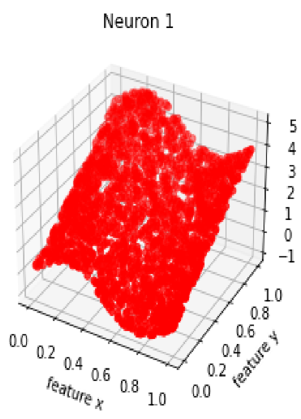


(l) Test

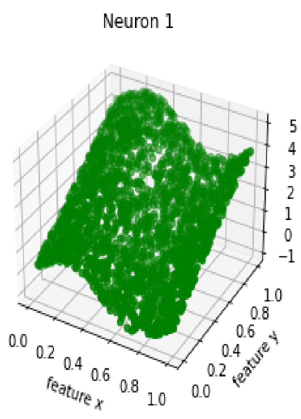
Figure 25: Plots of neurons in 2nd hidden layer for training, validation, and testing dataset.



(a) Train

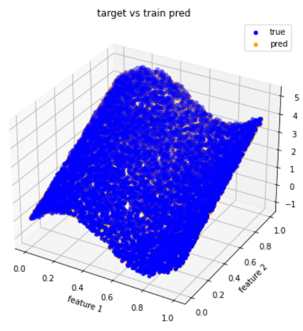


(b) Validation

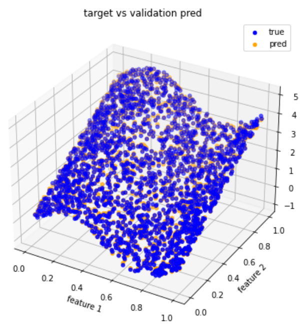


(c) Test

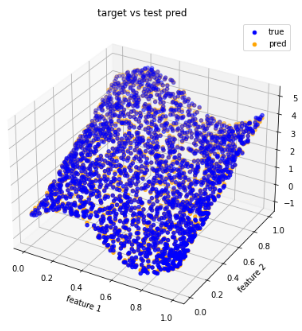
Figure 26: Plots of neurons in output layer for training, validation, and testing dataset.



(a)

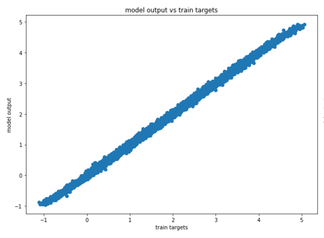


(b)

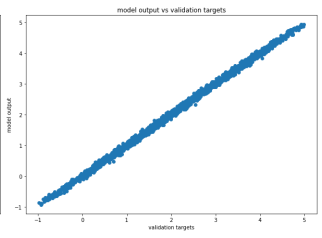


(c)

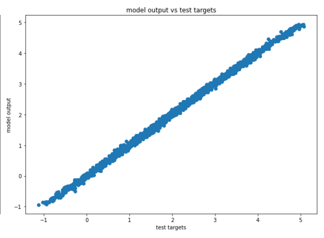
Figure 27: (a) Model output vs train targets, (b) Model output vs validation targets, (c) Model output vs test targets



(a)



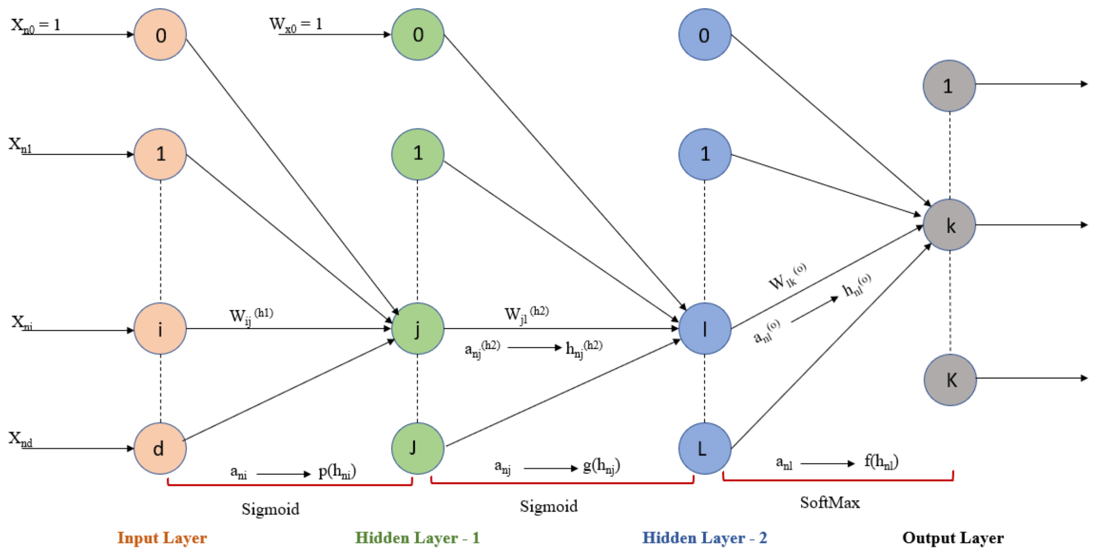
(b)



(c)

Figure 28: Scatter plot showing goodness of fit for (a) training, (b) validation, (c) test dataset

Appendix



Obtain Δw_{lk}, Δw_{jl}, Δw_{ij}

$$E_n = -\log(S_{nk})$$

$$\Delta w_{lk} = \frac{-\eta \partial E_n}{\partial w_{lk}}$$

$$\frac{\partial \log(S_{nk})}{\partial w_{lk}} = \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial w_{lk}}$$

$$= \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial (a_{nk})} \frac{\partial (a_{nk})}{\partial w_{lk}}$$

$$= (1 - S_{nk}) \frac{\partial (a_{nk})}{\partial w_{lk}}$$

Therefore,

$$\Delta w_{lk} = (1 - S_{nk}) h_{nl}$$

$$\Delta w_{jl} = \frac{-\eta \partial E_n}{\partial w_{jl}}$$

$$\frac{\partial \log(S_{nk})}{\partial w_{jl}} = \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial w_{jl}}$$

$$= \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial (a_{nk})} \frac{\partial (a_{nk})}{\partial w_{jl}}$$

$$= (1 - S_{nk}) \frac{\partial(a_{nk})}{\partial w_{jl}}$$

$$= (1 - S_{nk}) \frac{\partial(w_{lk}.h_{nl})}{\partial w_{jl}}$$

$$= (1 - S_{nk})\, w_{lk} \frac{\partial h_{nl}}{\partial w_{jl}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \; (1)$$

$$\frac{\partial h_{nl}}{\partial w_{jl}} = \frac{\partial g(a_{nl})}{\partial w_{jl}}$$

$$\frac{\partial h_{nl}}{\partial w_{jl}} = \frac{\partial g(a_{nl})}{\partial a_{nl}} \frac{\partial a_{nl}}{\partial w_{jl}}$$

$$= g(a_{nl})(1 - g(a_{nl})) \frac{\partial(w_{jl}h_{nj})}{\partial w_{jl}}$$

$$\frac{\partial h_{nl}}{\partial w_{jl}} = g(a_{nl}).\big(1 - g(a_{nl})\big).h_{nj} \quad \ldots\ldots\ldots\ldots\ldots..(2)$$

Put (2) in (1)

$$\Delta \mathrm{w}_{jl} = \underbrace{(1 - S_{nk}).w_{lk}.g(a_{nl}).(1 - g(a_{nl}))}_{\delta_{nl}}.h_{nj}$$

$$\Delta \mathrm{w}_{jl} = \eta \delta_{nl} h_{nj}$$

$$\Delta \mathrm{w}_{ij} = -\eta \frac{\partial E_n}{\partial w_{ij}}$$

$$= \frac{\eta\, \partial(log(S_{nk}))}{\partial w_{ij}}$$

$$= \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial(a_{nk})} \frac{\partial(a_{nk})}{\partial w_{ij}}$$

$$= (1 - S_{nk}) \frac{\partial(a_{nk})}{\partial w_{ij}}$$

$$= (1 - S_{nk}) \frac{\partial(w_{lk}.h_{nl})}{\partial w_{ij}}$$

$$= (1 - S_{nk}) w_{lk} \frac{\partial h_{nl}}{\partial w_{ij}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.(3)$$

$$\frac{\partial h_{nl}}{\partial w_{ij}} = \frac{\partial g(a_{nl})}{\partial w_{ij}}$$

$$= \frac{\partial g(a_{nl})}{\partial a_{nl}} \frac{\partial a_{nl}}{\partial w_{ij}}$$

$$= g(a_{nl})(1 - g(a_{nl}))\frac{\partial(w_{jl}h_{nj})}{\partial w_{ij}}$$

$$= g(a_{nl})(1 - g(a_{nl}))\frac{\partial(w_{jl}h_{nj})}{\partial w_{ij}}$$

$$\frac{\partial h_{nl}}{\partial w_{ij}} = g(a_{nl}).\left(1 - g(a_{nl})\right).w_{jp}\frac{\partial\left(h_{nj}\right)}{\partial w_{ij}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..(4)$$

$$\frac{\partial h_{nj}}{\partial w_{ij}} = \frac{\partial p(a_{nj})}{\partial w_{ij}}$$

$$= \frac{\partial p(a_{nj})}{\partial a_{nj}}\frac{\partial a_{nj}}{\partial w_{ij}}$$

$$= p(a_{nj})(1 - p(a_{nj}))\frac{\partial(w_{ij}x_{ni})}{\partial w_{ij}}$$

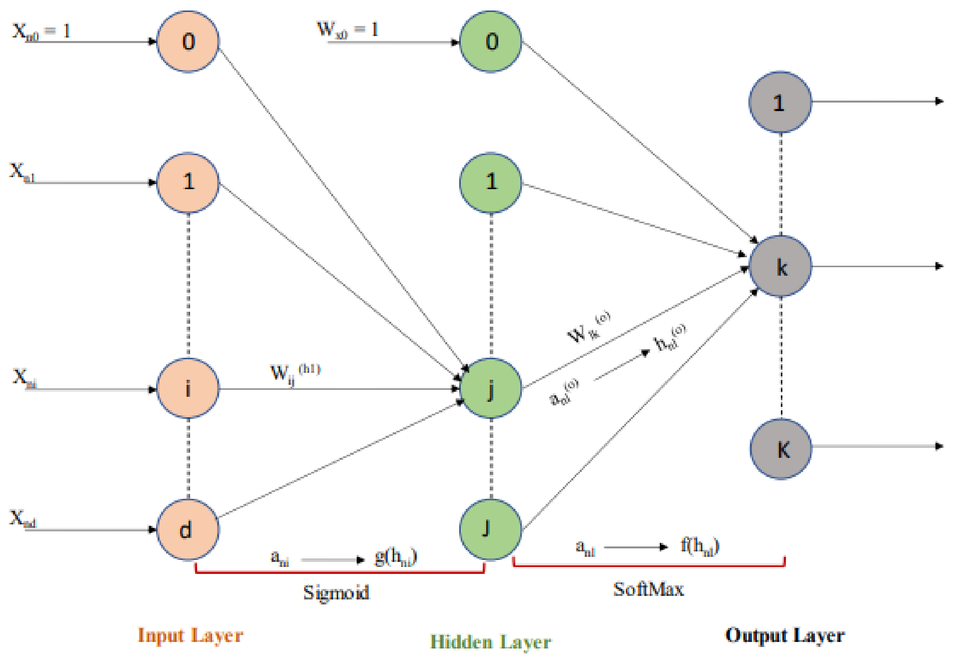
$$= p\left(a_{nj}\right).\left(1 - p\left(a_{nj}\right)\right).x_{ni} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

Put (5) in (4)

$$\frac{\partial h_{nl}}{\partial w_{ij}} = g(a_{nl}).(1 - g(a_{nl})).w_{jl}p(a_{nj}).(1 - p(a_{nj})).x_{ni} \qquad (6)$$

Put (6) in (3)

$$\Delta \mathrm{w}_{ij} = (1 - S_{nk}).w_{lk}.g(a_{nl}).\left(1 - g(a_{nl})\right).w_{jl}p(a_{nj}).(1 - p(a_{nj})).x_{ni}$$



Obtain $\Delta \mathrm{w}_{jk}$, $\Delta \mathrm{w}_{ij}$

$$\mathrm{E}_n = -\log(S_{nk})$$

$$\Delta \mathrm{w}_{jk} = \frac{-\eta \partial E_n}{\partial w_{jk}}$$

$$\frac{\partial \log(S_{nk})}{\partial w_{jk}} = \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial w_{jk}}$$

$$= \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial (a_{nk})} \frac{\partial (a_{nk})}{\partial w_{jk}}$$

$$= (1 - S_{nk}) \frac{\partial (a_{nk})}{\partial w_{jk}}$$

Therefore,

$$\Delta \mathrm{w}_{jk} = (1 - S_{nk}) h_{nj}$$

$$\Delta \mathrm{w}_{ij} = \frac{-\eta \partial E_n}{\partial w_{ij}}$$

$$\frac{\partial \log(S_{nk})}{\partial w_{ij}} = \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial w_{ij}}$$

$$= \frac{1}{S_{nk}} \frac{\partial f(a_{nk})}{\partial (a_{nk})} \frac{\partial (a_{nk})}{\partial w_{ij}}$$

$$= (1 - S_{nk}) \frac{\partial (a_{nk})}{\partial w_{ij}}$$

$$= (1 - S_{nk}) \frac{\partial (w_{jk}. h_{nj})}{\partial w_{ij}}$$

$$= (1 - S_{nk})\, w_{jk} \frac{\partial h_{nj}}{\partial w_{ij}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \text{(1)}$$

$$\frac{\partial h_{nj}}{\partial w_{ij}} = \frac{\partial g(a_{nj})}{\partial w_{ij}}$$

$$\frac{\partial h_{nj}}{\partial w_{ij}} = \frac{\partial g(a_{nj})}{\partial a_{nj}} \frac{\partial a_{nj}}{\partial w_{ij}}$$

$$= g(a_{nj})(1 - g(a_{nj})) \frac{\partial (w_{jk} x_{ni})}{\partial w_{ij}}$$

$$\frac{\partial h_{nj}}{\partial w_{ij}} = g\left(a_{nj}\right). \left(1 - g\left(a_{nj}\right)\right). x_{ni} \qquad \ldots\ldots\ldots\ldots\ldots..(2)$$

Put (2) in (1)

$$\Delta \mathrm{w}_{jl} = \underbrace{(1 - S_{nk}). w_{jk}. g(a_{nj}). (1 - g(a_{nj}))}_{\delta_{nl}}. x_{ni}$$

$$\Delta \mathrm{w}_{ij} = \eta \delta_{nj} x_{ni}$$

4 Data and codes

All the data and codes are provided in github repository(`https://github.com/nikhilmahar/CSE671_Asignment_1`).

5 Contribution

Nikhil Mahar, Rohit Kumar and Sudhir Kumar have contributed equally.