Programming Assignment 1

Report submitted to

Indian Institute of Technology, Mandi for the course

of

CS671: Deep Learning and Applications

by

Piyush Verma (B20163)

Yash Bhotmagey (B20173)

INDIAN INSTITUTE OF TECHNOLOGY, MANDI

February 2023

Abstract

The perceptron was constructed as a mathematical model of the biological neuron which later led to the development of various machine learning and deep learning models we see today. It is essentially a linear machine learning algorithm used for binary classification tasks. The model can be further extended to multi-class classification and regression tasks.

In this assignment, we implemented the Perceptron model from scratch for both classification and regression tasks. We trained the models on linearly separable and non linearly separable datasets for the classification problem and univariate and bivariate datasets for the regression problem. Moreover, we tested the models on the test datasets by plotting the outputs, errors, and drawing inferences from the plots.

Contents

Classification Tasks

The goal here is to compute decision boundaries in the feature space that can best classify the data points. The datasets are first divided into training and testing sets with a 70-30 split with respect to each of the classes. A perceptron model is then trained on the training sets for pairs of classes and one-vs-one approach is used to classify each data point.

Dataset 1: Linearly Separable Classes

The dataset contains 2-dimensional linearly separable data with three classes and each class contains 500 data points.

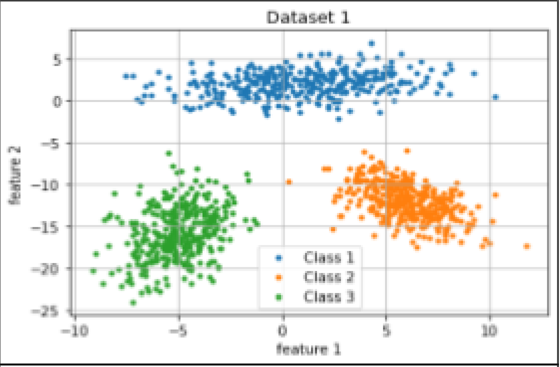

Figure 1 : Scatterplot for Dataset 1

It can clearly be seen that the dataset is linearly separable and we expect to see linear decision boundaries separating each class.

Average Error v/s Epochs

The model is trained on each pair of classes and the average error is computed for every successive epoch.

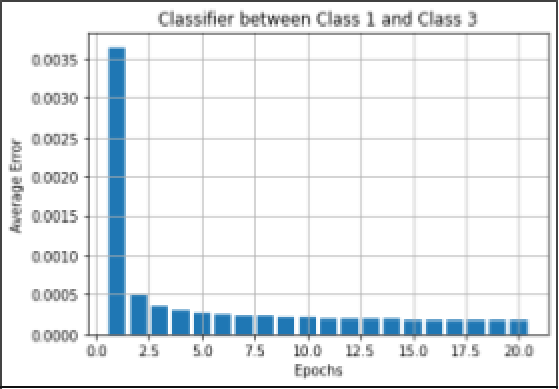

Figure 2: Average error v/s Epochs for Classifier 1-3

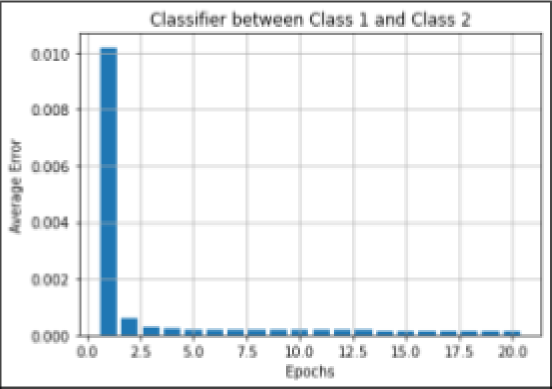

Figure 3: Average Error v/s Epochs for Classifier 1-2

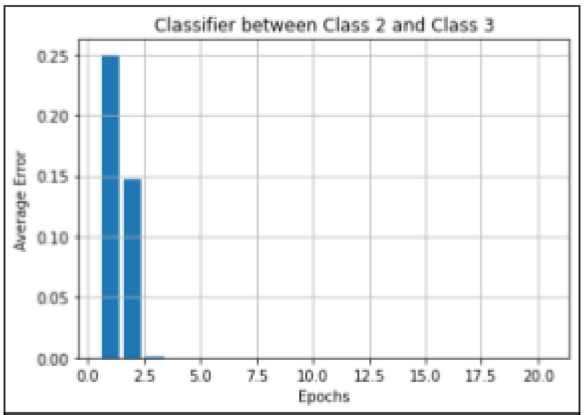

Figure 4: Average error v/s Epochs for Classifier 2-3

The average error decreases with every successive epoch for all three classifiers. The rate of decrease is greater in the initial epochs which later becomes almost stationary. A decreasing learning rate (α = 1 / epoch_number) was used in the model hence the convergence is smooth. It can be concluded that the model is able to find good approximations for the decision boundaries between every pair of classes.

Decision Region Plots

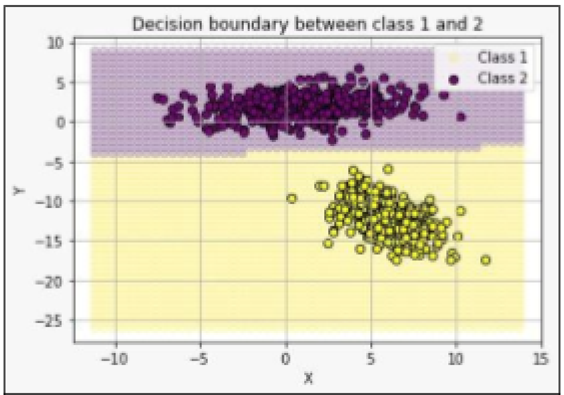

Figure 5: Decision regions for Classifier 1-2

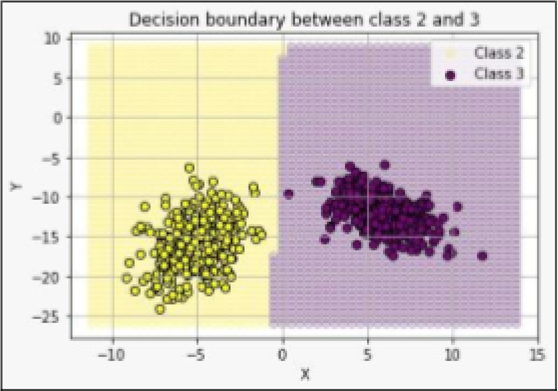

Figure 6: Decision regions for Classifier 2-3

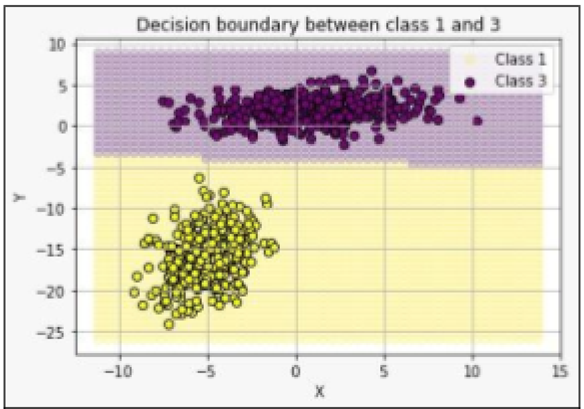

Figure 7: Decision regions for Classifier 1-3

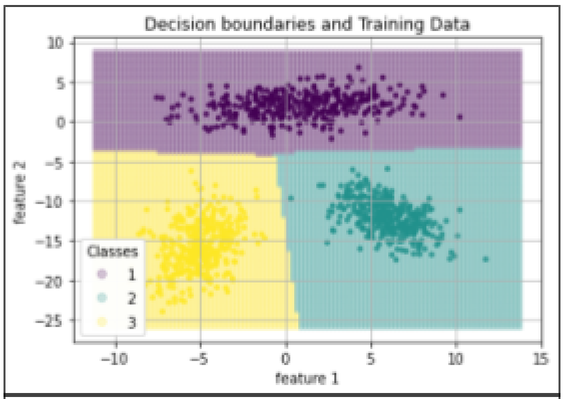

Figure 8: Decision regions computed by the model

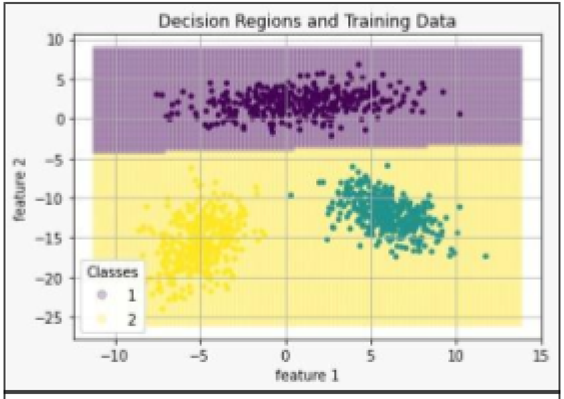

Figure 9: Decision regions computed by the model

To visualize the decision boundaries computed by the model for training dataset, the decision regions are plotted for all the classifiers. The model is correctly able to classify all three classes in the training data and the decision boundaries between the classes are linear in nature. The decision regions appropriately encapsulate the three classes.

Figures 5 and 6 show the decision regions for all the three classes combined. The model correctly estimated the decision regions first as seen in Figure 5. However, when the script is run again with different weight initializations, the model fails to approximate a decision boundary between classes 2 and 3. Depending on the initial weight vectors the model computes different decision regions for the data. This might be due to the model converging to some local minima.

Confusion Matrix and Classification Accuracy

Accuracy score: 1.0

Confusion matrix:

	Predicted Class		
Actual Class	149	0	0
	0	149	0
	0	0	149

The confusion matrix further shows that all data points in classes 1, 2 and 3 are correctly classified for the testing data as well. It can be concluded that the decision boundaries computed by the model are great approximations of actual boundaries for the classes.

Dataset 2: Nonlinearly Separable Classes

The dataset contains 2-dimensional data with three classes that are non linearly separable where classes 1, 2 and 3 contain 300, 500 and 1000 data points respectively.

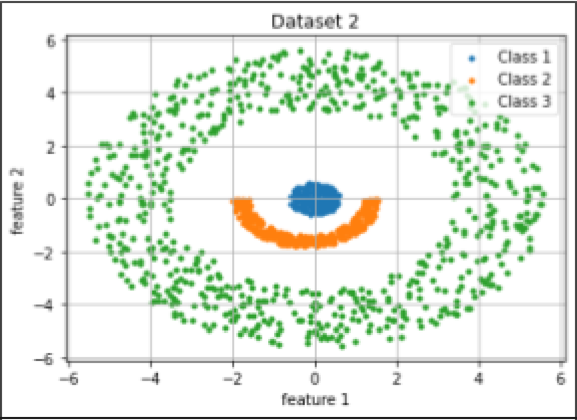

Figure 10: Scatterplot for Dataset 2

Average Error v/s Epochs

The model is trained on each pair of classes and the average error is computed for every successive epoch.

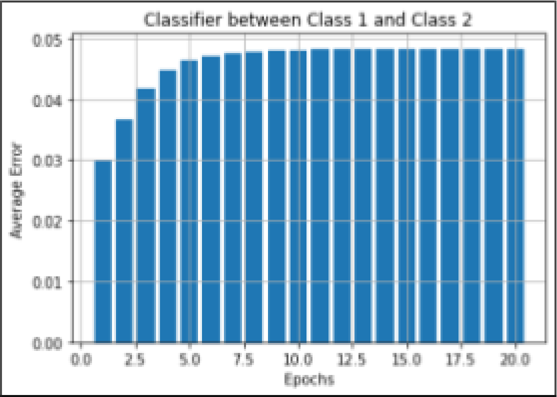

Figure 11: Average Error v/s Epochs for Classifier 1-2

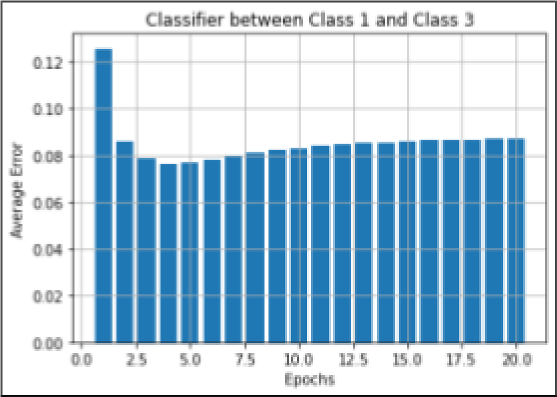

Figure 12: Average Error v/s Epochs for Classifier 1-3

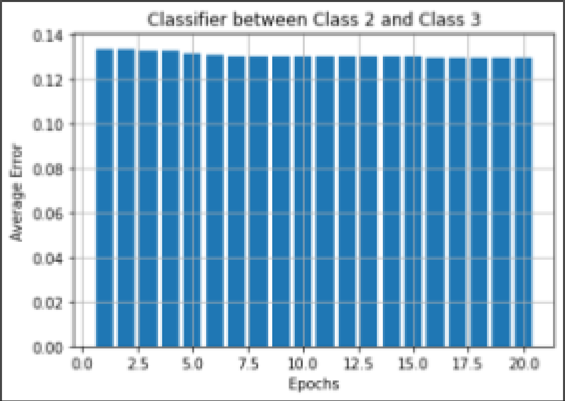

Figure 13: Average Error v/s Epochs for Classifier 2-3

The average error for Classifier 1-2 increases first and then becomes stationary while the average error for Classifier 1-3 decreases first and then becomes stationary and there is almost no change in average error for Classifier 2-3. All three classifiers however show very little change in error after 5 epochs.

Decision Region Plots

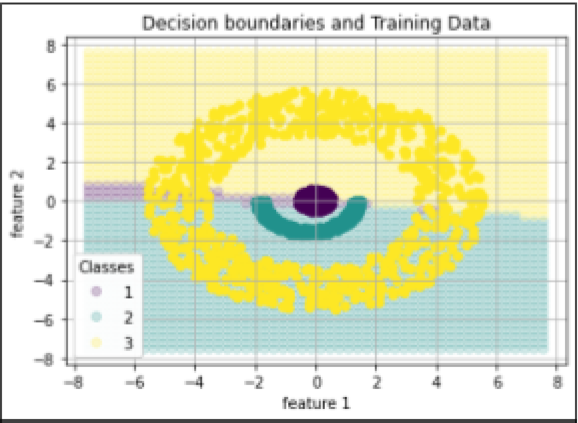

Figure 14: Decision regions computed by the model

As evident from Figure 14, non linear decision boundaries are required to correctly classify classes 1, 2 and 3. The model, consisting only of a single perceptron node, cannot compute a non linear decision boundary. Hence it can be seen that the decision region for class 2 also includes almost half the data points from class 3 and class 1. Similar observations can be made for the decision regions for other classes.

Confusion Matrix and Classification Accuracy

Accuracy score: 0.57222

Confusion matrix:

	Predicted Class		
Actual Class	29	33	28
	16	132	2
	5	147	148

Sensitivity of the model for each class:

1. Class 1: 0.32222
2. Class 2: 0.88
3. Class 3: 0.49333

The decision regions computed by the model contain a lot of data points from different classes and it has no way of computing a perfect decision boundary for the non separable data.

Regression Tasks

The goal here is to estimate functions in the feature space that can best fit the data points. The datasets are first divided into training and testing sets with a 70-30 split. A perceptron model is then trained on the training sets to estimate a polynomial function (linear function in this case since a single perceptron node is used).

Dataset 1: Univariate Data

The dataset contains univariate data i.e. an independent variable and a dependent variable.

Average Error v/s Epochs

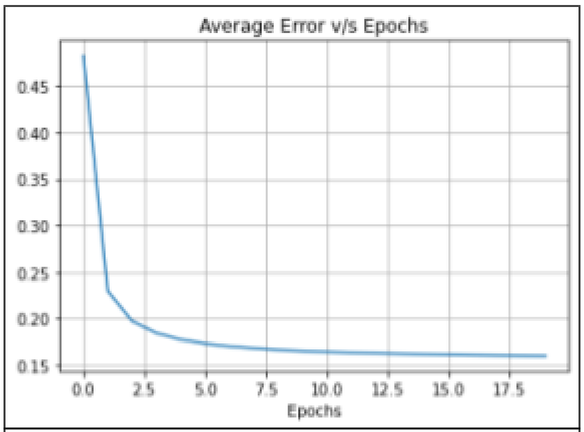

Figure 15: Average error v/s Epochs for Dataset 1

The model is trained on the training dataset and the average error is computed for every successive epoch. The average error of the model decreases with each epoch. The rate of decrease is high at first but it gradually decreases with each epoch. The convergence is smooth since the learning rate parameter also decreases with each epoch (α = 1 / epoch_number). The average error becomes stationary after around 12 epochs. Hence it can be concluded that the model converges to best fit linear function after 12 epochs.

Mean Squared Error

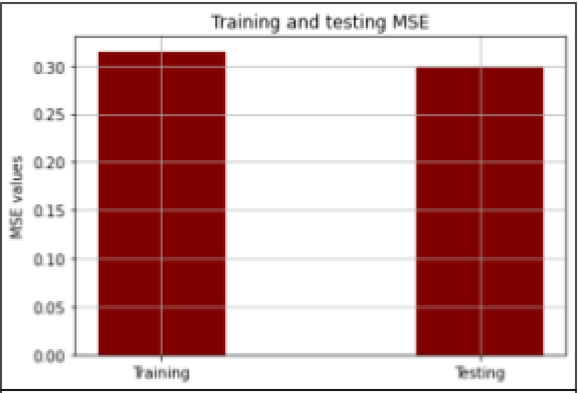

Figure 16: MSE for Training and Testing data

The MSE for both training and testing data is comparable. Hence the model has low variance ie. the model will perform similarly on unseen data from the same distribution. However the mean squared error on both the datasets is still high i.e. the model has high bias.

Model Output and Target Output

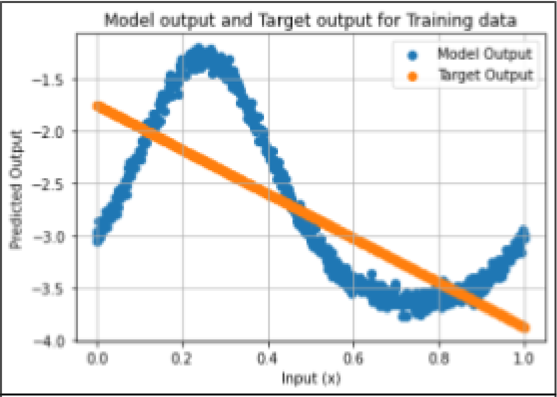

Figure 17: Target and Model outputs for training data

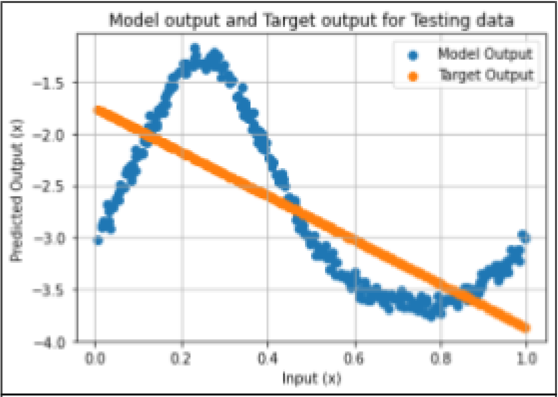

Figure 18: Target and Model outputs for Testing data

Figures 17 and 18 further show that the model has underfit both the datasets. Both the training and test data come from the same distribution. Hence the function learned on the training data shows similar results for testing data. From the plots it is evident that a second or higher order polynomial function will best fit the data. However, the single node perceptron model is only capable of estimating a linear function to best fit the dataset.

Target Output v/s Model Output

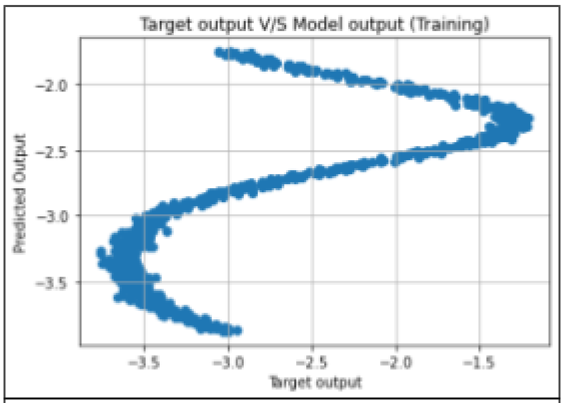

Figure 19: Predicted output v/s Target output

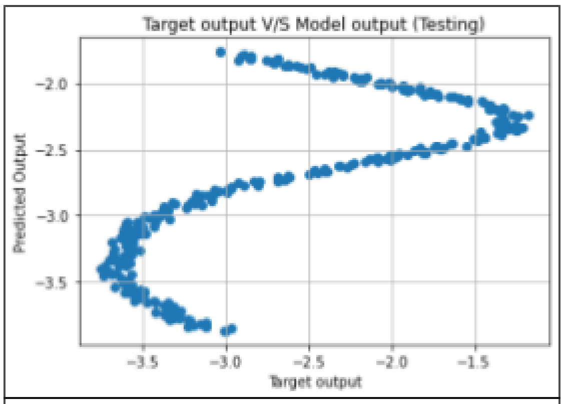

Figure 20: Predicted output v/s Target output for Testing data

The plots show the relationship between target output and the predicted output. A perfect model should display a scatter plot with points scattered around the linear function y = x i.e. equal values for target and predicted variables. This is not the case for the perceptron model as evident from Figures 16 and 17. However, the model does show some linearity in the range (-3.5, -1.5) which can also be seen in figures 14 and 15 where the predicted and the target outputs show similar trends in the range.

Dataset 2: Bivariate Data

The dataset contains bivariate data i.e. two independent variables and a dependent variable.

Average Error v/s Epochs

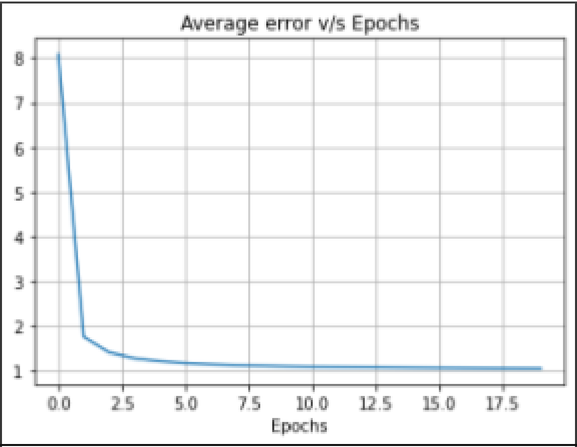

Figure 21: Average error v/s Epochs for Dataset 2

The model is trained on the training dataset and the average error is computed for every successive epoch. The average error of the model decreases with each epoch. The rate of decrease is high at first but it gradually decreases with each epoch. The convergence is smooth since the learning rate parameter also decreases with each epoch (α = 1 / epoch_number). The average error becomes stationary after around 7 epochs. Hence it can be concluded that the model converges to best fit linear function after 7 epochs.

Mean Squared Error

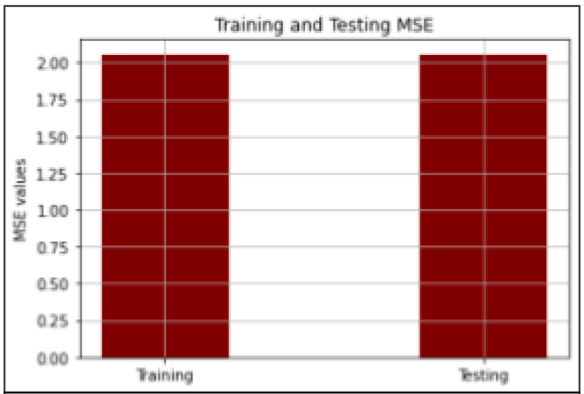

Figure 22: MSE for Training and Testing data

Similar to the univariate case, the MSE for both the training and testing datasets is comparable. Hence the model has low variance ie. the model will perform similarly on unseen data from the same distribution. However the mean squared error on both the datasets is still very high i.e. the model has high bias.

Model Output and Target Output

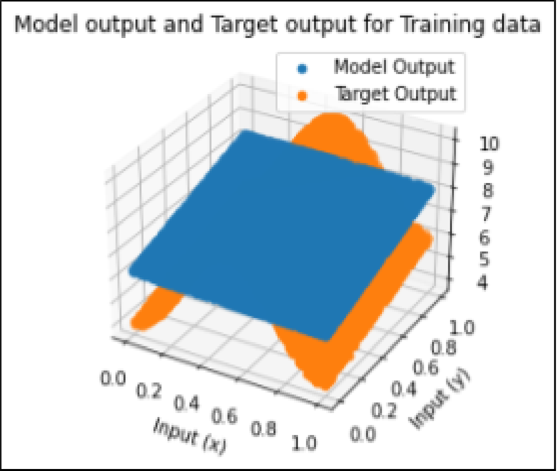

Figure 23: Model and Target outputs for Training data

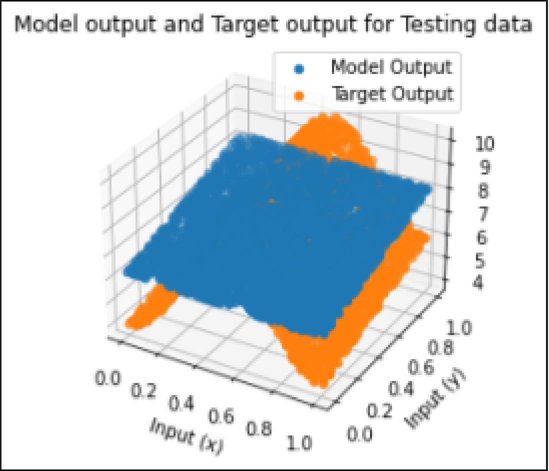

Figure 24: Model and Target outputs for Testing data

Figures 23 and 24 further show that the model has underfit both the datasets. Both the training and test data come from the same distribution. Hence the function learned on the training data shows similar results for testing data. From the plots it is evident that the target output is distributed along a curved surface and a second or higher order polynomial function will best fit the data. However, the single node perceptron model is only capable of estimating a linear plane to best fit the dataset. This will require multiple perceptrons or multiple layers of perceptrons or a combination of both to estimate a function good enough to approximate the data.

Target Output v/s Model Output

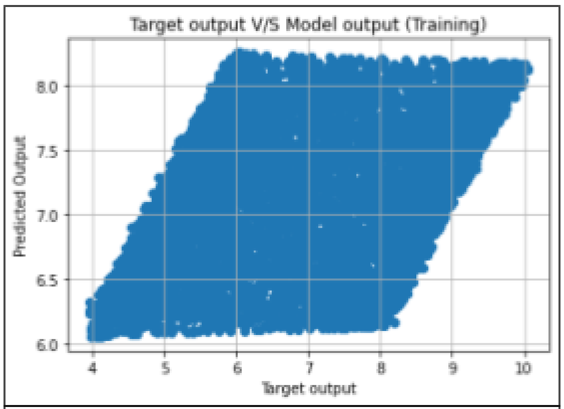	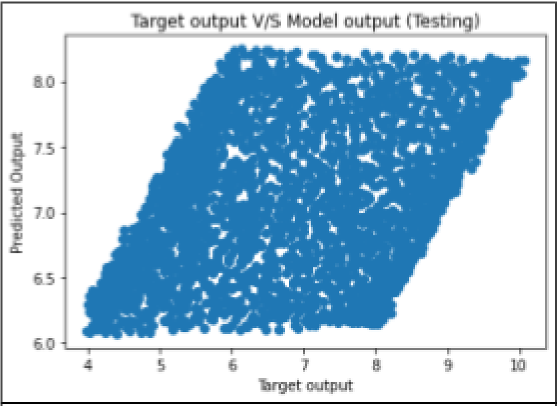
Figure 25: Predicted v/s Target outputs for Training data	Figure 26: Predicted v/s Target outputs for Testing data

Correlation between Predicted and Target values (training): 0.37352

Correlation between Predicted and Target values (testing): 0.37941

The plots show the relationship between target output and the predicted output. Similar to the univariate case, the target values do not match the predicted values. Although they do show some correlation as computed above which can also be seen in figures 23 and 24 where the predicted and the target outputs show similar trends. However the predicted outposts are not good approximations of the actual outputs.