CS671: Deep Learning and Applications
Programming Assignment 1

RAUNAV GHOSH(M.Tech-CSP)
T22104

PRASHANT DHANANJAY KULKARNI(M.Tech-CSE)
T22060

SACHIN BAHULEYAN(M.Tech-CSE)
T22058

Contents

1 Introduction

Supervised learning using perceptron

A perceptron is a basic building block of artificial neural networks, modelled after the functioning of biological neurons. It receives input values and produces an output signal based on a weighted sum of the inputs.
The perceptron algorithm is used for supervised learning in machine learning. The algorithm adjusts the weights of the inputs to the perceptron based on the error between the predicted output and the actual output, in order to improve the accuracy of the model.
The perceptron is used for classification and regression problems where it can identify patterns and make predictions based on the data it receives. Figure 1 shows a general model of a perceptron.

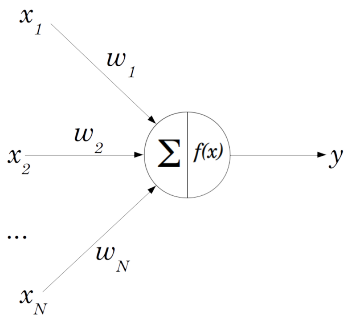


Figure 1: A perceptron

Here for the input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, weight vectors $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$ and the activation function $f(.)$, the output of the perceptron is given as

$$s = f(\mathbf{w}^T\mathbf{x} + w_0)$$

The activation function $s = f(.)$ can be selected based on the choice of the task. For *classification task*, it is common to use *sigmoid activation* functions(logistic or hyperbolic tangent) for the perceptron output. In case of *regression task*, a *linear function* is chosen as activation function for the perceptron output.

2 Classification task

2.1 Problem Statement

Given Datasets:
Dataset 1: Linearly separable classes: 3 classes, 2-dimensional linearly separable data is given. Each class has 500 data points.
Dataset 2: Nonlinearly separable classes: 2-dimensional data of 2 or 3 classes that are nonlinearly separable. The number of examples in each class and their order is given at the beginning of each file. Divide the data from each class into training, and test data. From each class, train, and test split should be 70% and 30% respectively.
Model - Perceptron with sigmoidal activation function for each of the data-sets. Use one-against-one approach for 3-class classification. Implement gradient descent method for perceptron learning algorithm.

2.2 Introduction

Classification is a fundamental task in machine learning that involves predicting a class or category for a given input. In classification, a model is trained on a labeled data-set where each data point can be used to make accurate predictions on unseen examples.
A single neuron(perceptron) can learn a linear function that can classify two classes. A classification perceptron model is built by leaning a *discriminant function(hyperplane)* that separates the two classes. This is illustrated in Figure 2, where two classes are separated by the hyperplane.

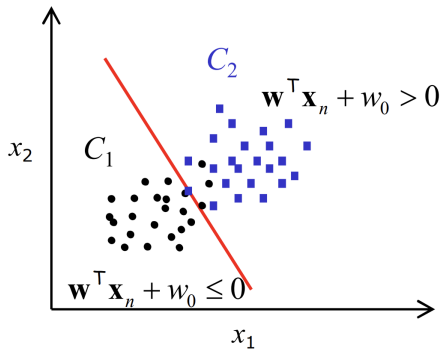


Figure 2: Separation of 2 classes using an hyperplane(straight line)

The hyperplane is learned by solving an optimization problem which places the plane such that the error of prediction due to the hyperplane is minimum. This is achieved using gradient descent method which minimises the loss function per iteration.

2.3 Methodology

To learn the weights of the discriminant function, we minimise the *loss function* which is given as

$$E_n = \frac{1}{2}(y_n - s_n)^2 \tag{1}$$

where y_n is the actual label assigned to n^{th} training example and s_n is the signal predicted by the perceptron. E_n is called the instantaneous error for the n^{th} training example. We find a solution for the problem using **Gradient Descent Algorithm**.

Gradient descent method allows us to calculate the local minima of a function to estimate the learning parameter.
Let $\Delta\mathbf{w}$ be the change in weight per iteration. Then the change in weight vector is expressed as

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w} \tag{2}$$

It is directly proportional to the decrease in the change of the instantaneous function, i.e.,

$$\Delta\mathbf{w} \propto -\frac{\partial E_n}{\partial \mathbf{w}} \implies \Delta\mathbf{w} = -\eta\frac{\partial E_n}{\partial \mathbf{w}} \tag{3}$$

Further expanding, we get can write the expression for $\Delta\mathbf{w}$ as,

$$\Delta\mathbf{w} = \eta\delta^{\circ}\hat{\mathbf{x}} \tag{4}$$

where $\hat{\mathbf{x}}$ is the augmented input vector and

$$\delta^{\circ} = (y_n - s_n)\frac{df(a_n)}{da_n}$$

The activation function used for classification is *sigmoid function*. For *logistic sigmoid function*, we choose the labels $\{0, 1\}$ and for *tanh* we choose labels $\{-1, +1\}$.
The algorithm can be summarised as:

1. Initialised the weight parameter w with random values within input range.
2. Choose a training example $\mathbf{x}_n$.
3. Compute output of the neuron for $\mathbf{x}_n$

   $$s_n = f(a_n) = \frac{1}{1 + \exp -a_n}$$

   or

   $$s_n = f(a_n) = \tanh a_n$$

   where

   $$a_n = \mathbf{w}^T\hat{\mathbf{x}}_\mathbf{n}$$

   and $\hat{\mathbf{x}}_\mathbf{n} = [1, \mathbf{x}_\mathbf{n}]$

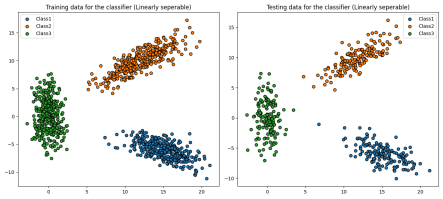
4. Compute instantaneous error $E_n = \frac{1}{2}(y_n - s_n)^2$
5. Compute change in weights $\Delta \mathbf{w} = \eta \delta^{\circ} \hat{\mathbf{x}}$
6. Update weights $\mathbf{w} = \mathbf{w} + \Delta \mathbf{w}$
7. Repeat steps 2 through 5 for on *epoch*.
8. Compute the average error

$$E_{av} = \frac{1}{N} \sum_{n=1}^{N} E_n(\mathbf{w})$$

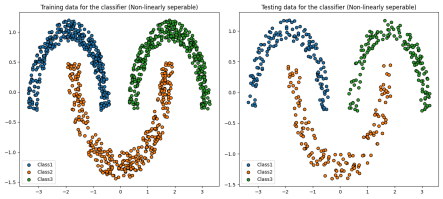
9. Repeat steps 2 to 8 till convergence.

2.4 Results

For a collection of three class datasets, we use one-one approach, i.e., we build three perceptrons(class 1 vs class2, class2 vs class3 and class 3 vs class1) to discriminate between two classes each. The majority class predicted by the perceptrons are chosen to be the prediction of the classifier.
The classifiers were build on the both linearly separable and non-linearly separable data. The datasets can be observed in Figure 3a and Figure 3b where the total datasets were split in 7:3 for training and testing.



(a) Linearly separable data for classification



(b) Non-linearly separable data for classification

Figure 3: Datasets for building classifiers

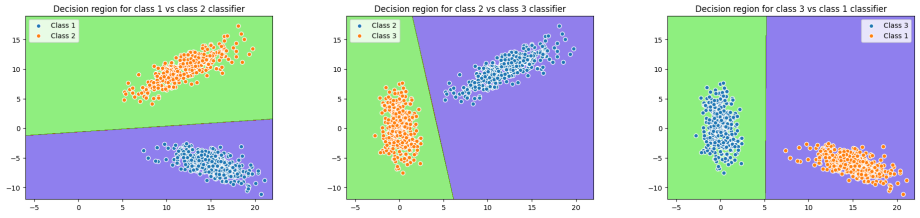


Figure 4: Decision region learnt by each perceptron(Linearly separable data)

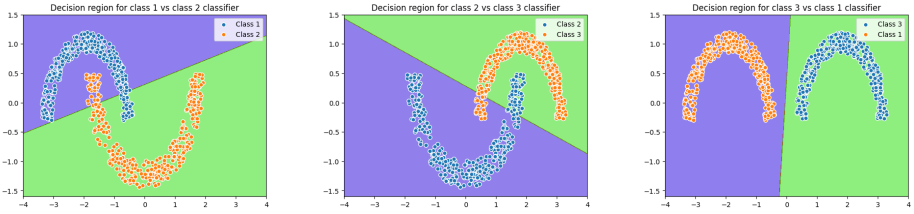


Figure 5: Decision region learnt by each perceptron(Non-linearly separable data)

The decision region learned by each of the perceptron is given in Figure 4 for linearly separable data and Figure 5 for non-linearly separable data. The overall decision region learnt by the classifiers is given in Figure 6 for linearly separable data and Figure 7 for non-linearly separable data.

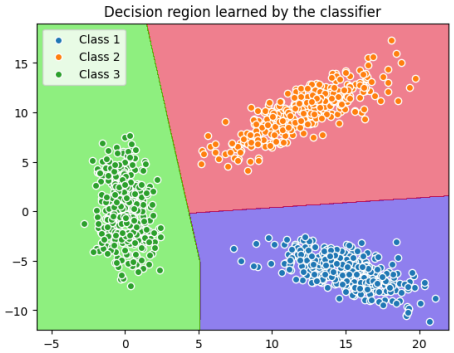


Figure 6: Decision region learnt by the classifier(for linearly separable data)

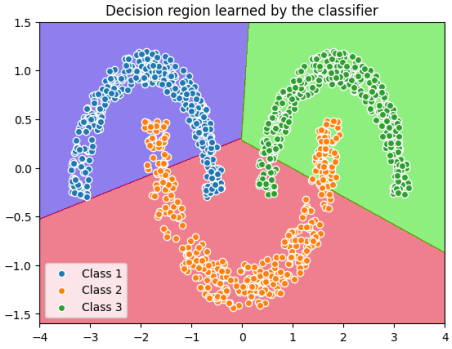
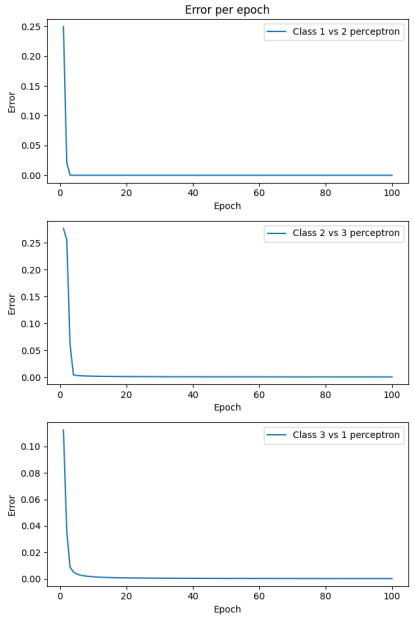


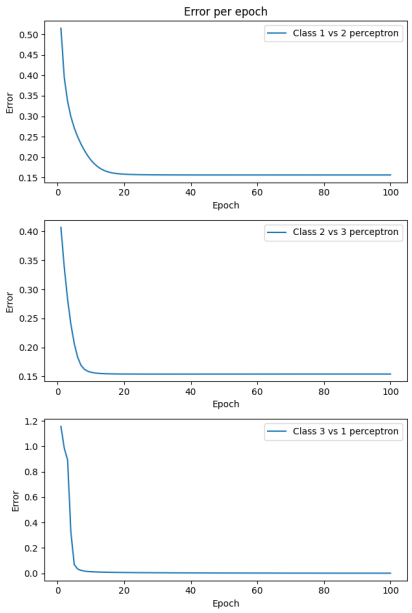
Figure 7: Decision region learnt by the classifier(for non-linearly separable data)

We have experimented with both *logistic sigmoid* and tanh activation function

and obtained similar results. We decided to use *logistic sigmoid* for linearly separable data and tanh for non-linearly separable data for best decision regions with a learning rate of 0.05 and 0.002 respectively.
In case of linearly separable data, we have observed that the loss function always reaches convergence for any sigmoid function. The average error per epoch while learning classes for each of the perceptron is given in Figure 8a and Figure 8b.



(a) Average error per epoch while learning (linearly separable data)



(b) Average error per epoch while learning (non-linearly separable data)

Figure 8: Squared error per epoch while training

2.5 Observations

The models are summarised as below:
Linearly separable data
Learning rate : 0.05
Confusion matrix on testing data

150	0	0
0	150	0
0	0	150

Accuracy : 100%

This shows that our model identifies each class perfectly for linearly separable type data.

Non-linearly separable data
Learning rate : 0.002
Confusion matrix on testing data

131	19	0
15	118	17
0	13	137

Accuracy : 85.77%

This shows a best fit of the classifier to the non-linearly separable data gives us an accuracy of 85.77%. This measurements can improve if a non-linear discriminant function can be approximated for separating each class.

3 Regression task

3.1 Problem Statement

Given Datasets:
Dataset 1: 1-dimensional (Univariate) input data
Dataset 2: 2-dimensional (Bivariate) input data
Divide the data into training and test data. Train and test split should be 70% and 30% respectively.
Model - Perceptron with linear activation function for each of the data-sets. Implement gradient descent method for perceptron learning algorithm.

3.2 Introduction

Regression is a type of supervised machine learning algorithm used to predict a continuous value based on input features. It involves finding a relationship between dependent and independent variables using a given training data-set. The dependent variables are all called the *target* and the independent variables are called the *input*. The goal is to build a model that can accurately predict the value of the output variable for new data points.

The main difference between regression and classification is the type of output

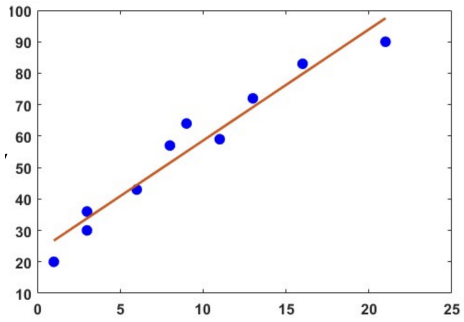


Figure 9: Approximation of data points with straight line

variable they predict. While regression predicts a continuous output variable, classification predicts a discrete output variable, typically a label or a category. A single neuron(perceptron) can learn a linear function that can predict the output for the new data-points. Similar to classification, the hyperplane that approximates the data-points in training data is linear in nature. Unlike classification, the perceptron uses a *linear activation function* at the output.

3.3 Methodology

Similar to the case of classification, we learn the weight parameters using **Gradient descent algorithm**. Unlike for the classification task, the choice of activation function is *linear*. The algorithm can be summarised as:

1. Initialised the weight parameter w with random values within input range.
2. Choose a training example $\mathbf{x}_n$.
3. Compute output of the neuron for

$$s_n = f(a_n) = a_n$$

   where

$$a_n = \mathbf{w}^T \hat{\mathbf{x}}_\mathbf{n}$$

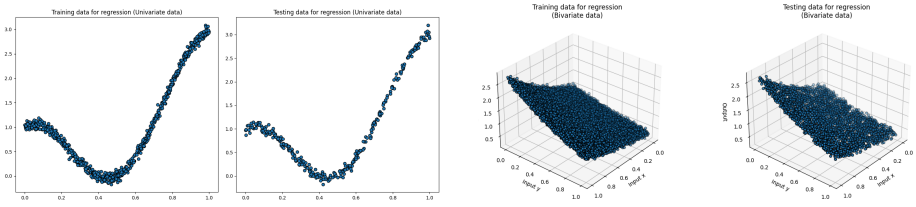
4. Compute instantaneous error $E_n = \frac{1}{2}(y_n - s_n)^2$
5. Compute change in weights $\Delta\mathbf{w} = \eta\delta^\circ\hat{\mathbf{x}}$
6. Update weights $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
7. Repeat steps 2 through 5 for on *epoch*.
8. Compute the average error

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E_n(\mathbf{w})$$

9. Repeat steps 2 to 8 till convergence.

3.4 Results

The dataset given for regression task was split into 7:3 ratio for training and testing purposes. This can be visualised in Figure 10a and 10b. Based on

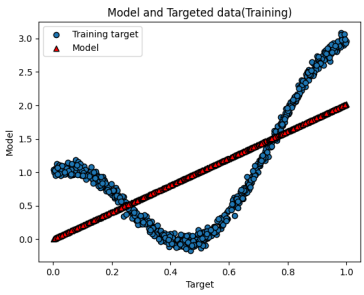


(a) Univariate Dataset (b) Bivariate Dataset

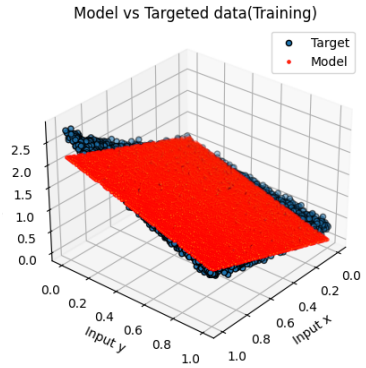
Figure 10: Dataset given for regression task

the data, perceptron learning algorithm was implemented to obtain a linear function to get a regression model. This is visualised in Figure 11a and 11b for univariate and bivariate data repectively. The error per epoch for the gradient descent algorithm can be observed in Figure 12a and 12b while training both univarite and bivariate data models.

The model on the test data can be observed in Figure 13.

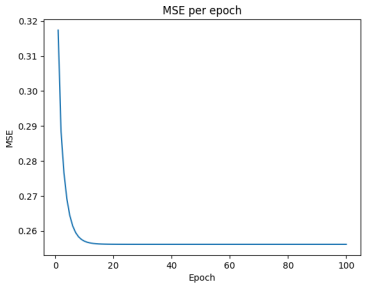


(a) Regression model for Univariate data

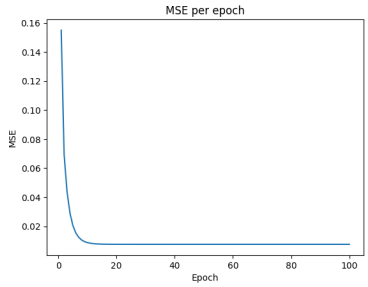


(b) Regression model for Bivariate data

Figure 11: Learnt regression model using perceptron



(a) Univarite data model



(b) Bivariate data model

Figure 12: Minimization of loss function over epochs

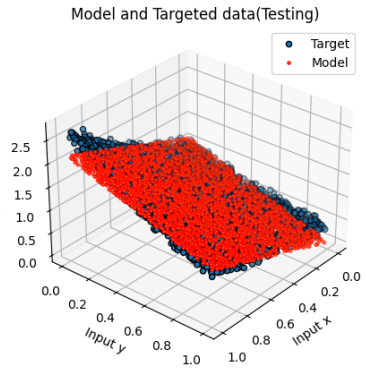
3.5 Observations

The regression models are summarised below:
Univariate data model
Learning rate : 0.005

Bivariate data model
Learning rate : 0.0005

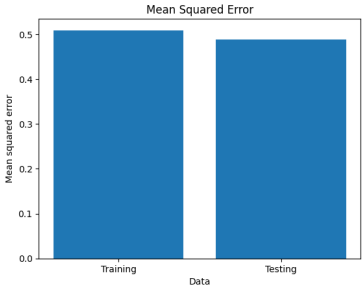
Figure 15 and 16 illustrate the values that model predicts with 0 error. The points intersecting the matched-line(red line) shows the output of the model that matches with the target output.

(a) Univariate data model

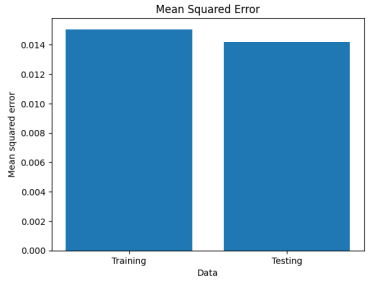


(b) Bivariate data model

Figure 13: Regression models on the test data

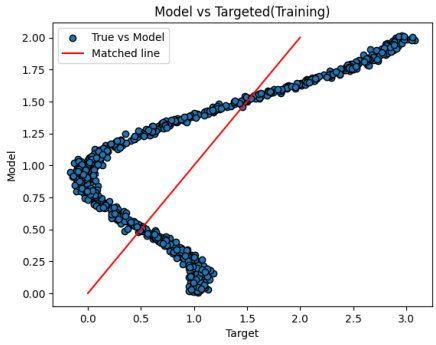


(a) Univariate data model

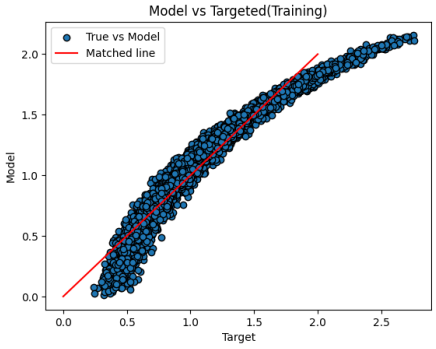


(b) Bivariate data model

Figure 14: Regression models mean squared error

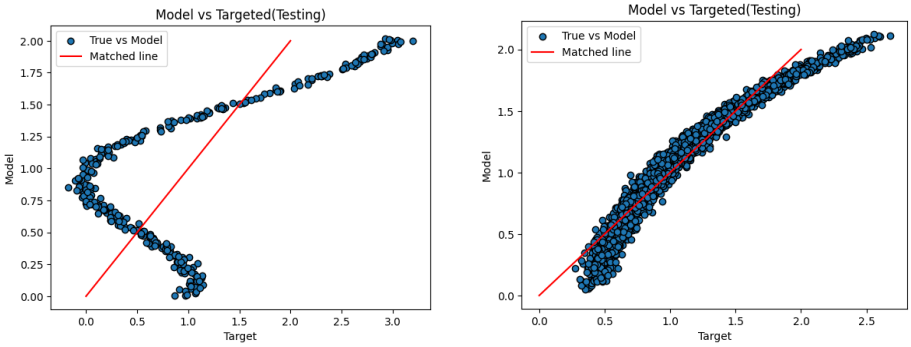


(a) Univariate data model



(b) Bivariate data model

Figure 15: Regression models vs target values(Training data)



(a) Univariate data model (b) Bivariate data model

Figure 16: Regression models vs target values(Testing data)