CSE641 - Deep Learning
Assignment 1
ReadME

Submitted by:
Kashish Narang (MT19026)
Chirag Chawla (MT19089)
Saumya Jain (MT19098)

Question 1 :

Methodology :

We have implemented PTA (Perceptron Training Algorithm). Here are the steps :

1. Initialize the weights and bias to all 0's.
2. Compute the net Input (Z) using Z = $W^T . X_i$
3. Multiply Z with desired output : Z = Y_i * Z
4. If (Z <= 0), there is an error, so we will update the weights by following equation :
   W = W + $Y_i . X_i$
5. Else, we continue to the next data point.
6. Repeat step 2-5 till we got no Error in all data points.

Helper Functions :

1. PTA() : It takes data points and labels as input and returns the final weights and bias for which it is converging.
2. plot_data() : It takes weights, bias, data points, label and title, then prints the decision boundary along with data points represented in scattered form. It is generally used to plot 2D data points and weights, bias.
3. plot_not_data() : It takes weights, bias,not data points, not label and title, then prints the decision boundary along with data points represented in scattered form. It is basically used to plot 1D data points and weights, bias.

Preprocessing Steps :

1. Creating the data points for each operation.
2. Creating all the helper functions.

Question 2:

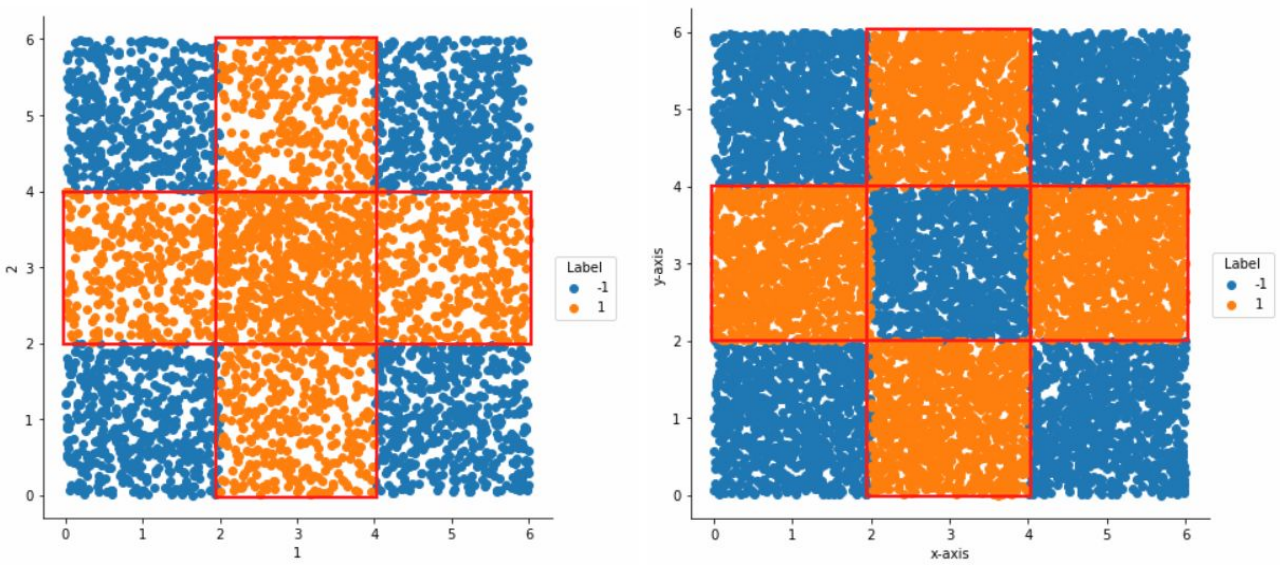
Methodology :

We have implemented the Madaline training algorithm for Multi Layer Perceptron.

1. We will apply the changes only if desired output is not equal to the predicted output.
2. If the above condition holds, then we will find the culprit neuron.
3. We start with neuron having the smallest net input value, flip its output then compute the final output.
4. If now desired value will be equal to the predicted value, which means that particular neuron is culprit and we will apply Adaline on this Neuron only.
5. If not, then find the next neuron with the next smallest net input value and repeat the step 3-4.
6. Do these steps for all data points.
7. Repeat 1-6 steps for multiple epochs.

Preprocessing Steps :

1. First we have created a dataset for both f1 and f2. F1 has 4000 data points and F2 has 9000 data points using random distribution.



2. Creating the helper functions to compute the final output of the network.

Question 3:

Methodology :

We have implemented a Single Neuron Neural Network with generalised delta weight update rule and different activation functions such as sigmoid, tanh, sine. First task is to Initialize the weights and bias to all 0's. For a certain number of epochs or early stopping if weights do not change for 5 consecutive epochs, we run the training of the neural network with all the training data. Training part include the linear summation of weights multiplied by single training example, applying activation function and getting the prediction, calculating the error based on true output, update the weights if misclassification turns out based on delta rule.Once the algorithm converges(which did not happen in this case), we predicted the output of unknown test set created separately.

Preprocessing Steps :

1. Creating the training and testing dataset

```
#Data for training
X_train = np.array([[0],[2],[4],[6],[8],[10]])
y_train = np.array([0, 1, 0, 1, 0, 1])

#Data for testing
X_test = np.array([[-2],[12],[14],[16]])
y_test = np.array([1, 0, 1, 0])
```

2. Creating all the helper functions and class 'SingleNeuronNN'.

```
#Single layer neural network
class SingleNeuronNN:
    def __init__(self,wei,ep,lr,act):
        self.wei = wei
        self.ep = ep
        self.lr = lr
        self.act = act
```

Contribution of Members:

We collaborated with each other on all the questions and listed our primary responsibility below.

Saumya Jain	:	Question 1
Chirag Chawla	:	Question 2
Kashish Narang	:	Question 3