Homework – 2
Name: Chetanraj Kadam
SUID: 250256631

Problem Statement:

In this task we are going to modify backpropagation algorithm such that it will have constraint on misclassification rates. Such that we want to lower False Negative (FN) rate below certain threshold and at the same time we have to minimize False Positive (FP) rate as possible.

Implementation and Code details:

For this task I selected “Pima Indians Diabetes Database” from UCI datasets. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. In this dataset there 8 feature values and 1 class variable (0 or 1). If person has diabetes then class label will be 1 otherwise its 0.

First dataset loaded from csv file and then divided into X as attributes and Y as class variable. Again this dataset split into test (80%) and train (20%).

```
dataset = np.loadtxt("pima-indians-diabetes.csv", delimiter=",")
X = dataset[:,0:8]
Y = dataset[:,8]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,random_state=4)
```

Neural Network 1:

Now we have to build Neural network with class variables holding input, weights and outputs. And functions for forward propagation of weights and backpropagation of error.

```
# First Neural Network with normal backpropagation implementation

class NeuralNetwork(object):

    def __init__(self):
        self.input = X_train.shape[1]
        self.learning = 0.04
        self.hidden = (X_train.shape[1]*2)
        self.output = 1

        self.W1 = np.random.randn(self.input, self.hidden)
        self.W2 = np.random.randn(self.hidden, self.output)
```

```
    def backpropgation(self,X, Y, O):
        self.o_error = Y - O
        self.o_delta = self.o_error*self.sigmoid_derivative(O)

        self.z2_error = self.o_delta.dot(self.W2.T)
        self.z2_delta = self.z2_error*self.sigmoid_derivative(self.z2)

        self.W1 += self.learning*X.T.dot(self.z2_delta)
        self.W2 += self.learning*self.z2.T.dot(self.o_delta)
```

Backpropagation:

Basic idea of this we calculate error from difference between desired output value and obtained output value. From this we calculate required nudges in weights we want to happen in previous layer. Then this process is recursively apply in previous layers to decrease error and making output of network to look more like actual decision we want to make. Now in input space we want to find inputs such that error is minimum and we have to move in that downhill direction. This is computed by Gradient Decent.

Now we train this network with training set with repetition of 250 times.

Neural Network 2:

Now coming to the task, we here want to look for constraint of misclassification such that this network should not predict more FN values i.e person who has diabetes should not be predicted as negative. And at the same time we want to minimize FP values i.e network should not classify as diabetes for person who don't have it.

To achieve this we have to modify our backpropagation algorithm. So here we are going to change error function such that it will penalize more for FN and normal for other values. By doing this weights and biases will be adjusted such that network will be more careful for FN values. But this also has drawback that we will see in results. So we update error function such that if output we got is less than 0.5 then we will use normal error function but when it is greater than 0.5 then we will take difference between 1 and that output because these are closer to be FN values. So this error function will penalize network more and update weights more for FN.

Output Results:

We will first test our first neural network with test dataset and obtain respective confusion matrix and FP rate and FN rate.

```
# First Neural Network with normal backpropagation

NN = NeuralNetwork()
y_pred1  = NN.accuracy(X_test)
y_pred1 = [round(x[0]) for x in y_pred1 ]

cm = confusion_matrix(Y_test,y_pred1)
FPR = cm[0,1]/(cm[0,1] + cm[0,0])
FNR = cm[1,0]/(cm[1,0] + cm[1,1])

print("Output with normal implementation of backpropagation:")
print("\nConfusion Matrix:\n")
print(cm)

print("\nFalse Positive Rate (FPR) :- %.2f%%" % (FPR * 100))
print("False Negative Rate (FNR) :- %.2f%%" % (FNR * 100))
```

Same way confusion matrix is calculated for Neural Network 2 model.
So after running this we will get following outputs:

- **Output 1:**

For Neural Network1

```
Output with normal implementation of backpropagation:

acc: 79.87%

Confusion Matrix:

[[95  7]
 [24 28]]

False Positive Rate (FPR) :- 6.86%
False Negative Rate (FNR) :- 46.15%
```

For Neural Network2 (With updated backpropagation and keeping FN rate below 10%)

```
Output after modification of backpropagation:

acc: 60.39%

Confusion Matrix:

[[45 57]
 [ 4 48]]

False Positive Rate (FPR) :- 55.88%
False Negative Rate (FNR) :- 7.69%
```

In this we can see FN rate with updated neural network is lowered from 46.15% to 7.69%. But at the same time accuracy is lowered from 79.87% to 60.39% and FP rate is massively increased from 6.86% to 55.88%.

- **Output 2:**

Now running both the models on test data and keeping FN below 5% we get following outputs.

For Neural Network1

```
Output with normal implementation of backpropagation:

acc: 77.27%

Confusion Matrix:

[[93  9]
 [26 26]]

False Positive Rate (FPR) :- 8.82%
False Negative Rate (FNR) :- 50.00%
```

For Neural Network2 (With updated backpropagation and keeping FN rate below 5%)

```
Output after modification of backpropagation:

acc: 55.19%

Confusion Matrix:

[[34 68]
 [ 1 51]]

False Positive Rate (FPR) :- 66.67%
False Negative Rate (FNR) :- 1.92%
```

In this result we can see that by keeping FN rate below 5% our accuracy lowered from 77.27% to 55.19% and at the same time FP rate is massively increased from 8.82% to 66.67%.

Conclusions:

- From the results obtained in different scenarios and by comparing results of the 2 neural network models we can conclude that by updating error function in backpropagation algorithm to penalize network for FN values we can achieve FN below certain level.
- Another conclusion we can draw is that in the process of lowering FN below some threshold our accuracy of network model is decreased by some margin every time. And another interesting conclusion is that FP rate will increase by large margin always when you are going to update error function to lower FN rate.

References:

- http://archive.ics.uci.edu/ml/datasets/diabetes
- https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
- https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6