**Practical Number 9**

**Aim:** Create a Neural Networks in python use the scikit-learn's built-in dataset, Plot the actual and predicted values graph..

**Software Used :** Pycharm Community Edition 2023.1, Python(3.9.12)

**Theory:**

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of [machine learning](https://www.ibm.com/topics/machine-learning) and are at the heart of [deep learning](https://www.ibm.com/in-en/topics/deep-learning) algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

**Code:**

import numpy as np

import matplotlib.pyplot as plt

class Perceptron:

def \_\_init\_\_(self, num\_inputs, num\_outputs):

self.weights = np.random.rand(num\_inputs, num\_outputs)

self.bias = np.random.rand(1, num\_outputs)

def sigmoid(self, x):

return 1 / (1 + np.exp(-x))

def predict(self, inputs):

return self.sigmoid(np.dot(inputs, self.weights) + self.bias)

def train(self, inputs, targets, num\_epochs, learning\_rate):

for i in range(num\_epochs):

predictions = self.predict(inputs)

error = targets - predictions

updates = learning\_rate \* np.dot(inputs.T, error)

self.weights += updates

self.bias += learning\_rate \* error.sum(axis=0)

inputs = np.array([[1.1, 2.2, 3.3, 4.4, 5.5],

[6.6, 7.7, 8.8, 9.9, 8.8],

[7.7, 6.6, 5.5, 4.4, 3.3],

[2.2, 3.3, 4.4, 5.5, 6.6],

[9.9, 8.8, 7.7, 6.6, 5.5],

[3.3, 4.4, 5.5, 6.6, 7.7],

[8.8, 9.9, 8.8, 7.7, 6.6],

[4.4, 5.5, 6.6, 7.7, 8.8],

[2.2, 3.3, 4.4, 5.5, 6.6],

[7.7, 6.6, 5.5, 4.4, 3.3]])

targets = np.array([[0, 1],

[1, 0],

[1, 0],

[0, 1],

[0, 1],

[1, 0],

[0, 1],

[1, 0],

[0, 1],

[1, 0]])

perceptron = Perceptron(5, 2)

perceptron.train(inputs, targets, 1000, 0.01)

predictions = perceptron.predict(inputs)

print("Predictions:")

print(predictions)

# Plot the actual value, predicted value, difference between actual value and predicted value, and 5 largest differences

fig, axs = plt.subplots(2, 2)

axs[0, 0].plot(targets)

axs[0, 0].set\_title("Actual Values")

axs[0, 1].plot(predictions)

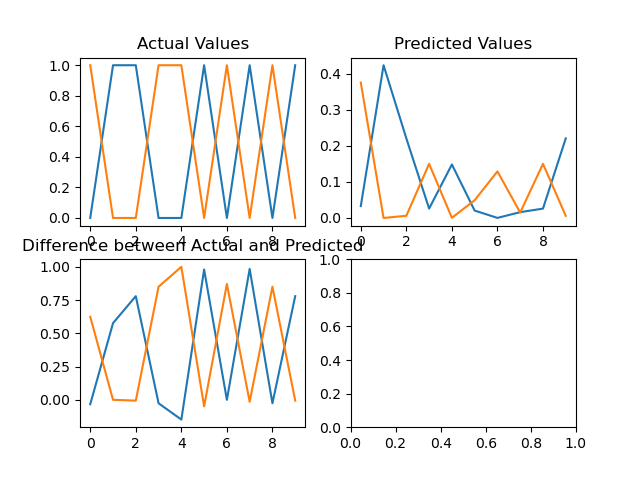
axs[0, 1].set\_title("Predicted Values")

axs[1, 0].plot(targets - predictions)

axs[1, 0].set\_title("Difference between Actual and Predicted")

plt.show()

**Results:**

****

Predictions:

[[3.29465292e-02 3.75802219e-01]

[4.23775173e-01 5.12091317e-05]

[2.20838724e-01 6.11238666e-03]

[2.60243678e-02 1.50103511e-01]

[1.48456884e-01 5.28955518e-04]

[2.05257037e-02 4.92578295e-02]

[2.28467604e-04 1.29239394e-01]

[1.61695593e-02 1.49709462e-02]

[2.60243678e-02 1.50103511e-01]

[2.20838724e-01 6.11238666e-03]]

**Conclusion** : In this practical, we have successfully studied and implemented a Neural Networks in python use the scikit-learn's built-in dataset, Plot the actual and predicted values graph..