Deep Dodhiwala | 2018140016 | BE-IT

Experiment 3 - Multicategory Single Layered Classifier

Aim:

To implement RDPTA and MCPTA training algorithms for single layered neural networks

Problem Statement:

Implement RDPTA (R-Category Discrete Perceptron Training Algorithm) and MCPTA (Multi-Category Perceptron Training Algorithm) for the given problem:

$$\text{Class} = 1 \quad \text{for } \mathbf{x} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^t, \quad \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^t$$

$$\text{Class} = 2 \quad \text{for } \mathbf{x} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^t, \quad \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^t$$

$$\text{Class} = 3 \quad \text{for } \mathbf{x} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^t, \quad \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^t$$

$$\text{Class} = 4 \quad \text{for } \mathbf{x} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^t, \quad \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^t$$

Tool/ Language:

Python, numpy

RCPTA Code:

```
class RCPTA:

    def_init___(self):
        # 4 x 4 weight matrix of random values self.W =
        np.zeros((4,4))
def predict(self, X_test):
        res = np.dot(X_test, self.W) return
        np.where(res >= 0, 1, 0)
def fit(self, X_train, d_train, learning_rate, epochs):

        print("Initial Weights: ", self.W)
```

```
for k in range(epochs):
        print("########################### Epoch ", k, "
###########################")
for i in range(0,8): E = 0 print("######## X ", i, "
    #########") print("X_train: ", X_train[i])
    for j in range(0,4):

        print("Starting Weights: ", str(self.W[j])) z =

        np.dot(self.W[j].T, X_train[i])


        print("Z: ", z) y_net =
        uni_activation(z)

        print("Y: ", str(y_net))

        e = d_train[i][j] - y_net print("Error:
        ", E)

        E = E + 0.5*(e)**2

        self.W[j] = self.W[j] + learning_rate * e * X_train[i] print("New

        Weights: ", str(self.W[j]))


    if(E == 0):
      print("Model fit complete")
      print("Num Epochs: ", (k+1))
      print("Final Error: ", E) break
  print("Final Weights: ", str(self.W))
```

Output:

```
Num Epochs:  4
Final Error:  0.0
Final Weights:  [[ 1. -1. -2. -1.]
 [ 2. -1.  1. -2.]
 [-1.  1.  0. -1.]
 [-1. -1.  0.  0.]]
```

MCPTA Code:

```
class MCPTA:

    def_init____(self):
        # 4 x 4 weight matrix of random values self.W =
        np.zeros((4,4))
def predict(self, X_test):
        res = np.dot(X_test, self.W) return
        np.where(res >= 0, 1, 0)
def fit(self, X_train, d_train, learning_rate, epochs):

        print("Initial Weights: ", self.W)

        for k in range(epochs):
                print("############################ Epoch ", (k+1), "
############################")
for i in range(0,8): E = 0 print("######## X ", i, "
          #########") print("X_train: ", X_train[i])
          for j in range(0,4):

                print("Starting Weights: ", str(self.W[j])) z =

                np.dot(self.W[j].T, X_train[i])


                print("Z: ", z) y_net
                = sigmoid(z)
```

```
print("Y: ", str(y_net)) e

= d_train[i][j] - y_net E =

E + 0.5*(e)**2

print("Error: ", E)

self.W[j] = self.W[j] + (learning_rate * e * y_net * (1 - y_net) * X[i]) print("New

Weights: ", str(self.W[j]))


if(E <0.25):
    print("Num  Epochs:  ",  (k+1))
    print("Final Error: ", E) break
print("Final Weights: ", str(self.W))
```

Output:

```
Num Epochs:  8
Final Error:  0.24645643966221747
Final Weights:  [[ 0.98527061 -0.26500754 -1.55818758 -0.75591037]
 [ 1.03784349 -0.50210392  0.78387578 -1.43160128]
 [-1.43941956  1.26018238 -0.16700673 -0.75365022]
 [-1.32841085 -1.20028229  0.17104602  0.12224247]]
```

Conclusion:

Thus, we have solved the given problem using MCPTA and RCPTA. We have found the final weights which help in solving the problem.