Name- Yash Gandhi BE IT Batch A 2017140014

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:

$$\text{Class} = 1 \text{ for } \mathbf{x} = [1 \quad 0 \quad 0]^t, \quad [1 \quad 1 \quad 0]^t$$
$$\text{Class} = 2 \text{ for } \mathbf{x} = [1 \quad 0 \quad 1]^t, \quad [1 \quad 1 \quad 1]^t$$
$$\text{Class} = 3 \text{ for } \mathbf{x} = [0 \quad 1 \quad 0]^t, \quad [0 \quad 1 \quad 1]^t$$
$$\text{Class} = 4 \text{ for } \mathbf{x} = [0 \quad 0 \quad 0]^t, \quad [0 \quad 0 \quad 1]^t$$

**Assume initial weights as zero and bias input as 1.

Algorithms:

Algorithm (RDPTA)

- Given are P training pairs arranged in the training set.
  { $Y_1, d_1, Y_2, d_2, \ldots\ldots, Y_p, d_p$ }
  where, Y_i is $(J \cdot 1)$; d_i is $(K \cdot 1)$ and
  $i = 1, 2, \ldots, P$
- Note that the J^{th} component of each y_i has the value – 1 since input vectors have been augmented. Integer *q* denotes the training step and *p* denotes the counter within the training cycle.

Step 1 : | > 0 chosen

Step 2 : Weights **W** are initialized as small random values ; **W** is $(K \cdot J)$

q □ 1 ; p □ 1 ; E □ 0

Step 3 : Training step starts here.

Input is presented, output is computed.

Y □ Y_p ; d □ d_p

o_k □ sgn for k = 1, 2,…….. K

where W_k is the k^{th} row of **W**

Step 4 : Weights are updated

$W_k \square W_k + c (d_k - o_k) Y$ for $k = 1, 2, \ldots, K$

where, W_k is the k^{th} row of **W**.

Step 5 : Cumulative cycle error is computed by adding the present error.

$E \square E + (d_k - o_k)^2$ for $k = 1, 2, \ldots, K$

Step 6 : If $p < P$ then

$p \square p + 1$

$q \square q + 1$ go to step 3

Otherwise, go to step 7.

Step 7 : The training cycle is completed for E =0, terminate the training session and output weights **W**, q and E.

If $E > 0$, then

$E \square 0$

$p \square 1$ and initiate a new training cycle by going to step 3.

Algorithm (MCPTA)

- Given are P training pairs arranged in the training set.

  $\{ Y_1, d_1, Y_2, d_2, \ldots\ldots, Y_p, d_p \}$

  where, Y_i is $(J \cdot 1)$; d_i is $(K \cdot 1)$ and

  $i = 1, 2, \ldots, P$

- Note that the J^{th} component of each y_i has the value – 1 since input vectors have been augmented. Integer *q* denotes the training step and *p* denotes the counter within the training cycle.

Step 1 : $\eta > 0$; $E_{max} > 0$ chosen

Step 2 : Weights **W** are initialized as small random values ; **W** is $(K \cdot J)$

$q \square 1$; $p \square 1$; $E \square 0$

Step 3 : Training step starts here.

Input is presented, output is computed.

$Y \square Y_p$; $d \square d_p$

$o_k \square f$ for $k = 1, 2, \ldots\ldots K$

where W_k is the k^{th} row of **W**

and f(net) is $\lfloor - 1$

Step 4 : Weights are updated

$W_k \square W_k + (d_k - o_k)$ for $k = 1, 2, \ldots, K$

where, W_k is the k^{th} row of **W**.

Step 5 : Cumulative cycle error is computed by adding the present error.

$E \square E + (d_k - o_k)^2$ for $k = 1, 2, \ldots, K$

Step 6 : If $p < P$ then

$p \square p + 1$

$q \square q + 1$ go to step 3

Otherwise, go to step 7.

Step 7 : The training cycle is completed for E < Emax, terminate the training session and output weights **W**, q and E.

If E > E_{max}, then

E □ 0

p □ 1 and initiate a new training cycle by going to step 3.

Tool/Language:

Any Programming language: Java/Python/C/C++

Weight Update Equations:

For unipolar binary
o=1 if net>=0 else 0
RDPTA : Wnew= Wi +C(d-o)*xi*

For unipolar sigmoid
*F(net)=1/1+exp^(-l*net)*
o=f(net)
MCPTA : Wnew= Wi +C(d-o)*o*(1-o)*xi*

Code:

```
#include <stdio.h>
#include <stdlib.h>

int activation(int O)
{
   if(O>=0)
   O=1;
   else
   O=0;

   return O;
}

int main()
{
   int m,n,class,i,j,k;

   m=8,n=4,class=4;
   int input[8][4] = {
                {1,0,0,-1},
                {1,1,0,-1},
                {1,0,1,-1},
                {1,1,1,-1},
                {0,1,0,-1},
                {0,1,1,-1},
                {0,0,0,-1},
                {0,0,1,-1}};
   int d[8][4]= {
                    {1,0,0,0},
```

```
                {1,0,0,0},
                {0,1,0,0},
                {0,1,0,0},
                {0,0,1,0},
                {0,0,1,0},
                {0,0,0,1},
                {0,0,0,1}};
float weight[4][4]= {
                {0,0,0,0},
                {0,0,0,0},
                {0,0,0,0},
                {0,0,0,0}};


float e=1;
float sum;
float o;
float c=1;
int count=0;
while(e!=0)
{
   e=0;
   count++;
   for(i=0;i<m;i++)
   {
      for(j=0;j<class;j++)
      {
        sum=0;
        for(k=0;k<n;k++)
        {
         sum=sum + input[i][k]*weight[j][k];
        }

        o=activation(sum);
        for(k=0;k<n;k++)
        {
           weight[j][k]= weight[j][k]+ c*(d[i][j]-o)*input[i][k];
        }

        e=e+0.5*(d[i][j]-o)*(d[i][j]-o);
      }

      printf("\n Error after input %d is %f ", i,e);
   }
printf("\n%d epochs\n",count);

}

   printf("\n Final Weights after %d epochs\n",count);

for (i = 0; i < class; i++) {
   for (j = 0; j < n; j++) {
      printf("%f\t", weight[i][j]);
   }
   printf("\n");
}
```

```
      for(j=0;j<class;j++)
         {
           sum=0;
           for(k=0;k<n;k++)
           {
            sum=sum + input[7][k]*weight[j][k];
           }

           o=activation(sum);
           printf("\nOutput %f",o);
         }

   return 0;
}
```

MCPTA

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
float activation(float O)
{
   float res=1/(1+exp(-O));
   return res;
}

int main()
{
   int m,n,class,i,j,k;

   m=8,n=4,class=4;
   int input[8][4] = {
                {1,0,0,-1},
                {1,1,0,-1},
                {1,0,1,-1},
                {1,1,1,-1},
                {0,1,0,-1},
                {0,1,1,-1},
                {0,0,0,-1},
                {0,0,1,-1}};
   int d[8][4]= {
                    {1,0,0,0},
                    {1,0,0,0},
                    {0,1,0,0},
                    {0,1,0,0},
                    {0,0,1,0},
                    {0,0,1,0},
                    {0,0,0,1},
                    {0,0,0,1}};
   float weight[4][4]= {
                    {0,0,0,0},
                    {0,0,0,0},
                    {0,0,0,0},
                    {0,0,0,0}};
```

```
float e=1;
float sum;
float o;
float c=1;
int count=0;

while(e>0.1)
{
   e=0;
   count++;
   for(i=0;i<m;i++)
   {
      for(j=0;j<class;j++)
      {
        sum=0;
        for(k=0;k<n;k++)
        {
         sum=sum + input[i][k]*weight[j][k];
        }

        o=activation(sum);
        for(k=0;k<n;k++)
        {
           weight[j][k]= weight[j][k]+ c*(d[i][j]-o)*input[i][k]*o*(1-o);
        }

        e=e+0.5*(d[i][j]-o)*(d[i][j]-o);
      }

      printf("\n Error after input %d is %f ", i,e);
   }
printf("\n%d epochs\n",count);

}

   printf("\n Final Weights after %d epochs\n",count);

for (i = 0; i < class; i++) {
   for (j = 0; j < n; j++) {
      printf("%f\t", weight[i][j]);
   }
   printf("\n");
}
   for(j=0;j<class;j++)
      {
        sum=0;
        for(k=0;k<n;k++)
        {
         sum=sum + input[7][k]*weight[j][k];
        }

        o=activation(sum);
        printf("\nOutput %f",o);
      }
```

```
    return 0;
}
```

Results:

RDPTA- Initial Weights are all 0 and bias is taken as -1

Total no. of Epochs = 9

```
1 epochs
 Error after input 0 is 1.500000
 Error after input 1 is 1.500000
 Error after input 2 is 2.500000
 Error after input 3 is 2.500000
 Error after input 4 is 3.500000
 Error after input 5 is 3.500000
 Error after input 6 is 4.500000
 Error after input 7 is 5.000000
 Weights after 1 epochs
-1.000000       0.000000        -1.000000       1.000000
0.000000        -1.000000       0.000000        2.000000
-1.000000       1.000000        0.000000        1.000000
-1.000000       0.000000        0.000000        0.000000


2 epochs
 Error after input 0 is 0.500000
 Error after input 1 is 0.500000
 Error after input 2 is 1.000000
 Error after input 3 is 1.000000
 Error after input 4 is 2.000000
 Error after input 5 is 2.000000
 Error after input 6 is 2.500000
 Error after input 7 is 3.000000
 Weights after 2 epochs
0.000000        -1.000000       -1.000000       1.000000
1.000000        -1.000000       0.000000        2.000000
-1.000000       1.000000        0.000000        1.000000
-1.000000       -1.000000       0.000000        0.000000
```

```
9 epochs
 Error after input 0 is 0.000000
 Error after input 1 is 0.000000
 Error after input 2 is 0.000000
 Error after input 3 is 0.000000
 Error after input 4 is 0.000000
 Error after input 5 is 0.000000
 Error after input 6 is 0.000000
 Error after input 7 is 0.000000
 Weights after 9 epochs
1.000000        0.000000        -3.000000       1.000000
3.000000        -1.000000       2.000000        4.000000
-1.000000       1.000000        0.000000        1.000000
-1.000000       -1.000000       0.000000        0.000000

 Final Weights after 9 epochs
1.000000        0.000000        -3.000000       1.000000
3.000000        -1.000000       2.000000        4.000000
-1.000000       1.000000        0.000000        1.000000
-1.000000       -1.000000       0.000000        0.000000

Output 0.000000
Output 0.000000
Output 0.000000
Output 1.000000
```

MCPTA- All initial weights are 0 bias is taken as -1
Number of epochs=150, Emax=0.1

```
149 epochs

 Error after input 0 is 0.013065
 Error after input 1 is 0.025075
 Error after input 2 is 0.036787
 Error after input 3 is 0.050297
 Error after input 4 is 0.059193
 Error after input 5 is 0.071356
 Error after input 6 is 0.085576
 Error after input 7 is 0.100392
150 epochs

 Error after input 0 is 0.012965
 Error after input 1 is 0.024887
 Error after input 2 is 0.036511
 Error after input 3 is 0.049920
 Error after input 4 is 0.058750
 Error after input 5 is 0.070822
 Error after input 6 is 0.084933
 Error after input 7 is 0.099639
 Final Weights after 150 epochs
4.693266        -0.152911       -4.972585       2.353114
4.124913        -0.239467       4.074080        6.123759
-4.938935       4.726759        -0.134519       2.378151
-4.787407       -4.767481       0.148387        -2.061567

Outout for 7th pattern

Output 0.000658
Output 0.114085
Output 0.074975
Output 0.901140

...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion:

RDPTA (R-Category Discrete Perceptron Training Algorithm) and MCPTA (Multi-Category Perceptron Training Algorithm) were implemented for the given problem and the weights were updated in 9 and 150 epochs respectively. As MCPTA uses continuous activation function the error at the last epoch is 0.1 whereas it is 0 in RCPTA.