Name – Yash Gandhi BE IT Batch A 2017140014

Aim:

To apply Winner-Take-all learning to a given problem

Problem Statement:

Use winner take all learning to solve the following clustering problem

---

Consider the given input patterns P_1, P_2, P_3, P_4, P_5 and P_6 as well as initial weight W_1, W_2, W_3.

$$P_1 = \begin{bmatrix} -0.1961 \\ 0.9806 \end{bmatrix} \quad P_2 = \begin{bmatrix} 0.1961 \\ 0.9806 \end{bmatrix} \quad P_3 = \begin{bmatrix} 0.9806 \\ 0.1961 \end{bmatrix}$$

$$P_4 = \begin{bmatrix} 0.9806 \\ -0.1961 \end{bmatrix} \quad P_5 = \begin{bmatrix} -0.5812 \\ -0.8137 \end{bmatrix} \quad P_6 = \begin{bmatrix} -0.8137 \\ -0.5812 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix} \quad W_2 = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix} \quad W_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

Consider, $\alpha = 0.5$, perform clustering using winner take all algorithm (or competitive learning). The input patterns are presented in the following sequence P_4, P_3, P_2 ,P_1, P_6, P_5. (Run for at least 10 epochs)

Tool/Language:

Programming language: C, Visualization: Online- Metachart.com

Algorithm:

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

  $\{ Y_1, Y_2, \ldots\ldots, Y_P \}$ and $W_1, \ldots W_n$ Weight vectors for N neurons

  i = 1, 2, ….., P

- Integer *q* denotes the training step for each Input and *p* denotes the counter for epoch.

Step1: q=0 ; p =1 ,pmax=10 ;
Step 2: Training step starts here.
Input is presented, output is computed.
$Y = Y_p$;
$O=Y_p * W_k^T$ for k = 1, 2,…….. K
where W_k is the k^{th} row of **W**
Step 3: Find Omax for for given input let the index of the row in Weight vector for which we get Omax be u
Step 4 : Weights are updated
$W_u = W_u + C*(Y_p - W_u)$ C is learning constant
where, W_u is the u^{th} row of **W**.
Step 5: If p < pmax then
p = p + 1
q = q + 1 go to step 2
Otherwise, check the clusters formed.

Code:

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

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

   m=6,n=2,class=3;
   float input[6][2] = {
                {-0.1961,0.9806},
                { 0.1961,0.9806},
                { 0.9806,0.1961},
                { 0.9806,-0.1961},
                {-0.5812,-0.8137 },
                {-0.8137,-0.5812}};

   float weight[3][2]= {
                    {0.7071,-0.7071},
                    {0.7071,0.7071},
                    {-1,0}};


   float sum;
   float c=0.5;
   int count=1;
   int index;
   float temp;
   while(count<=10)
   {
      printf("\n ****Epoch  %d ****\n",count);
      for(i=0;i<m;i++)
      {
         temp=0;
         index=0;
         for(j=0;j<class;j++)
         {
            sum=0;
            for(k=0;k<n;k++)
            {
             sum=sum + input[i][k]*weight[j][k];
            }

            if(temp<sum)
            {
               temp=sum;
               index=j;
            }
         }

         printf("\nwinner after step %d is neuron %d",i+1,index+1);
         for(int x=0;x<n;x++)
         {
          weight[index][x]=weight[index][x]+c*(input[i][x]-weight[index][x]);
```

```
            }

            printf("\n Weights after step  %d \n",i+1);

            for (int p = 0; p < class; p++)
            {
              for (int q = 0; q < n; q++)
              {
                printf("%f\t", weight[p][q]);
              }
             printf("\n");
            }
        }
    count++;
    }
        printf("\n Final Weights after %d epochs\n",count-1);
    for (i = 0; i < class; i++) {
        for (j = 0; j < n; j++) {
            printf("%f\t", weight[i][j]);
        }
        printf("\n");
    }


    return 0;
}
```

Output:

1. **Epoch No. ,Winner neuron Index, Winner's updates weights and final weights.**

EPOCH 1

```
****Epoch  1 ****

winner after step 1 is neuron 2
 Weights after step  1
0.707100        -0.707100
0.255500        0.843850
-1.000000       0.000000

winner after step 2 is neuron 2
 Weights after step  2
0.707100        -0.707100
0.225800        0.912225
-1.000000       0.000000

winner after step 3 is neuron 1
 Weights after step  3
0.843850        -0.255500
0.225800        0.912225
-1.000000       0.000000

winner after step 4 is neuron 1
 Weights after step  4
0.912225        -0.225800
0.225800        0.912225
-1.000000       0.000000

winner after step 5 is neuron 3
 Weights after step  5
0.912225        -0.225800
0.225800        0.912225
-0.790600       -0.406850

winner after step 6 is neuron 3
 Weights after step  6
0.912225        -0.225800
0.225800        0.912225
-0.802150       -0.494025
```

EPOCH 10

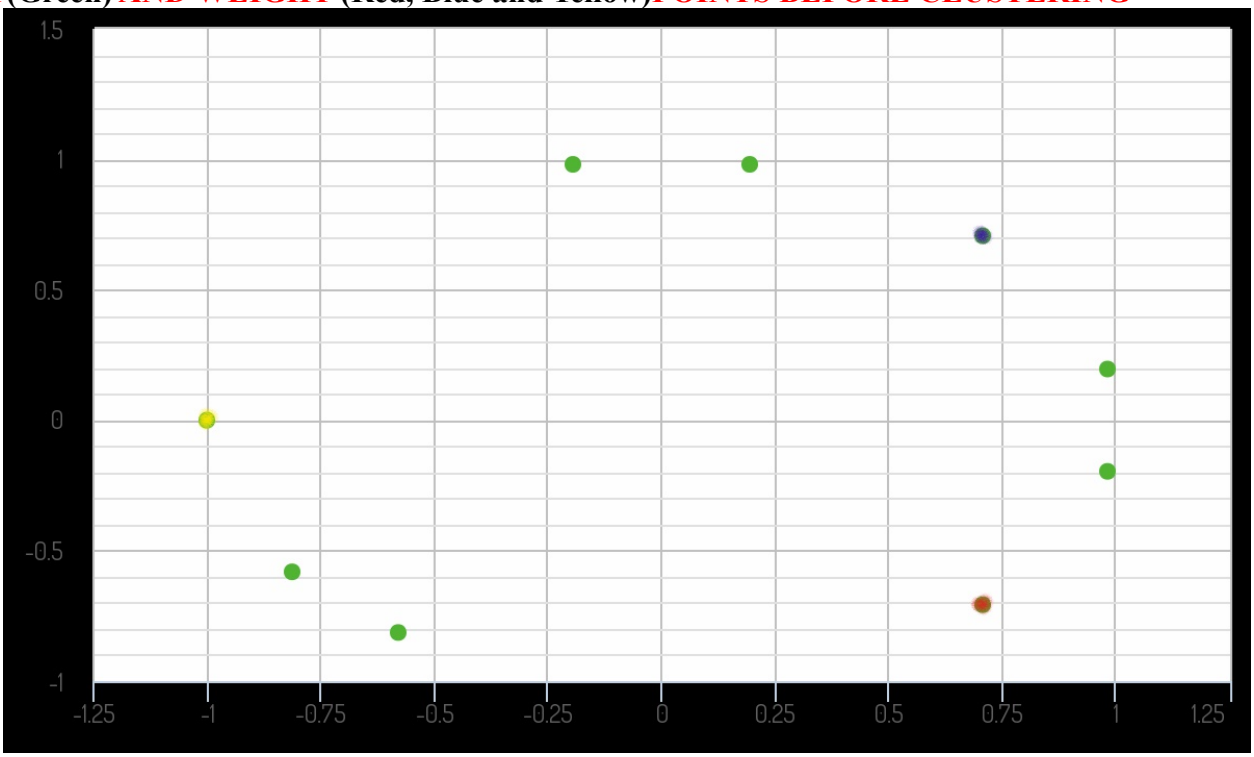
```
****Epoch  10 ****

winner after step 1 is neuron 2
 Weights after step  1
0.980599        -0.065369
-0.065365       0.980599
-0.736201       -0.658697

winner after step 2 is neuron 2
 Weights after step  2
0.980599        -0.065369
0.065367        0.980600
-0.736201       -0.658697

winner after step 3 is neuron 1
 Weights after step  3
0.980599        0.065365
0.065367        0.980600
-0.736201       -0.658697

winner after step 4 is neuron 1
 Weights after step  4
0.980600        -0.065367
0.065367        0.980600
-0.736201       -0.658697

winner after step 5 is neuron 3
 Weights after step  5
0.980600        -0.065367
0.065367        0.980600
-0.658701       -0.736199

winner after step 6 is neuron 3
 Weights after step  6
0.980600        -0.065367
0.065367        0.980600
-0.736200       -0.658699
```

FINAL WEIGHT VECTORS

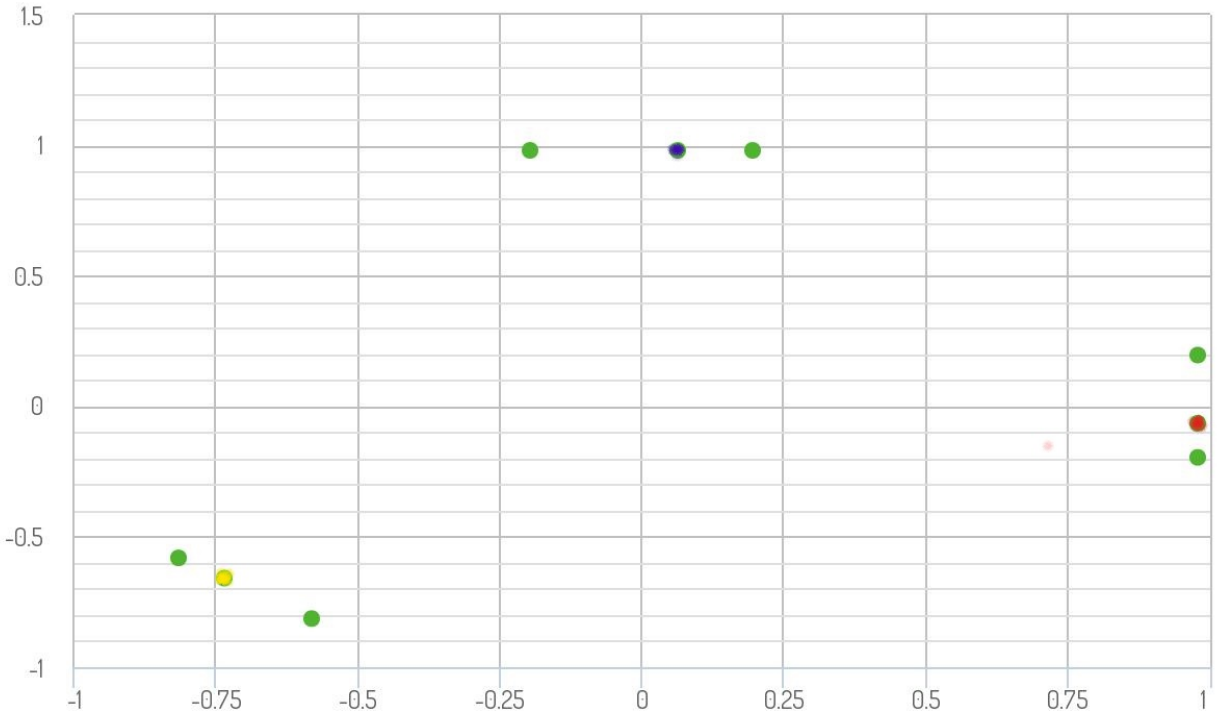
```
 Final Weights after 10 epochs
0.980600        -0.065367
0.065367        0.980600
-0.736200       -0.658699
```

2. **Visualization of the input patterns before after each epoch**

INPUT(Green) AND WEIGHT (Red, Blue and Yellow)POINTS BEFORE CLUSTERING



INPUT(Green) AND WEIGHT (Red, Blue and Yellow)POINTS AFTER CLUSTERING



Conclusion:

After performing Winner Takes All clustering Algorithm we can see using the visualization that the weight vectors are updated and made to align with the nearby clusters after 10 epochs. All the 3 weight vectors are included in 3 clusters each. Hence we can use this unsupervised algorithm in order to find the most efficient weights w.r.t to the inputs given.