# EE 550 Project 4

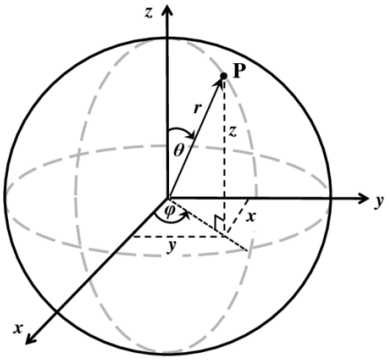
Sezen Perçin – 2018401000

## Winner Takes All Network

For this homework I chose the part A and implemented the “Winner Takes All Network”.

1)Generating the data points:

I have generated the data points using the spherical coordinates. I have chosen 3 region and created the corresponding random theta and phi values to generate clusters that contains 30 data points each on the unit sphere.




$x = r \sin\theta \cos\varphi$

$y = r \sin\theta \sin\varphi$

$z = r \cos\theta$

$r = \sqrt{x^2 + y^2 + z^2}$

$\theta = \cos^{-1}(z/r)$

$\varphi = \tan^{-1}(y/x)$

Region 1:

phi ∈ (pi/6, 2pi/6)  phi=pi/6+rand(30,1)*(pi/6);

theta ∈ (pi/6, 2pi/6)  theta=pi/6+rand(30,1)*(pi/6);

r=1;

x= r.*sin(theta).*cos(phi)

y= r.*sin(theta).*sin(phi)

z= r.*cos(theta)];  Shown in red.

Fig.1.1

In a similar way:

Region 2:  phi ∈ (10pi/6, 11pi/6)  theta ∈ (4pi/6, 5pi/6)  Shown in green.

Region 3:  phi ∈ (4pi/6, 5pi/6)  theta ∈ (4pi/6, 5pi/6)  Shown in blue.

2)

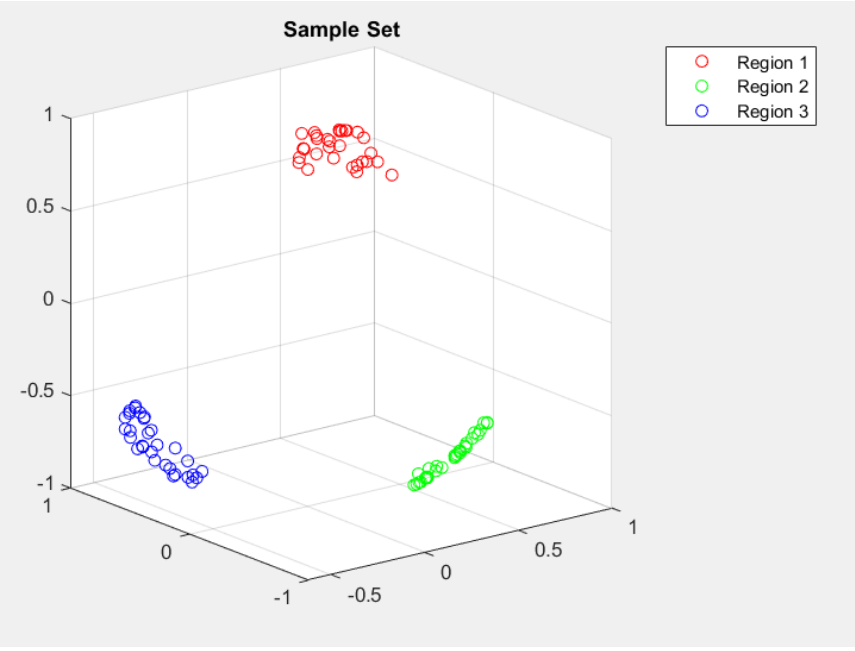



Fig 2.1

3) I implemented a  part where the weights are randomly initialized similar to the generation of the data set in the script which can be seen below:

```
%Randomly intitializing the weight vectors
phi=rand(1,1)*(2*pi);
theta=rand(1,1)*(pi);
w1=[(r.*sin(theta).*cos(phi)); (r.*sin(theta).*sin(phi)); r.*cos(theta)];
phi=rand(1,1)*(2*pi);
theta=rand(1,1)*(pi);
w2=[(r.*sin(theta).*cos(phi)); (r.*sin(theta).*sin(phi)); r.*cos(theta)];
phi=rand(1,1)*(2*pi);
theta=rand(1,1)*(pi);
w3=[(r.*sin(theta).*cos(phi)); (r.*sin(theta).*sin(phi)); r.*cos(theta)];
% constructing the weight matrix
w=[w1 w2 w3];
```

Their place on the unit sphere with other data points can be seen in the Figure 4.3 and 4.4 in the part 4 below.

4) I started by constructing the training set and the test set out of the data set.  I took 27 and 3 samples from each cluster for training and testing purposes respectively. To increase efficiency, I shuffled the training set before the start of the algorithm.

Then for each sample:

```
for mu=1:27
    y=w'*trainset(mu,:)';
    maxofy=max(y);
    winner=find(y==maxofy);
    outputtrain=(y==maxofy);
    delta=lrate*[trainset(mu,:)'-w(:,winner)];
    toplot=[w(:,winner)'; (w(:,winner)+delta)'];
    %plotting the trajectories
    if winner==1
        plot3(toplot(:,1),toplot(:,2),toplot(:,3),'k-');
    elseif winner==2
        plot3(toplot(:,1),toplot(:,2),toplot(:,3),'m-');
    elseif winner==3
        plot3(toplot(:,1),toplot(:,2),toplot(:,3),'c-');
    end
    w(:,winner)=w(:,winner)+delta;

end
```

Fig 4.1

Output vector is found by multiplying the weight vectors w1,w2,w3 with the input components x1,x2,x3. I performed this operation by multiplying the transpose of the weight matrix with the sample vector x.

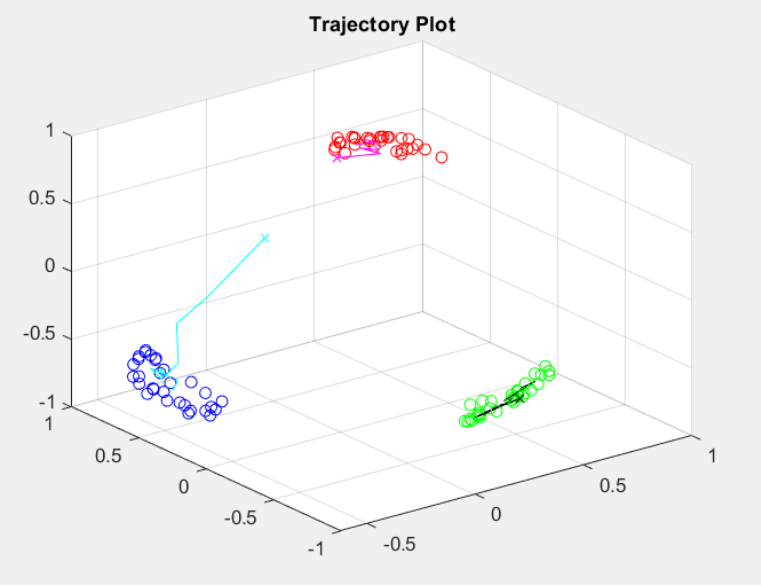
Maximum element of y which is the value at the winning unit is found and its index is labeled as the “winner”.The output vector which gives 1 for the winning unit and the zero for the others is  stored in the variable “outputtrain”.

Since the winner takes all algorithm labels the winner unit as 1 and the other  units as 0,  when the update rule below is used, the delta values for non winning units take the value of zero (since oi=1 for the winner unit and oi=0 for others). Therefore,  the learning only takesplace  at the weight vector that belongs to the winner unit. Therefore in the algorithm I only updated the winning units. The update rule below also enable us the perform the normalization operation on the weights:

$$\Delta w_{ij} = \eta \cdot o_i \cdot (\xi_j^{\mu} - w_{ij}), \quad \forall i$$

Fig. 4.2

In each step, the movement or the change in the trajectory of the weight vector is shown and can be seen below where the movement of w1 is shown in key black, w2 is shown in magenta and w3 is shown in cyan colors. Also the respectively colored “x” signs indicates their initialized starting points.



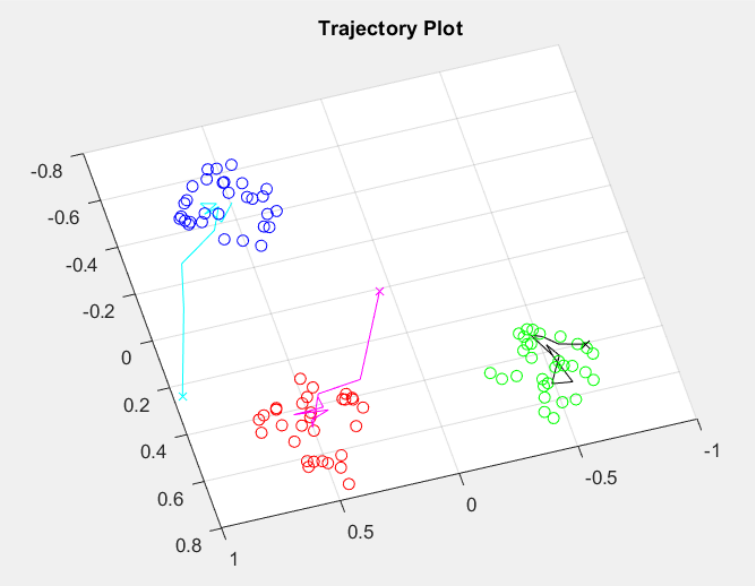




Fig. 4.3

Fig. 4.4

If we look at the trajectory plot from the below, the movements of the weight vectors can be seen. As the system is trained, it can be observed that the weight vectors move towards the center of the clusters. This is a result of the update that is performed on the weight vector. Each sample makes the closest weight vector which is used in the calculations of the winning unit, closer to itself. That causes the movement to be into the cluster centers. With this way the samples which are close to one cluster produce the maximum results with the corresponding weight vector since their dot product get bigger as they get closer which also yields to winning unit or de region the sample belongs.

However, if the initial vectors are not placed in a suitable sense (if some of the systems are far away from any input vector) they never get to learn and never win. Therefore, they become the “dead units”. The examples can be seen below:

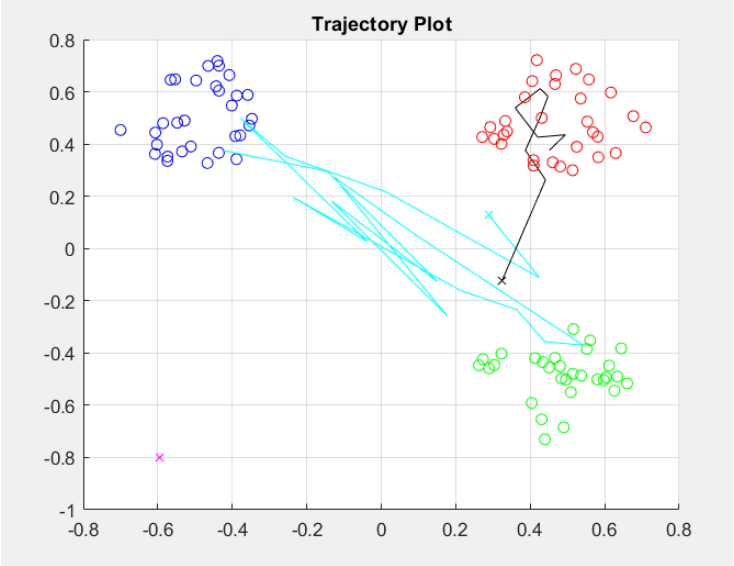



Fig 4.5

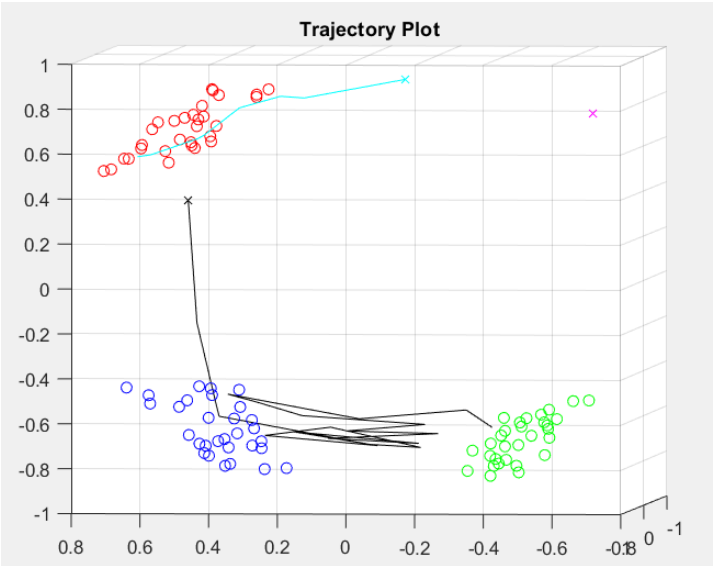



Fig. 4.6

Therefore, the selection of the initial weight vectors are really important. I also added a secure case in the script which is commented out where the winning operation takes place without any trouble.

5) For the testing I have used the test set that I have constructed from the data set before. In this data set the samples are put in order so that the first 3 are from region 1, the next three samples are from region 2 and the last three samples are from the region 3. Its construction can be seen below on Fig 5.1:

```
%forming training set
trainset=[p1(1:27,:); p2(1:27,:); p3(1:27,:)];
%forming test set
testset=[p1(28:30,:); p2(28:30,:); p3(28:30,:)];
```

Fig 5.1

```
%Trying the test set
class=zeros(9:1);
outputtest=[];
for i=1:9
    ytest=w'*testset(i,:)';
    class(i)= find(ytest==max(ytest));
    outputtest=[outputtest [(ytest==max(ytest)); class(i)]];
end
```

Fig. 5.2

As it can be seen in Fig. 5.2, each sample in test set is multiplied with the weight matrix and the output along with the class (which is the indice of the winning unit) is added to the array “outputtest”. In array “output test” each column contains the system output (the first 3 entry) and the class of region that is found (thelast entry). The results can be seen below in Fig 5.3:

```
outputtest =

     0     0     0     1     1     1     0     0     0
     1     1     1     0     0     0     0     0     0
     0     0     0     0     0     0     1     1     1
     2     2     2     1     1     1     3     3     3
```

Fig 5.3

As it is indicated in part 4, weight vector 1 (w1) is shown in black (associated with 1th unit on the output vector), w2 is shown in magenta (associated with 2nd unit on the output vector) and w3 is shown in cyan (associated with 3rd unit on the output vector)on the trajectory plot.

It can also be seen in Fig.4.3 and 4.4, at the end of the training the w1 moved to the center of "region 2" , w2 moved to the center of "region 1" and the w3 moved to the center of "region 3".

Therefore, the results in 5.3 validates the accuracy of the system since:

-The first three samples belong to region 1 and their output is 2 which is associated with w2 that moved to the center of region 1.

-The next set of three samples belong to region 2 and their output is 1 which is associated with w1 that moved to the center of region 2.

-The last three samples belong to region 3 and their output is 3 which is associated with w3 that moved to the center of region 3.

References:

Fig. 1.1: 'https://www.researchgate.net/figure/Figure-A1-Spherical-coordinates_fig8_284609648'

Fig 4.2: Taken from the course note "Lecture Notes 23 (24th May)" of EE550 on the Moodle platform.