PROJECT 5

Elif Konyar

5/16/2020

Project 5

In this project, we are asked to implement winner take-all network in order to cluster the data points that we generated.

Question 1

In this question, input data from 3 clusters is generated. There will be 3 clusters. In each cluster, there are 30 sample patterns. First, the z coordinate is generated such that it is between -1 and 1. Then, for the x and y coordinates, a theta angle is generated randomly, such that

$x^2+y^2+z^2=1$.

After determining z randomly between -1 and 1, x and y are found from the triangle in the xy plane:

$x=(1-z^{2)*}\cos(\text{theta})$

$y=(1-z^2)*\sin(\text{theta})$

The data points are generated such that they form 3 clusters, and so the generation is controlled by the theta and the z value by giving the ranges accordingly.

```
library(data.table)

## Warning: package 'data.table' was built under R version 3.4.2

library(rgl)

## Warning: package 'rgl' was built under R version 3.4.4

number_of_test_instances_in_cluster=3
set.seed(990)
number_of_instances_in_cluster=30
cluster1_theta <- runif(number_of_instances_in_cluster,0,(1/6)*pi)
z <- runif(number_of_instances_in_cluster,0.4,1)
x <- sqrt(1-z^2)*cos(cluster1_theta)
y <- sqrt(1-z^2)*sin(cluster1_theta)
cluster1=as.data.table(cbind(x,y))
cluster1=as.data.table(cbind(cluster1,z))
cluster1[,cluster:=1]
```

```
cluster2_theta <- runif(number_of_instances_in_cluster,pi,(7/6)*pi)
z <- runif(number_of_instances_in_cluster,0,0.6)
x <- sqrt(1-z^2)*cos(cluster2_theta)
y <- sqrt(1-z^2)*sin(cluster2_theta)
cluster2=as.data.table(cbind(x,y))
cluster2=as.data.table(cbind(cluster2,z))
cluster2[,cluster:=2]


cluster3_theta <- runif(number_of_instances_in_cluster,(10/6)*pi,(11/6)*pi)
z <- runif(number_of_instances_in_cluster,-1,-0.6)
x <- sqrt(1-z^2)*cos(cluster3_theta)
y <- sqrt(1-z^2)*sin(cluster3_theta)
cluster3=as.data.table(cbind(x,y))
cluster3=as.data.table(cbind(cluster3,z))
cluster3[,cluster:=3]
```

Ranges:

First cluster:

z value: (0.4,1)

theta: (0,30) (determines the x and y values, so it is the angle in the xy plane)

Second cluster:

z value: (0, 0.6)

theta: (180, 210)

Third cluster:

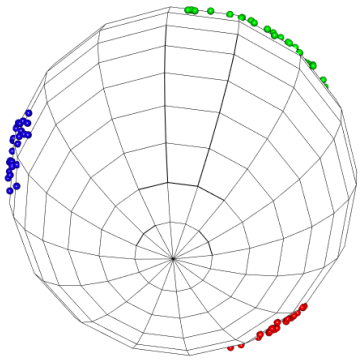
z value: (-1, -0.6)

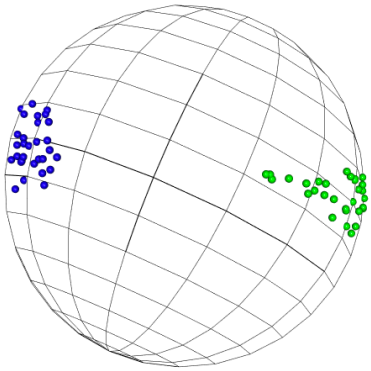
theta: (300, 330)

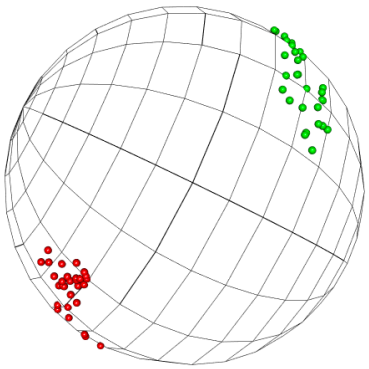
Question 2

In this question, we are asked to plot the sample points. Cluster 1 is in green, cluster 2 is in blue and cluster 3 is in red.

```
#----------------------Question 2----------------------------
spheres3d(0,0,0,lit=FALSE,color="white")
spheres3d(0,0,0,radius=1,lit=FALSE,color="black",front="lines")
spheres3d(cluster1$x,cluster1$y,cluster1$z,col="green",radius=0.02)
spheres3d(cluster2$x,cluster2$y,cluster2$z,col="blue",radius=0.02)
spheres3d(cluster3$x,cluster3$y,cluster3$z,col="red",radius=0.02)
par3d(windowRect = c(20, 30, 800, 800))
```







After plotting all the point, the test set is seperated. The first 3 sample patterns from each cluster is seperated as the test set.

```
test=cluster1[1:number_of_test_instances_in_cluster,]
cluster1=cluster1[(number_of_test_instances_in_cluster+1):.N]
test=rbind(test,cluster2[1:number_of_test_instances_in_cluster,])
cluster2=cluster2[(number_of_test_instances_in_cluster+1):.N]
test=rbind(test,cluster3[1:number_of_test_instances_in_cluster,])
cluster3=cluster3[(number_of_test_instances_in_cluster+1):.N]
```

Question 3

Weights are initialized in this question. They are created similarly to the first question, but no range is specified so that they are generated pure randomly. There will be 9 weights.

w_ij=weight of output neuron i from input neuron j. So, w_i*=weight vector of output neuron i

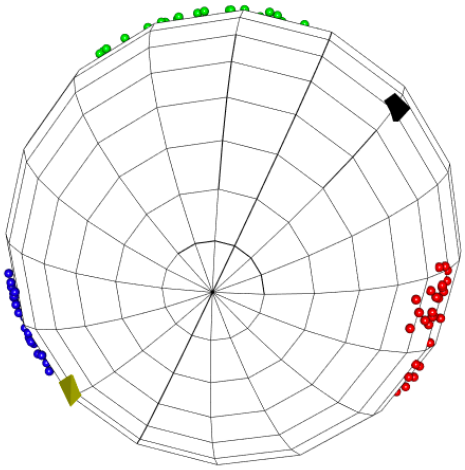
```
number_of_clusters=3
number_of_inputs=3
number_of_instances=90

number_of_weights=number_of_clusters*number_of_inputs

set.seed(750)
weights_theta <- runif(number_of_clusters,0,2*pi)
z <- runif(number_of_clusters,-1,1)
x <- sqrt(1-z^2)*cos(weights_theta)
y <- sqrt(1-z^2)*sin(weights_theta)
weights=as.data.table(cbind(x,y))
weights=as.data.table(cbind(weights,z))

weights
```

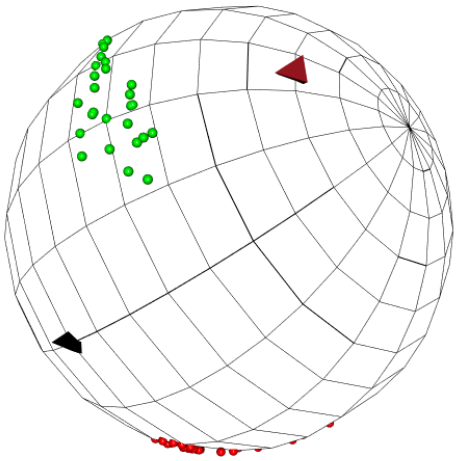
```
           x          y             z
1:  0.9275090 -0.3738004 -0.0005894122
2:  0.3180942  0.8362826  0.4465953303
3: -0.8192860 -0.5722761  0.0356433927
```

The weights can be seen below:

```
spheres3d(0,0,0,lit=FALSE,color="white")
spheres3d(0,0,0,radius=1,lit=FALSE,color="black",front="lines")
spheres3d(cluster1$x,cluster1$y,cluster1$z,col="green",radius=0.02)
spheres3d(cluster2$x,cluster2$y,cluster2$z,col="blue",radius=0.02)
spheres3d(cluster3$x,cluster3$y,cluster3$z,col="red",radius=0.02)
par3d(windowRect = c(20, 30, 800, 800))
shapelist3d(tetrahedron3d(), weights$x, weights$y, weights$z, size =  0.05,
            color = c("black","brown","yellow"))
```

The rgl package does not support putting crosses as far as I know, so I put tetrahedrons with different colors to demonstrate weights.





Question 4

In this question, the competitive learning algorithm is implemented. The data is shuffled so that the network is fed with the sample patterns from each cluster randomly. The learning rate is taken as 0.2.

```
data=rbind(cluster1,cluster2)
data=rbind(data,cluster3)

learning_rate=0.2

#shuffling the data
set.seed(23)
rows <- sample(nrow(data))
data_shuffled <- data[rows, ]

weight_matrix=copy(as.matrix(weights))
```

The algorithm is given below. In the output, for each output neuron input_vector*t(weight_vector) is calculated. Input vector and weight vectors are defined as row vectors here. Then, the winner unit is decided such that the output for that neuron is the maximum.

Then, the update rule for the winning unit is (normalization is applied to prevent the weights grow out of bound):

weight_change=learning_rate*(instance-weights_of_winner_unit)
(delta_w=learning_rate*(zeta-w_i*))

However, when the weights are checked, I saw that they are not of unit 1 as it is expected. Thus, I scaled the weight such that its direction is not changed, but the norm became 1. Then the history of weights are stored in weight_trajectory table.

```
weight_trajectory=data.table()
weight_trajectory=copy(weights[,output_index:=1:.N])
weight_trajectory[,iteration:=0]
for (i in 1:(number_of_instances-(number_of_clusters*number_of_test_instances
_in_cluster))){
  instance=as.matrix(data_shuffled[i,1:number_of_inputs])
  output=instance%*%t(weight_matrix[,1:number_of_inputs])
  winner_unit=which(output==max(output))

  #weight update
  weight_change=learning_rate*(instance-weight_matrix[winner_unit,1:number_of
_inputs])
  beforescaled_weight=weight_matrix[winner_unit,1:number_of_inputs]+weight_ch
ange
  weight_norm=(beforescaled_weight[1,1]*beforescaled_weight[1,1]+beforescaled
_weight[1,2]*beforescaled_weight[1,2]+beforescaled_weight[1,3]*beforescaled_w
eight[1,3])^(1/2)
  weight_matrix[winner_unit,1:number_of_inputs]=(1/weight_norm)*beforescaled_
weight
  weight_trajectory=rbind(weight_trajectory,data.table(x=weight_matrix[winner
_unit,1],y=weight_matrix[winner_unit,2],z=weight_matrix[winner_unit,3],output
_index=winner_unit,iteration=i))
}
```

The initial weights are:

```
weights
```

```
##             x          y             z output_index
## 1:  0.9275090 -0.3738004 -0.0005894122            1
## 2:  0.3180942  0.8362826  0.4465953303            2
## 3: -0.8192860 -0.5722761  0.0356433927            3
```

The final weights in the order of output_index are:

```
weight_matrix
```

```
##               x          y          z
## [1,]  0.5277244 -0.4265337 -0.7345583
## [2,]  0.6225859  0.1727734  0.7632405
## [3,] -0.8917973 -0.2343928  0.3869853
```

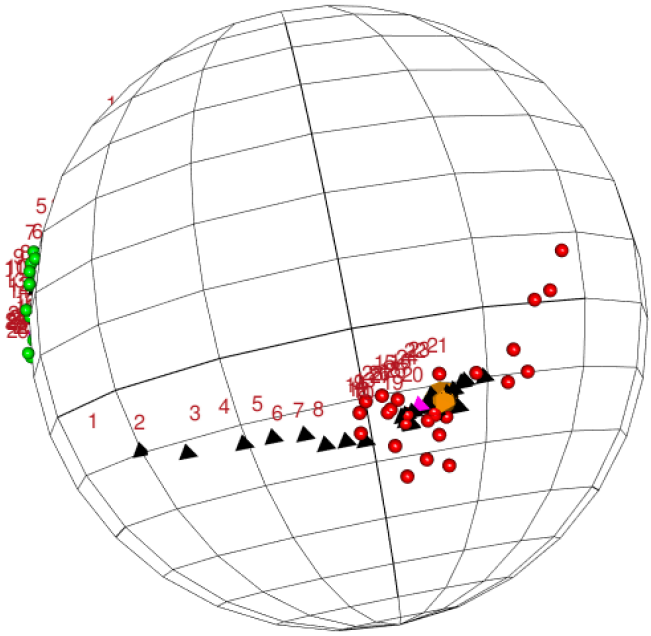
The weight trajectories are seperated for the upcoming sections because in order to show the trajectories, I could not combine the points with a line due to this package so I put the iteration numbers in the plot. When I was locating the iteration numbers, I had to locate numbers accordingly.

```
#weight_matrix=as.data.table(weight_matrix)
weight_trajectory_c1=weight_trajectory[output_index==1]
weight_trajectory_c1[,iteration:=1:.N]
weight_trajectory_c1[,color:=ifelse(iteration==max(iteration),"#FF00FF","#000
```

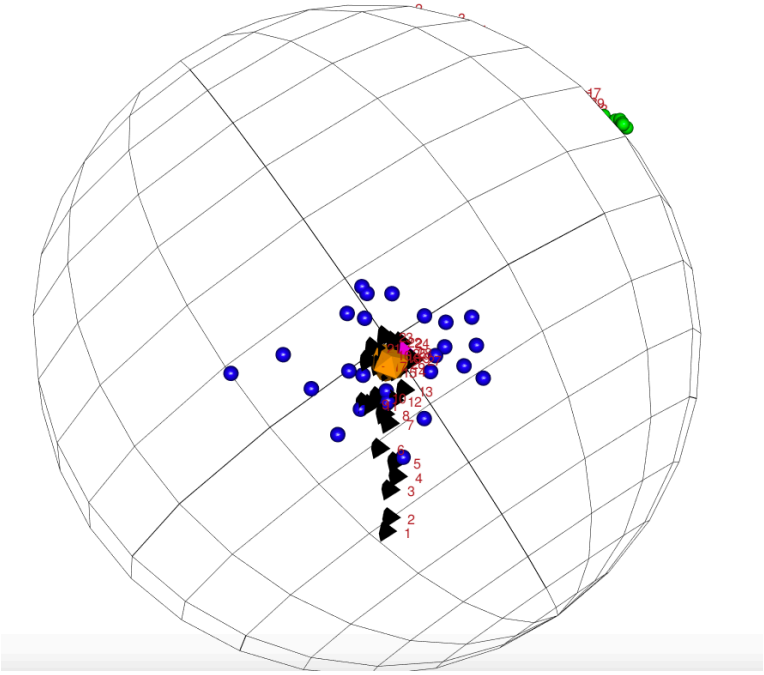
```
000")]
weight_trajectory_c2=weight_trajectory[output_index==2]
weight_trajectory_c2[,iteration:=1:.N]
weight_trajectory_c2[,color:=ifelse(iteration==max(iteration),"#FF00FF","#000
000")]
weight_trajectory_c3=weight_trajectory[output_index==3]
weight_trajectory_c3[,iteration:=1:.N]
weight_trajectory_c3[,color:=ifelse(iteration==max(iteration),"#FF00FF","#000
000")]
```

Below, first the centers are calculated for each cluster. Then, the training points with the weight trajectories are shown. The numbers indicate the iterations. The tetrahedrons are the weights in each iteration. They are in black, but the final weight is shown in purple.

```
centers=data_shuffled[,.(center_x=mean(x),center_y=mean(y),center_z=mean(z)),
by=.(cluster)]

spheres3d(0,0,0,lit=FALSE,color="white")
spheres3d(0,0,0,radius=1,lit=FALSE,color="black",front="lines")
spheres3d(cluster1$x,cluster1$y,cluster1$z,col="green",radius=0.02)
spheres3d(cluster2$x,cluster2$y,cluster2$z,col="blue",radius=0.02)
spheres3d(cluster3$x,cluster3$y,cluster3$z,col="red",radius=0.02)
par3d(windowRect = c(20, 30, 1000, 1000))
#shapelist3d(tetrahedron3d(), weight_trajectory$x, weight_trajectory$y, weigh
t_trajectory$z, size =  0.02,
#            color = "black")#weight_trajectory$output_index)
shapelist3d(tetrahedron3d(), weight_trajectory_c1$x, weight_trajectory_c1$y,
weight_trajectory_c1$z, size =  0.02,
            color = weight_trajectory_c1$color)
texts3d(weight_trajectory_c1$x+0.15,weight_trajectory_c1$y+0.1, weight_trajec
tory_c1$z+0.01, text = weight_trajectory_c1$iteration,color="brown")
shapelist3d(tetrahedron3d(), weight_trajectory_c2$x, weight_trajectory_c2$y,
weight_trajectory_c2$z, size =  0.02,
            color = weight_trajectory_c2$color)
texts3d(weight_trajectory_c2$x+0.04,weight_trajectory_c2$y+0.04, weight_traje
ctory_c2$z+0.04, text = weight_trajectory_c2$iteration,color="brown")
shapelist3d(tetrahedron3d(), weight_trajectory_c3$x, weight_trajectory_c3$y,
weight_trajectory_c3$z, size =  0.02,
            color = weight_trajectory_c3$color)
texts3d(weight_trajectory_c3$x,weight_trajectory_c3$y-0.04, weight_trajectory
_c3$z+0.05, text = weight_trajectory_c3$iteration,color="brown")
shapelist3d(icosahedron3d(), centers$center_x, centers$center_y, centers$cent
er_z, size =  0.04,
            color = "#FFA500")
```

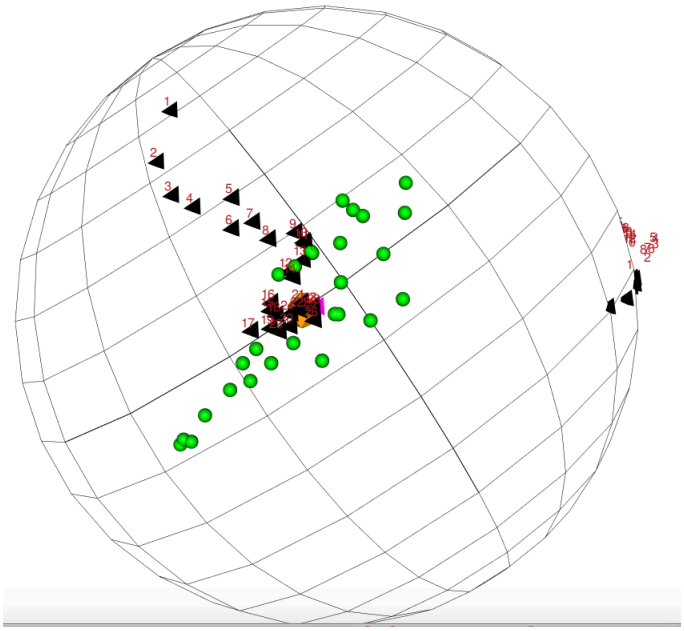


Here cluster 3 is shown above. Black triangles show the weight trajectory. The numbers show the iteration number. The purple triangle in the middle shows the final weight. The orange icosahedron shows the center of the cluster. It can be seen that the purple triangle and the orange icosahedron is very close.



Here, cluster 2 is shown above. Similarly, the black triangles show the weight trajectory. The orange icosahedron is the center of this cluster, and the purple triangle right in the

middle of the cluster is the final weight. Again, the center and the final weight vector is very close.



Now, cluster 1 is shown above. Similarly, the black triangles show the weight trajectory. The orange icosahedron is the center of this cluster, and the purple triangle right in the middle of the cluster is the final weight. Again, the center and the final weight vector is very close.

Below, which output neuron indicates which cluster is shown. They are found after calculating each output neuron's dot product to each cluster center. The maximum is chosen afterwards. It is the same as choosing the minimum distance one.

```
#weight_matrix-centers
weight_center=as.data.frame(as.matrix(centers[,2:4])%*%t(as.matrix(weight_mat
rix[,1:3])))
#weight_center
a=t(colnames(weight_center)[apply(weight_center,1,which.max)])
centers=cbind(centers,t(a))
#cluster 2:output3, cluster 1:output 2, cluster 3:output1
```

As a result, output neuron 1 indicates cluster 3 (from the first question) (red points), output neuron 2 indicates cluster 1 (green points) and output neuron 3 cluster 2 (blue points).

Question 5

In this question, 9 test sample patterns are tested. For each test instance, instance*t(weight_vector_of_each_neuron) is calculated. Here again, instance and weight vectors are row vectors. By saying weight vector, w_i is indicated which means weight

vector of output neuron i. Then the winning unit is determined by the maximum of these values.

```
output_tests=as.data.table(as.matrix(test[,1:3])%*%t(weight_matrix[,1:number_
of_inputs]))
setnames(output_tests,old=names(output_tests),new=c("output1","output2","outp
ut3"))
test=cbind(test,output_tests)
test_melted=melt(test,id.vars = c("x","y","z","cluster"))
test_melted_max=test_melted[,.(value=max(value)),by=.(x,y,z,cluster)]
test_melted=merge(test_melted,test_melted_max,by=c("x","y","z","cluster","val
ue"))
test_melted
```

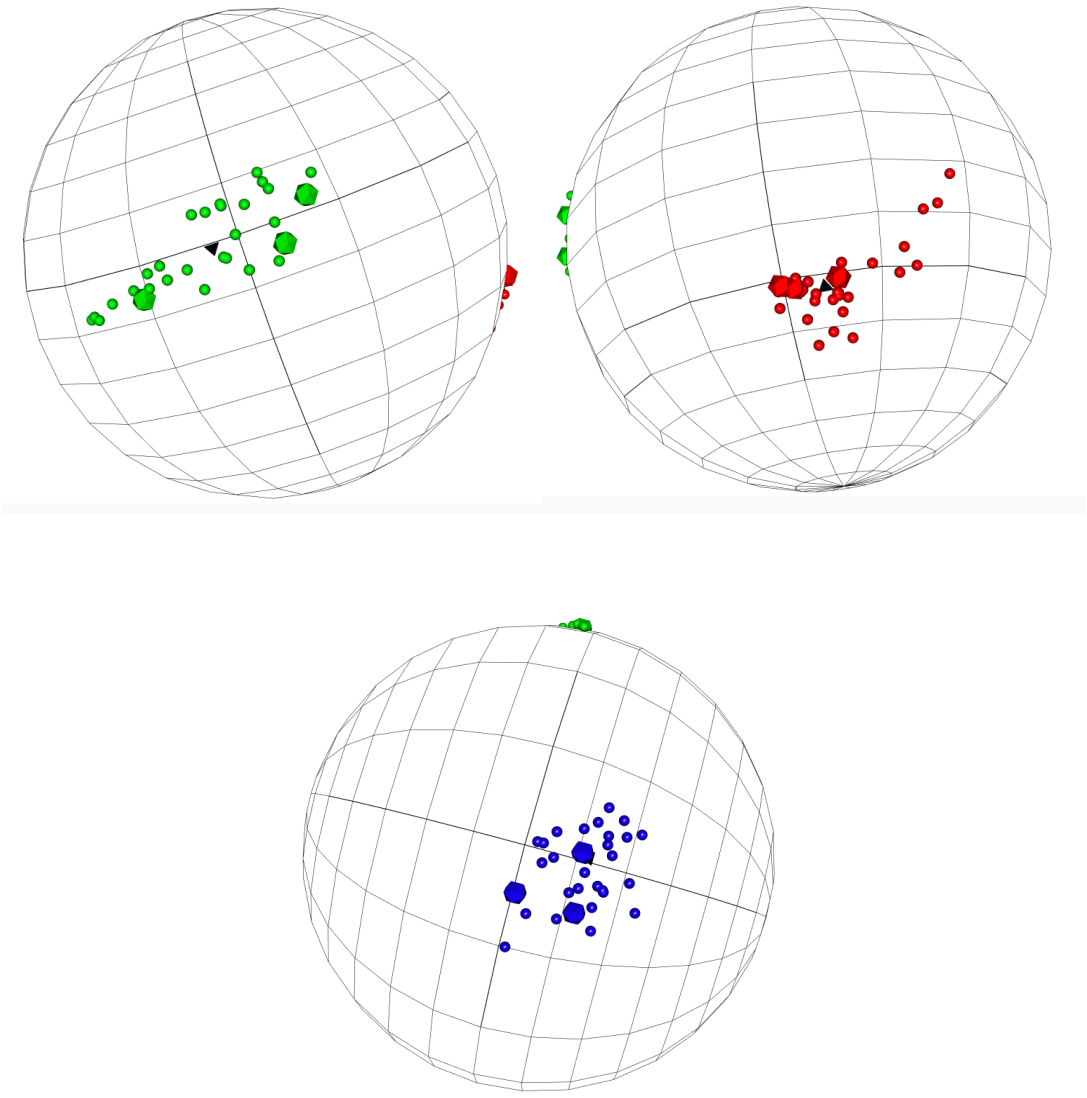
```
##             x           y          z cluster     value variable
## 1: -0.9771676 -0.01073469  0.2121987       2 0.9560693  output3
## 2: -0.9534893 -0.23756000  0.1855353       2 0.9778010  output3
## 3: -0.8921585 -0.21059847  0.3996267       2 0.9996369  output3
## 4:  0.3643060  0.07585081  0.9281852       1 0.9483454  output2
## 5:  0.4916859 -0.40393573 -0.7714148       3 0.9984161  output1
## 6:  0.6157936 -0.41849529 -0.6675777       3 0.9938464  output1
## 7:  0.6532978 -0.40385711 -0.6403916       3 0.9874249  output1
## 8:  0.7993376  0.11361998  0.5900424       1 0.9676311  output2
## 9:  0.8494089  0.24762169  0.4660344       1 0.9273088  output2
```

As it can be seen from the previous question (center-weight), output3 indicates cluster 2, output1 indicates cluster 3 and output2 indicates cluster 1. Hence, the test set is correctly clustered when variable and cluster columns are compared.

The test set is demonstrated below. The training data is shown in green, blue and red points. The test data is shown in icosahedrons with the corresponding cluster colors green (cluster 1-output2), blue (cluster 2- output3) and red (cluster 3- output1). The tetrahedrons show the final weights.

```
weight_matrix=as.data.table(weight_matrix)
spheres3d(0,0,0,lit=FALSE,color="white")
spheres3d(0,0,0,radius=1,lit=FALSE,color="black",front="lines")
spheres3d(cluster1$x,cluster1$y,cluster1$z,col="green",radius=0.02)
spheres3d(cluster2$x,cluster2$y,cluster2$z,col="blue",radius=0.02)
spheres3d(cluster3$x,cluster3$y,cluster3$z,col="red",radius=0.02)
shapelist3d(icosahedron3d(), test_melted[variable=="output2"]$x, test_melted[
variable=="output2"]$y, test_melted[variable=="output2"]$z, size =  0.04,
            color = "green") #cluster1
shapelist3d(icosahedron3d(), test_melted[variable=="output3"]$x, test_melted[
variable=="output3"]$y, test_melted[variable=="output3"]$z, size =  0.04,
            color = "blue") #cluster2
shapelist3d(icosahedron3d(), test_melted[variable=="output1"]$x, test_melted[
variable=="output1"]$y, test_melted[variable=="output1"]$z, size =  0.04,
            color = "red") #cluster3
par3d(windowRect = c(20, 30, 1000, 1000))
```

```
shapelist3d(tetrahedron3d(), weight_matrix$x, weight_matrix$y, weight_matrix$
z, size =  0.02)
```



Here, the test set is shown with the training set. The dots are the training samples. The icosahedrons in green, red and blue are the test set. Their color is given according to the winner unit when that sample pattern is fed to the network. Thus, it can be seen that they are clustered correctly. The black triangles are the final weights. In the final plot, one of the blue test sample is very close to the final weight, so the black triangle cannot be seen easily.