Semih Akbayrak
20.05.2015

EE550

HW5 Report

In this project, a classification by using unsupervised learning model is expected from us. Competetive learning model is the method we has to use.

Firstly, as it is expected I defined 90 sample data which are divided into three categories. In order to define these vectors I used angle approximation. Two angle variable, alfa from 0 to 2*pi and beta from 0 to pi will be enough to describe all the points in the sphere. In order to classify 3 class from 90 point in the sphere I took some parts of the angles for each classes, and then I drawed 30 random points from each surface part.

```
figure
sphere
hold on
x=[];
y=[];
z=[];
for i=1:90
  if i<=30          %for data vectors in the first category
    min1 = 0;
    max1 = 2/20;
    min2 = 13/20;
    max2 = 15/20;
  elseif i<=60      %for data vectors in the second category
    min1 = 3/20;
    max1 = 5/20;
    min2 = 18/20;
    max2 = 20/20;
  else              %for data vectors in the third category
    min1 = 6/20;
    max1 = 8/20;
```

```
        min2 = 13/20;
        max2 = 15/20;
    end
    alfa = 2*pi*((max1-min1)*rand+min1); %alfa and beta angles to define x,y,z compounds of datas
    beta = pi*((max2-min2)*rand+min2);
    z(i) = sin(beta);
    x(i) = cos(beta)*cos(alfa);
    y(i) = cos(beta)*sin(alfa);
    plot3(x,y,z,'bo')
end
```

Then I drawed 3 weight points in the surface randomly and plotted them as well.

%drawing 3 weights with length 1 randomly

```
w1=[];
w2=[];
w3=[];
alfa = 2*pi*rand(1,3);
beta = pi*rand(1,3);
w1(3) = sin(beta(1));
w1(1) = cos(beta(1))*cos(alfa(1));
w1(2) = cos(beta(1))*sin(alfa(1));
w2(3) = sin(beta(2));
w2(1) = cos(beta(2))*cos(alfa(2));
w2(2) = cos(beta(2))*sin(alfa(2));
w3(3) = sin(beta(3));
w3(1) = cos(beta(3))*cos(alfa(3));
w3(2) = cos(beta(3))*sin(alfa(3));
plot3(w1(1),w1(2),w1(3),'m*','markers',50)
hold on
plot3(w2(1),w2(2),w2(3),'m*','markers',50)
hold on
plot3(w3(1),w3(2),w3(3),'m*','markers',50)
hold on
```

Let me take the learning coefficient as 2.

```
n = 0.2; %learning coefficient
```

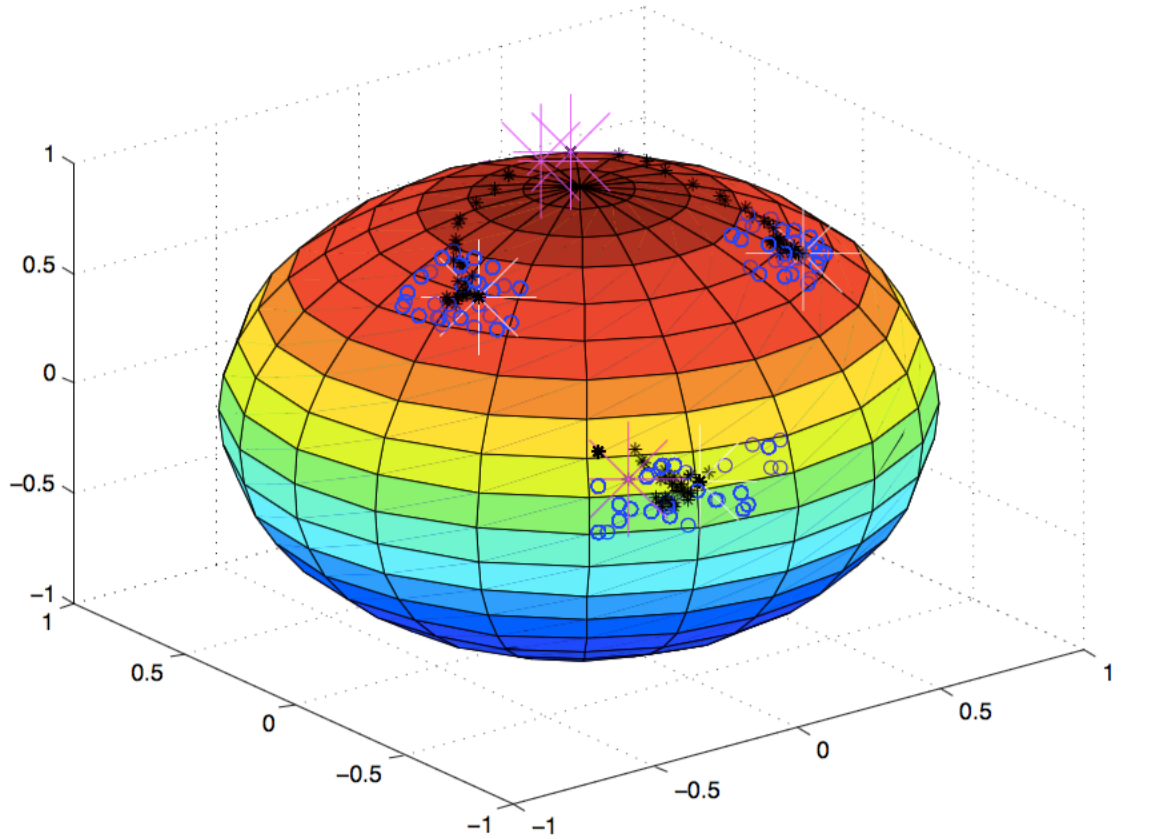
Then here is the main algorithm part. In this part, I multiply each input with the weights to find the outputs. Then I compare the outputs with each other in order to find the winner output and according to winner output I update the corresponding weight. This process maintains for 90 inputs. In the end I reach the final weights.

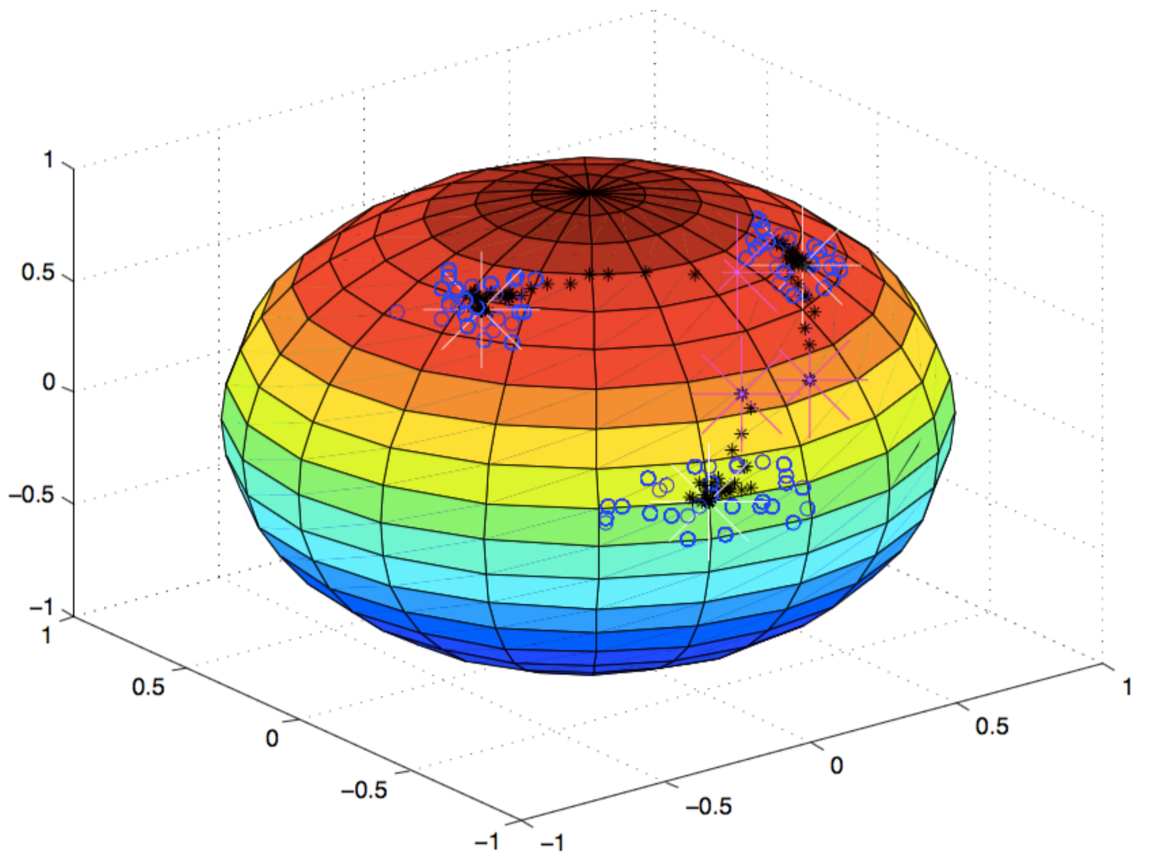
```
%main algorithm
for i=1:90
    o=[0 0 0];
    zay = [x(i) y(i) z(i)];
    %3 output values
    o(1) = x(i)*w1(1) + y(i)*w1(2) + z(i)*w1(3);
    o(2) = x(i)*w2(1) + y(i)*w2(2) + z(i)*w2(3);
    o(3) = x(i)*w3(1) + y(i)*w3(2) + z(i)*w3(3);
    %determination of winner output and weight updating with normalization
    if o(1)>o(2)
        if o(1)>o(3)
            deltaw = n*(zay-w1);
            w1 = w1 + deltaw;
            w1 = w1/sqrt(w1(1)^2+w1(2)^2+w1(3)^2)
        else
            deltaw = n*(zay-w3);
            w3 = w3 + deltaw;
            w3 = w3/sqrt(w3(1)^2+w3(2)^2+w3(3)^2)
        end
    else
        if o(2)>o(3)
            deltaw = n*(zay-w2);
            w2 = w2 + deltaw;
            w2 = w2/sqrt(w2(1)^2+w2(2)^2+w2(3)^2)
        else
            deltaw = n*(zay-w3);
            w3 = w3 + deltaw;
            w3 = w3/sqrt(w3(1)^2+w3(2)^2+w3(3)^2)
        end
    end
    %plotting weight updates
    plot3(w1(1),w1(2),w1(3),'k*')
    hold on
    plot3(w2(1),w2(2),w2(3),'k*')
    hold on
    plot3(w3(1),w3(2),w3(3),'k*')
    hold on
end
%final weights in the plot
plot3(w1(1),w1(2),w1(3),'w*','markers',50)
hold on
plot3(w2(1),w2(2),w2(3),'w*','markers',50)
```

hold on

plot3(w3(1),w3(2),w3(3),’w*','markers',50)

Below I attached two different simulation plots.





Blue points indicates the input datas. Purple stars are initial weights, whereas white ones are final weights. Little black stars shows the convergence path.

If initial weights are very far away from the class centers, sometimes weights converge to wrong points but in general algorithm is able to find the centers of the clusters.