Onur Poyraz
2010401036
20 May 2015

Homework V

In this project I deal with the continuous Competitive Learning Model. In this model there is an output neurons and there is only one of them fire at a given data set. The highest the dot product of input vector and weight will be burn. This system is named as a winner take all circuits. Actually I use this system in the Continuous Hopfield Model Iris data set part. In there, I define three nodes output unit and according the firing neuron I decide the type of the vector.

In this project first of all I define 90 inputs. 30 of them is in first quadrant, 30 of them in third quadrant and 30 of them is in eighth quadrant. In addition to them I define randomly selected weight vectors. Before the algorithm I normalize all of them.

```
%% Normalized random inputs
input=zeros(90,3);
for u=1:90
   n=0;
   for i=1:3
      if u<=30
         input(u,i)=rand/10+0.45; % Generate random inputs between 0.45 and 0.55(1st quadrant)
         n=n+input(u,i)^2; % Normalization coefficient (sum of squares of inputs)
      elseif u>=31 && u<=60
         input(u,i)=-rand/10-0.45; % Generate random inputs between -0.45 and -0.55(8th quadrant)
         n=n+input(u,i)^2; % Normalization coefficient (sum of squares of inputs)
      elseif u>=61
         if i==1 || i==3
            input(u,i)=rand/10+0.45; % Generate random inputs between 0.45 and 0.55
            n=n+input(u,i)^2; % Normalization coefficient (sum of squares of inputs)
         else
            input(u,i)=-rand/10-0.45; % Generate random inputs between -0.45 and -0.55(3rd quadrant)
            n=n+input(u,i)^2; % Normalization coefficient (sum of squares of inputs)
         end
      end
   end
   input(u,:)=input(u,:)./sqrt(n); % Normalization of the input
end
%% Normalized random weights
w=zeros(3,3);
for i=1:3
   n=0;
   for j=1:3
      w(i,j)=rand-0.5; % Generate random inputs between -0.5 and 0.5
      n=n+w(i,j)^2; % Normalization coefficient (sum of squares of each weights connected to the one output neuron)
   end
   w(i,:)=w(i,:)./sqrt(n); % Normalization of the weights
end
```

After doing that I plot the initial weights on the sphere. I use different color for each neuron. The cross sign(x) represents the initial randomly selected weight vectors. But because of I completely select this weights randomly sometimes one of the neurons can be die.

```
%% Plotting İnitial Weights
plot3(w(1,1),w(1,2),w(1,3),'mx'); hold on;
plot3(w(2,1),w(2,2),w(2,3),'kx'); hold on;
plot3(w(3,1),w(3,2),w(3,3),'rx'); hold on;
```

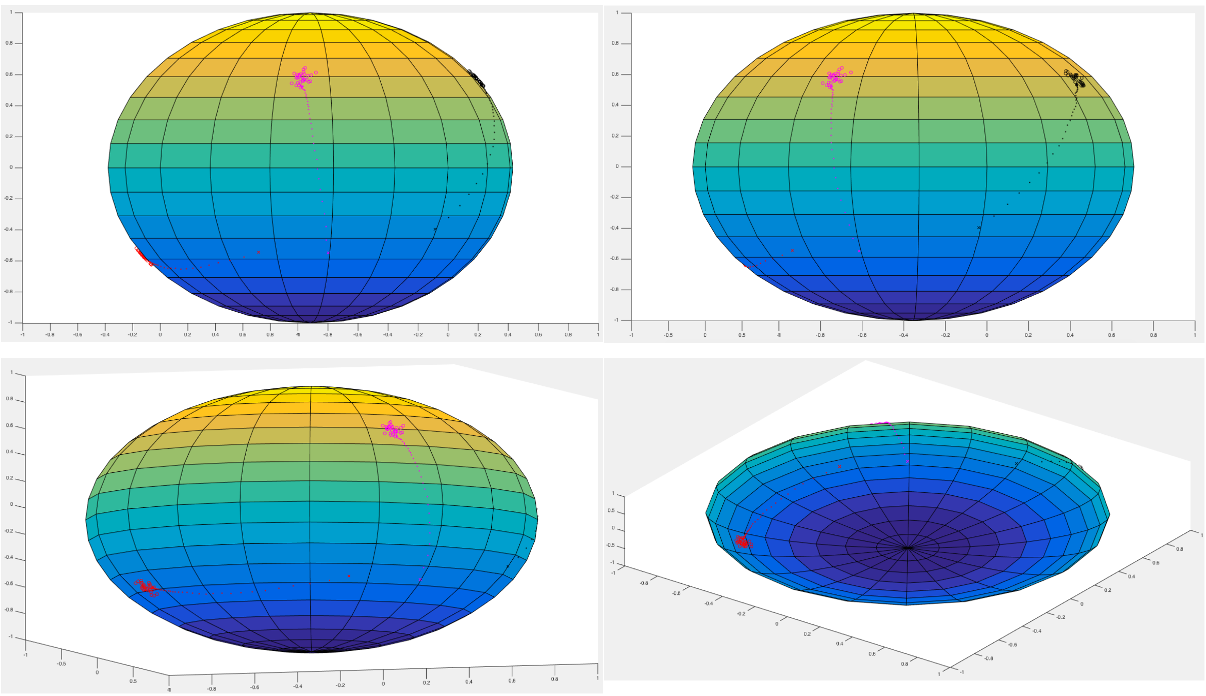
From now on I start to apply algorithm. First of all using randperm function I change the order of the inputs randomly. After that I started to train my network. For each sample vector I calculate the sum of dot products and I decided which neuron will be win. After deciding of the winner I apply weight update algorithm to the weights which are connected to the winner neuron and I normalize new weights again. In here the algorithm is finished and I draw the new weights as a dot(.) on the sphere after the training of the each new sample vector. In other words, from cross(x) to star(*) the dots represents the change of weights after each step.

```
%% Algorithm Part
index=randperm(90);
for k=1:90
    u=index(k);
    for i=1:3
        for j=1:3
            output(u,i)=output(u,i)+input(u,j)*w(i,j); %% Calculation of the output
        end
    end
    %% Determinening of which neuron will be win
    if output(u,1)>=output(u,2) && output(u,1)>=output(u,3)
        o(u,1)=1;
        o(u,2)=0;
        o(u,3)=0;
    elseif output(u,2)>=output(u,1) && output(u,2)>=output(u,3)
        o(u,1)=0;
        o(u,2)=1;
        o(u,3)=0;
    else
        o(u,1)=0;
        o(u,2)=0;
        o(u,3)=1;
    end
    %% Weight Update Part
    for i=1:3
        n=0;
        for j=1:3
            delta_w=o(u,i)*lf*(input(u,j)-w(i,j)); % Calculation of the weight update
            w(i,j)=w(i,j)+delta_w; % New weight
            n=n+w(i,j)^2; % Normalization coefficient for the new weight
        end
        w(i,:)=w(i,:)./sqrt(n); % Normalization of the weight
    end
    %% Plotting weights after each step
    if u ~= 90
        plot3(w(1,1),w(1,2),w(1,3),'m.'); hold on;
        plot3(w(2,1),w(2,2),w(2,3),'k.'); hold on;
        plot3(w(3,1),w(3,2),w(3,3),'r.'); hold on;
    end
end
```

From now on I finish the code and I start the draw everyone else. First of all I draw unit sphere. After that I draw the inputs according to the which neuron they activated.The sign of (o) represents this inputs. Finally I draw the last weights. As a mentioned before I star sign(*) represent the final position of weight vector.

```
%% Plotting of the inputs on the unit sphere
for u=1:90
   if o(u,1)==1
      plot3(input(u,1),input(u,2),input(u,3),'mo');
   elseif o(u,2)==1
      plot3(input(u,1),input(u,2),input(u,3),'ko');
   else
      plot3(input(u,1),input(u,2),input(u,3),'ro');
   end
   hold on;
end
%% Plotting for the final weights with different mark
plot3(w(1,1),w(1,2),w(1,3),'m*'); hold on;
plot3(w(2,1),w(2,2),w(2,3),'k*'); hold on;
plot3(w(3,1),w(3,2),w(3,3),'r*'); hold on;
```

Now I will put the output graphs of the system.



Figures shows the output of the systems. The convergence of the weight vectors to the clusters can be seen from this graphs.