```
%%%%%%%%%%%%%%%%%%%%%%%%%    Main Program Hebb.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


moption=input('\nDesea entrenar una nueva red (Y) o usar una red ya entrenada(N)? Y/N:','s');

if moption=='Y'

OS=input('\nDesea utilizar el método de Oja o de Sanger? O/S :','s');
nu=input('\nEnter the value of the learning rate:');
filename=input('\nEnter the name of the csv file with the training data set (must include the
extension and have the same format as the files given for TP2, for example:
tp1_training_dataset_2.csv):','s');
P2p=input('\nDo you want to use all the data in the file for training (Y) or preserve some some for
validation (N)? Y/N (if you choose Y you have the chance to validate the network later using another
data file):' , 's');
   if P2p=='N'
       P2=input('\nHow many cases do you want to use for training? (enter the number of cases):');
    end
Hebb_supp

else
    disp('Se ha cargado una red para ser testeada.')
  filename=input('\nIngrese el nombre del archivo csv con la data para el testeo (debe incluir la
extensión, por ejemplo: tp1_training_dataset_2.csv):','s');

Hebb_supp

end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Hebb_supp.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%nu=0.08;
%filename= ;
Xo=csvimport(filename,'columns',[2:857],'noHeader', true);
No=csvimport(filename,'columns',[1:1],'noHeader', true);

Q=3;
[P sx]=size(Xo);

for i=1:sx
%xi=Xo{:,i};
meanx= mean(Xo(:,i));
%varx=var(Xo(:,i));
Xm(:,i)=(Xo(:,i)-meanx);%/(varx);
meanx= 0;
%varx=0;
end


if moption=='Y'

if P2p=='Y'
    P2=P;
   casomin=1;
end



W= unifrnd(-sx^(-0.5),sx^(-0.5),3,sx);

for t=1:5000
     DW=0;
for i=1:P2
    xi=transpose(Xm(i,:));
    Y=W*xi;

for j=1:3
      for k=1:sx
            X2(k)=0;
            if OS=='O'
            for l=1:Q
```

```
                X2(k)=X2(k) + Y(l,1)*W(l,k);
            end
            elseif OS=='S'
                for l=1:j
                X2(k)=X2(k) + Y(l,1)*W(l,k);
                end
            end
            DW(j,k)=nu*(xi(k,1)-X2(k))*Y(j,1);

            end
      end
    normDW(t)=norm(DW,'fro');
    W=W+DW;
end
end


C{1}=[1 1 0];C{2}=[1 0 1];C{3}=[0 1 1];C{4}=[1 0 0];C{5}=[0 1 0];
C{6}=[0 0 1];C{7}=[1 1 0.6];C{8}=[0 0 0];C{9}=[1 0.6 0];

figure()
for i=1:P2
    xi=transpose(Xm(i,:));
    Y=W*xi;
    scatter3(Y(1,1),Y(2,1),Y(3,1),'MarkerFaceColor',C{No(i,1)},'MarkerEdgeColor',C{No(i,1)}); hold on
end
title('3D-Output-Training','FontSize', 12)
xlabel('x','FontSize', 12)
ylabel('y','FontSize', 12)
zlabel('z','FontSize', 12)
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 6 5])
print('3Doutput_training','-dpng','-r300')

if P2p=='N'
figure()
for i=P2+1:P
    xi=transpose(Xm(i,:));
    Y=W*xi;
    scatter3(Y(1,1),Y(2,1),Y(3,1),'MarkerFaceColor',C{No(i,1)},'MarkerEdgeColor',C{No(i,1)}); hold on
end
title('3D-Output-Validation','FontSize', 12)
xlabel('x','FontSize', 12)
ylabel('y','FontSize', 12)
zlabel('z','FontSize', 12)
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 6 5])
print('3Doutput-Validation','-dpng','-r300')
end




figure()
plot(1:t,normDW);
title('Training Error: normDW(t)','FontSize', 12)
xlabel('# of steps','FontSize', 12)
ylabel('normDW','FontSize', 12)
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 6 5])
print('Err_train','-dpng','-r300')

figure()
semilogx(1:t,normDW);
xlabel('# of steps','FontSize', 12)
ylabel('normDW','FontSize', 12)
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 6 5])
print('Err_train_log','-dpng','-r300')

save=input('\nDesea guardar la matriz de pesos entrenada (OJO:reescribirá la preexistente)? Y/
N :','s');
if save =='Y'
    dlmwrite('W.dat',W,'delimiter',' ');
end

disp(' ')
disp('Fin del programa.')
```

```
else

    W=dlmread('W.dat');

  C{1}=[1 1 0];C{2}=[1 0 1];C{3}=[0 1 1];C{4}=[1 0 0];C{5}=[0 1 0];
  C{6}=[0 0 1];C{7}=[1 1 0.6];C{8}=[0 0 0];C{9}=[1 0.6 0];

figure()
for i=1:P
    xi=transpose(Xm(i,:));
    Y=W*xi;
    scatter3(Y(1,1),Y(2,1),Y(3,1),'MarkerFaceColor',C{No(i,1)},'MarkerEdgeColor',C{No(i,1)}); hold on
end
title('3D-Output-Validation','FontSize', 12)
xlabel('x','FontSize', 12)
ylabel('y','FontSize', 12)
zlabel('z','FontSize', 12)
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 6 5])
print('3Doutput-Validation','-dpng','-r300')

disp(' ')
disp('Fin del programa.')

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%    Main Program Som.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



moption=input('\nDesea entrenar una nueva red (Y) o usar una red ya entrenada(N)? Y/N:','s');

if moption=='Y'
disp('Este programa genera un mapa autoorganizado de MxM mediante el método de Kohonen')

M1=input('\nIngrese el valor de M deseado:');
nsteps=input('\nIngrese número de pasos de entrenamiento:');

nu=input('\nEnter the value of the learning rate:');
filename=input('\nEnter the name of the csv file with the training data set (must include the extension, for example: tp1_training_dataset_2.csv):','s');

P2p=input('\nDo you want to use all the data in the file for training (Y) or preserve some some for validation (N)? Y/N:' , 's');
   if P2p=='N'
       P2=input('\nHow many cases do you want to use for training? Enter the number of cases(the rest of the cases will be used automatically for validation):');
    end
som_supp

else
   disp(' ')
  disp('Se ha cargado una red de 10x10 para ser testeada.')
  filename=input('\nIngrese el nombre del archivo csv con la data para el testeo (debe incluir la extensión, por ejemplo: tp1_training_dataset_2.csv):','s');
 M1=10;

 som_supp
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Som_supp.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%nsteps=100;
sigmao=2;
%filename= ;
Xo=csvimport(filename,'columns',[2:857],'noHeader', true);
No=csvimport(filename,'columns',[1:1],'noHeader', true);
%M=25;
M2=M1;
M=M1*M1;
[Pc sx]=size(Xo);
PP=Pc;
casomin=1;


if moption=='Y'

if P2p=='N'
    Pc=P2;
    casomin=P2+1;
end


W= unifrnd(-sx^(-0.5),sx^(-0.5),sx,M);


for t=1:nsteps
    DWt=0;
    for caso=1:Pc
for i=1:M
Wx(:,i)= transpose(Xo(caso,:))-W(:,i);
Ybar(i)=norm(Wx(:,i));
end
[MM,jmin] = min(Ybar);
for j=1:M
P{j}=[j/M2, mod(j,M2)];
end
DW=zeros(sx,M);
sigma=sigmao/(1+(t-1)*0.1);
for j=1:M
    D(j)=exp(-norm(P{j}-P{jmin})^2/sigma);
    DW(:,j)=nu*D(j).*Wx(:,j);
end
W= W+DW;
DWt=DWt+norm(DW,'fro');
    end
normDWt(t)=DWt/Pc;
end

figure()
plot(1:nsteps,normDWt);
title('Training Error: normDW(t)','FontSize', 12)
xlabel('# of steps','FontSize', 12)
ylabel('normDW','FontSize', 12)
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 6 5])
print('Err_train','-dpng','-r300')



else
    W=dlmread('W.dat');
    casomin=1;
end



Scont=zeros(M1,M1,9);
 for caso=casomin:PP
    for j=1:M
    Wx(:,j)= transpose(Xo(caso,:))-W(:,j);
    Ybar(j)=norm(Wx(:,j));
    end
    [MM,jmin] = min(Ybar);
    a=ceil(jmin/M2);
    if mod(jmin,M2)==0
    b=5;
```

```
    else
    b=mod(jmin,M2);
    end
    %S(a,b)=No(caso);

    Scont(a,b,No(caso))=Scont(a,b,No(caso)) +1;


 end

 S=zeros(M1,M1);
 for a=1:M1
     for b=1:M1
     [MMM,Nomax] = max(Scont(a,b,:));
     S(a,b)= Nomax;
     end
 end

figure()
colormap('redbluecmap');   % set colormap
imagesc(S);        % draw image and scale colormap to values range
colorbar;
title('Mapa','FontSize', 9)
set(gca,'FontSize',9)
set(gcf,'PaperUnits','centimeters','PaperPosition',[0 0 6 5])
print('Mapa','-dpng','-r300')

if moption== 'Y'
    save=input('\nDesea guardar la matriz de pesos entrenada (OJO:reescribirá la preexistente)? Y/
N :','s');
if save =='Y'
    dlmwrite('W.dat',W,'delimiter',' ');
end
end

disp(' ')
disp('Fin del programa.')
```