```
function elec502hw7()
global number_of_steps
global size_of_dataset
global dim
global SOMD1
global SOMD2
global min_learn_rate
global spatial_decay_rate
global initial_learn_rate
global learn_decay_rate
global min_spatial_decay_rate
global spatial_decay_power
%global smallest_neighbourhood
number_of_steps=200000;
size_of_dataset=150;
dim=2;
SOMD1=10; % Number of Rows
SOMD2=10; % Number of Columns
epsilon=eps;
initial_learn_rate=1/sqrt(SOMD1*SOMD2);
min_learn_rate=0.0006;
learn_decay_rate=0.0003;
spatial_decay_rate=(6)/(SOMD1*SOMD2); % The functions I use:
%alpha=initial_learn_rate*e^(-t*learn_decay_rate) but,
%if this is lessthan the min learn rate, the min learn rate is used as
 alpha instead
spatial_decay_power=0.01;
min_spatial_decay_rate=0.1;
% h(horizontal distance,vertical distance, t)=
%e^(-(1+t*spatial_decay_rate)*(1-
max(h_dist,v_dist)*spatial_decay_power))
% IF YOU'LL FEEL THESE FUNCTIONS AREN'T OPTIMAL, FEEL FREE TO CHANGE
 THEM
steps_to_see=[100 500 1000 2000 5000 10000 20000 25000 50000 75000
 100000 125000 150000 200000]; % Steps at which plots are produced
mdif=15;
mmin=0.4; %mdif and mmin relate to the marker width when plotting the
 SOM visualisation.
% mmin is the min thickness, mdif is the scaling with distance between
% weight vectors of consecutive PEs.
%smallest_neighbourhood=sqrt(dim)*sqrt(SOMD1*SOMD2);
% Generate the data
%X=rand(dim,size_of_dataset);
load fisheriris.mat
X=meas';
X=X/(max(max(X)));
y=species;
% Initialise the weights
W=rand(SOMD1,SOMD2,dim);
dist1=0*W;
% Perform the learning
for learning_step=1:number_of_steps
```

```
    % Choose the sample
    sample=round(rand(1)*size_of_dataset+0.5-epsilon);
    % vect is the corresponding vector
    vect=X(:,sample);
    for i=1:SOMD1
        for j=1:SOMD2
            %disp(vect)
            %disp(dist1(i,j,:))
            for k=1:dim
                dist1(i,j,k)=W(i,j,k)-vect(k);
            %dist1(i,j,2)=W(i,j,2)-vect(2);
            end
        end
    end
    % Having constructed a matrix of positions with respect to the
 chosen
    % vector we find distances.
    dist=dist1.^2;
    dist=sum(dist,3);
    [Y,I]=min(dist,[],1);
    [~,I2]=min(Y);
    W=weight_update(W,I2,I(I2),learning_step,dist1);
    if ismember(learning_step,steps_to_see)
        figure
        hold on
        %for r=0:SOMD1
        %    plot([0 1],[r/SOMD1 r/SOMD1],'k')
        %end
        %for r=0:SOMD2
        %    plot([r/SOMD2 r/SOMD2],[0 1],'k')
        %end
        % The loops below plot the SOM image
        for r=1:SOMD1
            for r2=1:SOMD2-1
                vectval=W(r,r2,:)-W(r,r2+1,:);
                vectval=sqrt(sum(vectval.^2));
                plot([r2/SOMD2 r2/SOMD2],[1-(r-1)/SOMD1 1-r/
SOMD1],'k','LineWidth',(mdif*vectval+mmin))
            end
        end
        for r2=1:SOMD2
            for r=1:SOMD1-1
                vectval=W(r,r2,:)-W(r+1,r2,:);
                vectval=sqrt(sum(vectval.^2));
                plot([1-(r2-1)/SOMD2 1-r2/SOMD2],[1-r/SOMD1 1-r/
SOMD1],'k','LineWidth',(mdif*vectval+mmin))
            end
        end
        title(['Visualisation of SOM at t=',num2str(learning_step)])
        %K=W(:,:,1)
        %K2=W(:,:,2)

        %K=reshape(K,1,100);
        %K2=reshape(K2,1,100);
```

```
        %scatter(K,K2)
        %hold on
        %for r=1:SOMD1
        %    vect1=K(r,:);
        %    vect2=K2(r,:);
        %    plot(K(r,:),K2(r,:))
        %end
    end
    %if mod(learning_step,5000)==0
    %    disp(learning_step)
    %end
end
i1=1:50;
i2=51:100;
i3=101:150;
m1=meas(i1,:);
m2=meas(i2,:);
m3=meas(i3,:);
Y1=zeros(10);
Y2=Y1;
Y3=Y2;
for sample=1:150
    vect=X(:,sample);
    for i=1:SOMD1
        for j=1:SOMD2
            %disp(vect)
            %disp(dist1(i,j,:))
            for k=1:dim
                dist1(i,j,k)=W(i,j,k)-vect(k);
            %dist1(i,j,2)=W(i,j,2)-vect(2);
            end
        end
    end
    % Having constructed a matrix of positions with respect to the
 chosen
    % vector we find distances.
    dist=dist1.^2;
    dist=sum(dist,3);
    [Y,I]=min(dist,[],1);
    [~,I2]=min(Y);
    if sample<=50
        Y1(I2,I(I2))=Y1(I2,I(I2))+1;
    end
    if sample>50 && sample <=100
        Y2(I2,I(I2))=Y2(I2,I(I2))+1;
    end
    if sample>100
        Y3(I2,I(I2))=Y3(I2,I(I2))+1;
    end
end
figure
imagesc(Y1)
title('setosa')
figure
```

```
imagesc(Y2)
title('virginica')
figure
imagesc(Y3)
title('versicolor')
K=W(:,:,1);
K2=W(:,:,2);
K=reshape(K,1,100);
K2=reshape(K2,1,100);
%disp(size(K))
%disp(size(K2))
%scatter(K,K2)
disp(W)
end


function W=weight_update(W,hpos,vpos,t,dist1)
W=W+activation_func(hpos,vpos,t).*(-dist1);
end


function val=activation_func(hpos,vpos,t)
global number_of_steps
global size_of_dataset
global dim
global SOMD1
global SOMD2
global min_learn_rate
global spatial_decay_rate
global initial_learn_rate
global learn_decay_rate
global min_spatial_decay_rate
global spatial_decay_power
alpha=max(initial_learn_rate*exp(-t*learn_decay_rate),min_learn_rate);
for i=1:SOMD1
    for j=1:SOMD2
        for k=1:dim
            val(i,j,k)=exp((1-spatial_decay_power*max(abs(j-
hpos),abs(i-vpos)))*-(1+t*spatial_decay_rate));
        end
    end
end
val(vpos,hpos,1:2)=[1 1];
val=alpha*val;
val=min(val,1);
end
```

```
(:,:,1) =

  Columns 1 through 7

    0.4621    0.4193    0.8394    0.5311    0.7781    0.6615    0.7735
    0.4693    0.5976    0.6541    0.6674    0.5271    0.6240    0.6829
```

```
    0.8232    0.4386    0.4524    0.5410    0.4166    0.7369    0.8674
    0.3927    0.8472    0.5139    0.6342    0.6335    0.7818    0.9507
    0.4952    0.3860    0.7300    0.3769    0.3611    0.3486    0.5823
    0.5082    0.4668    0.6225    0.7658    0.6877    0.7901    0.3940
    0.4535    0.3974    0.8038    0.7026    0.6835    0.7113    0.7209
    0.7676    0.5108    0.3849    0.7328    0.7922    0.5357    0.4002
    0.6139    0.4929    0.6816    0.6593    0.4022    0.4984    0.6202
    0.5889    0.4103    0.6441    0.6067    0.8464    0.8424    0.6805

  Columns 8 through 10

    0.4795    0.4797    0.7152
    0.5682    0.7016    0.5610
    0.5634    0.6708    0.6772
    0.5498    0.6094    0.6440
    0.7942    0.4264    0.8157
    0.6956    0.3893    0.6456
    0.4341    0.5942    0.3956
    0.5685    0.7278    0.4056
    0.4024    0.3893    0.8277
    0.7583    0.3945    0.9623


(:,:,2) =

  Columns 1 through 7

    0.5816    0.5228    0.4134    0.4443    0.6959    0.6829    0.7000
    0.3566    0.5843    0.5181    0.6199    0.5354    0.1889    0.6370
    0.6440    0.2114    0.4829    0.6801    0.1923    0.2627    0.3899
    0.5443    0.3155    0.5713    0.2892    0.2540    0.6212    0.3633
    0.4998    0.3675    0.6740    0.3851    0.3645    0.4730    0.4432
    0.6672    0.6056    0.4433    0.3723    0.4345    0.6125    0.4902
    0.2298    0.4931    0.6138    0.6646    0.4937    0.5792    0.5578
    0.2785    0.6213    0.3683    0.3404    0.3184    0.6206    0.5435
    0.3912    0.3199    0.5463    0.3414    0.2516    0.3153    0.3100
    0.5580    0.3117    0.3184    0.2178    0.2103    0.5087    0.3789

  Columns 8 through 10

    0.4821    0.4789    0.3672
    0.2904    0.3114    0.5164
    0.3884    0.2419    0.4684
    0.4911    0.6389    0.4317
    0.4194    0.7034    0.3667
    0.5320    0.5745    0.4778
    0.5698    0.6874    0.4783
    0.5838    0.4939    0.6025
    0.4282    0.4012    0.5872
    0.6879    0.6830    0.4727
```