```
%%% solution of travel salesperson problem using SOM Neural Network(for 6 cities)%%%
%%%%%%%%%%%%Assignment #2.3%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%Student Name: Farhad  Mohammad Kazemi%%%%%%%%%%%%%%%%

clc;
clear all;
tspdata=[.9 .5;.6 .9;.2 .7;.1 .2;.4 .4;.7 .1];% tspdata  --  the position of cities

winit=[0 .3;.9 .8;.5 .5;.6 1;.4 .1;.1 .7];% the weights between input layer and output↙
layer

tt=cputime;
% initialize parameters
gama=0.03;          % decrease rate of gain(=0.003/0.03)
                    % the larger, the qicker, but the smaller accuracy
alpha=0.5;          % learning rate (=0.1/0.05)
gain=10;            % initial gain
percent=0.2;        % percent of neighborhood (=0.2/0.15/0.1)

nsize=size(tspdata);    % get the number of cities
ncity=nsize(1);
m=ncity;                % number of nodes on the ring (m=ncity)

% get the center point
%datanew=tspdata(:,2:3);
datanew=tspdata(1:6,:);
maxv=max(datanew);
minv=min(datanew);
maxvalue=maxv(1)*maxv(1)+maxv(2)*maxv(2);
wcenter=(maxv-minv)/2+minv;

% initialize weights to the center point
w=[winit(:,1)*wcenter(1),winit(:,2)*wcenter(2)];
wold=w;

niter=1;
while 1>0,
   inhibit = zeros(m);  % reset the inhibitation status of all nodes

% show situation every iteration
    plot(datanew(:,1),datanew(:,2),'ko','MarkerFaceColor','r');
        hold on;
     plot(w(:,1),w(:,2),'.-');
     plot([w(1,1) w(m,1)],[w(1,2) w(m,2)],'.-');
     pause(0.0001);
     hold off;

    % rand input
   yrand=randperm(ncity);

   % read input
```

```
   for pattern=1:ncity
      newidx=yrand(pattern);
      a=datanew(newidx,:);

      % calculate distance between nodes(1..m) and input node a(1,2)
      for j=1:m
         if inhibit(j) == 1
            T(j) = maxvalue;
         else
            T(j) = (a(1)-w(j,1))^2 + (a(2)-w(j,2))^2;
         end;
      end

      [Tmin,Jmin] = min(T);
      inhibit(Jmin)=1;

      % when m<>ncity, y is not useful
      y(Jmin)=newidx;

      f = zeros(1,m);
      for j=1:m
         d=min(abs(j-Jmin),m-abs(j-Jmin));
         if d < percent*m
            f(j)=exp(-d*d/(gain*gain));
            w(j,1)=w(j,1)+alpha*f(j)*(a(1)-w(j,1));
            w(j,2)=w(j,2)+alpha*f(j)*(a(2)-w(j,2));
            %w(j,1)=w(j,1)+alpha*(a(1)-w(j,1));
            %w(j,2)=w(j,2)+alpha*(a(2)-w(j,2));
         end;
      end;

      % draw the winner to the node if the distance less than 0.01*mindst(0.01-0.001)
      distJ=sqrt((a(1)-w(Jmin,1))^2 + (a(2)-w(Jmin,2))^2);
      %if distJ < 0.01*mindst
      if distJ < 1
          w(Jmin,1)=a(1);
         w(Jmin,2)=a(2);
      end;
   end;
   gama=gama*0.998;
   gain=(1-gama)*gain;

   if w == wold
        break;
    end
   wold=w;
   niter = niter+1;
end
niter
solution=y;
```

```
% get the draw points from weights
datadraw=[w;w(1,:)];
ttt=cputime-tt
```