Mapa de Kohonen

Mateus Pontes Mota

6 de junho de 2018

Sumário

Capítulo 1

Enunciado

Implemente (anexando a listagem) um mapa de Kohonen (SOM) com 6x6 neurônios no MATLAB que aproxime essa distribuição de pontos. Documente os resultados do aprendizado do algoritmo plotando,em diferentes iterações, a evolução dos pesos da rede (a grade bidimensional).

Apresentar:

1. O código MATLAB implementado comentado.
2. Descreva sucintamente a lógica do programa entregue
3. Plote a grade bidimensional com a posição dos vetores de pesos e suas conexões após 100, 1000, 2000, 5000 iterações.

Capítulo 2

Descrição da Solução

Para solucionar este problema implementando em *MatLab* iniciamos carregando a base de dados e definindo os parâmetros. Notemos que os pesos estão alocados em um tensor, duas matrizes 6x6 concatenadas.

Listing 2.1: Carregamento dos dados e declaração dos Parâmetros

```
%% Clears e Imports
clear all;
clc;
load('dados.mat');
%% Declaração das Variavéis
dNeurons=6; %n de neuronios da rede
dAlpha0=0.6; %alfa inicial
dSigma0=20; %sigma inicial
dMaxIter=5000; %max de iterações
dT=0; %iteração
mtInput=pontos'; %matriz de entrada, transpor para que cada linha seja uma
    amostra
[dL, dC] = size(mtInput); %n de linhas e de colunas
tsW=rand(dNeurons,dNeurons,dC); %tensor dos pesos, duas matriz 6x6
    concatenadas
dThal=dMaxIter/log10(dSigma0); %Thal  utilizado no calculo de sigma
```

Em seguida fazemos o plot inicial do nosso mapa.

Listing 2.2: Plot Inicial

```
%% Plot Inicial
figure(1);
plot(mtInput(:,1),mtInput(:,2),'.b')
hold on;
plot(tsW(:,:,1),tsW(:,:,2),'or')
plot(tsW(:,:,1),tsW(:,:,2),'k','linewidth',2)
plot(tsW(:,:,1)',tsW(:,:,2)','k','linewidth',2)
hold off;
title('t=0');
drawnow;
```

Realizamos o treinamento da rede de atualizando a taxa de aprendizagem e o tamanho da vizinhança, em seguida atualizamos os pesos do neurônio vencedor e dos vizinhos, analisando se os vizinhos ultrapassam os limites da matriz.

Listing 2.3: Treinamento

```
%% Treinamento
while(dT<=dMaxIter)
    dAlpha=dAlpha0*exp(-dT/dMaxIter); %diminuir a taxa de aprendizagem
    dSigma=dSigma0*exp(-dT/dThal); %diminuir o tamanho da vizinhança
    for ii= randperm(dL) % seleção aleatoria das amostras
        mtDist=(mtInput(ii,1)-tsW(:,:,1)).^2+(mtInput(ii,2)-tsW(:,:,2)).^2;
           %calculo da distancia euclidiana
        [vtWinV,vtWinId]=min(mtDist); %busca do vencedor por coluna com o
           indice da linha
        [~,dMinCol]=min(vtWinV); %coluna do menor valor
        dMinLin=vtWinId(dMinCol); %linha do menor valor
        %atualização do neuronio vencedor
        tsW(dMinLin,dMinCol,:) = [tsW(dMinLin,dMinCol,1) tsW(dMinLin,dMinCol
           ,2)] + dAlpha*(mtInput(ii,:)-[tsW(dMinLin,dMinCol,1) tsW(dMinLin,
           dMinCol,2)]);
        %atualização dos vizinhos
        for jj=1:1:dSigma
            dLin=dMinLin-jj;
            dCol=dMinCol;
            if(dLin>=1) %limita linha minima
                tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                   dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                   ]);
            end
            dLin=dMinLin+jj;
            dCol=dMinCol;
            if(dLin<=6) %limita linha maxima
                tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                   dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                   ]);
            end
            dLin=dMinLin;
            dCol=dMinCol-jj;
            if(dCol>=1) %limita coluna minima
                tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                   dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                   ]);
            end
            dLin=dMinLin;
            dCol=dMinCol+jj;
            if(dCol<=6) %limita coluna maxima
                tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                   dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                   ]);
            end
        end

    end
    dT=dT+1;
```

E no final fazemos os plots, salvando em figuras.

Listing 2.4: Plots Finais

```
    if(  (dT==1) || (dT==100) || (dT==1000) || (dT==2000) || (dT==5000) )
        figure(1);
        plot(mtInput(:,1),mtInput(:,2),'.b')
        hold on;
```

```
        plot(tsW(:,:,1),tsW(:,:,2),'or')
        plot(tsW(:,:,1),tsW(:,:,2),'k','linewidth',2)
        plot(tsW(:,:,1)',tsW(:,:,2)','k','linewidth',2)
        hold off;
        title(['t=' num2str(dT)]);
        drawnow
        name=sprintf('T%deAlfa%d',dT,dAlpha0*10);
        print(name,'-dpng');
    end
end
```

Capítulo 3

Resultados

3.1 Taxa de aprendizagem = 0.3

Figura 3.1: 100 iterações

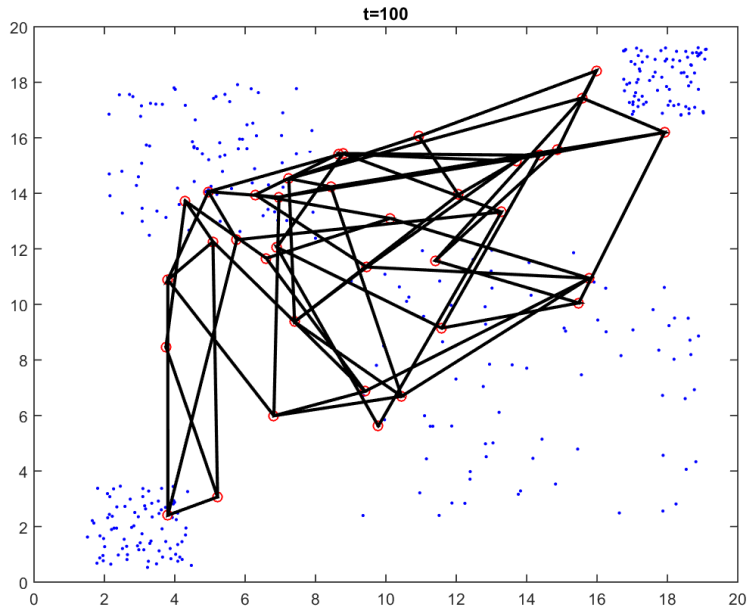


Figura 3.2: 1000 iterações

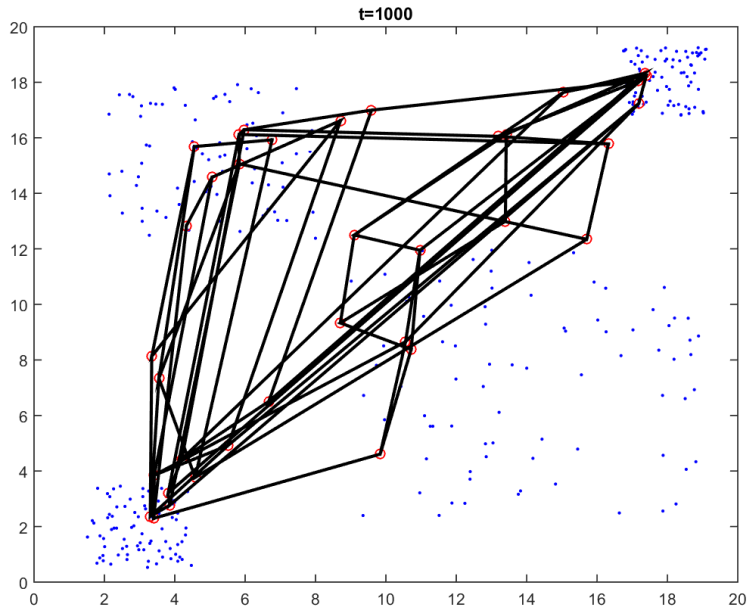


Figura 3.3: 2000 iterações

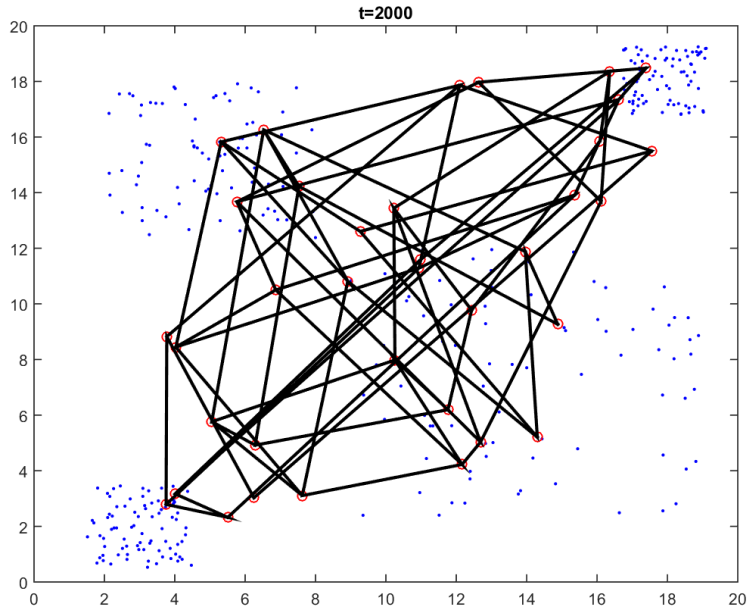
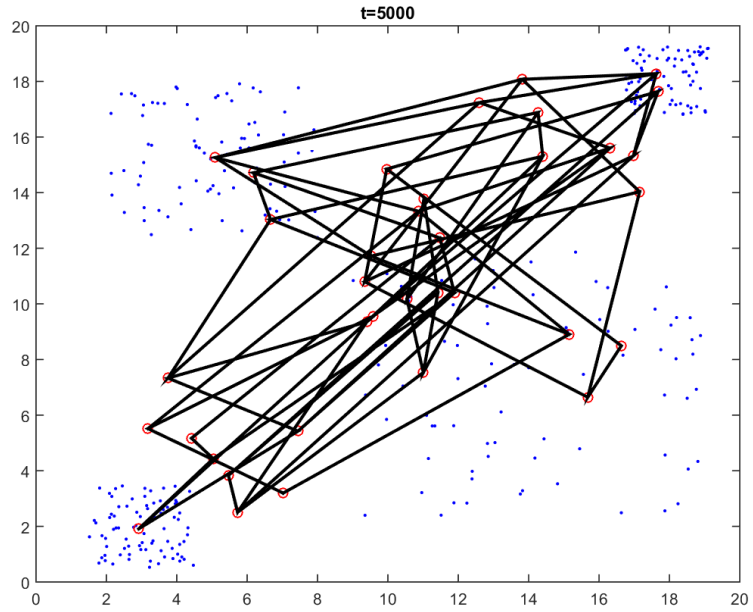


Figura 3.4: 5000 iterações



3.2 Taxa de aprendizagem = 0.6

Figura 3.5: 100 iterações

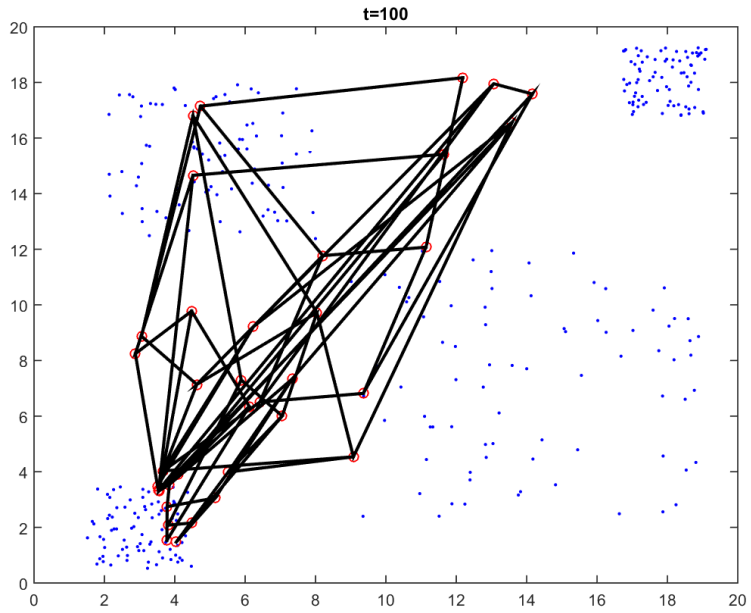


Figura 3.6: 1000 iterações

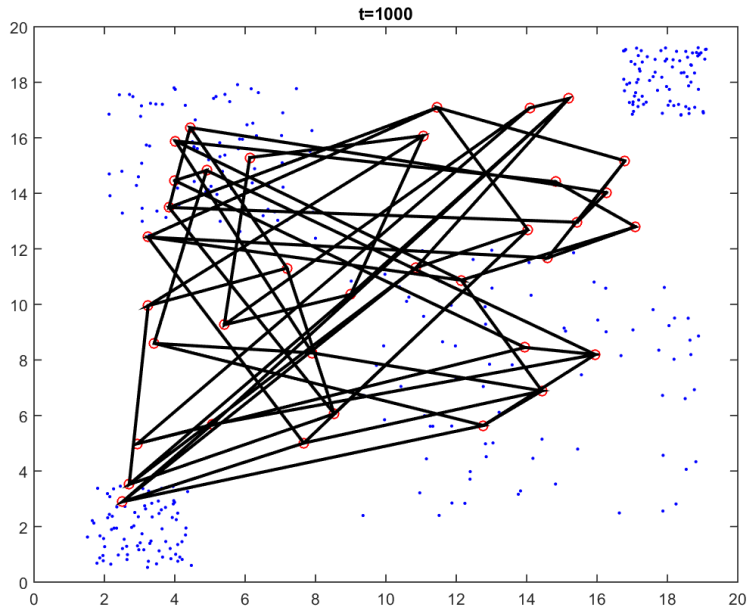


Figura 3.7: 2000 iterações

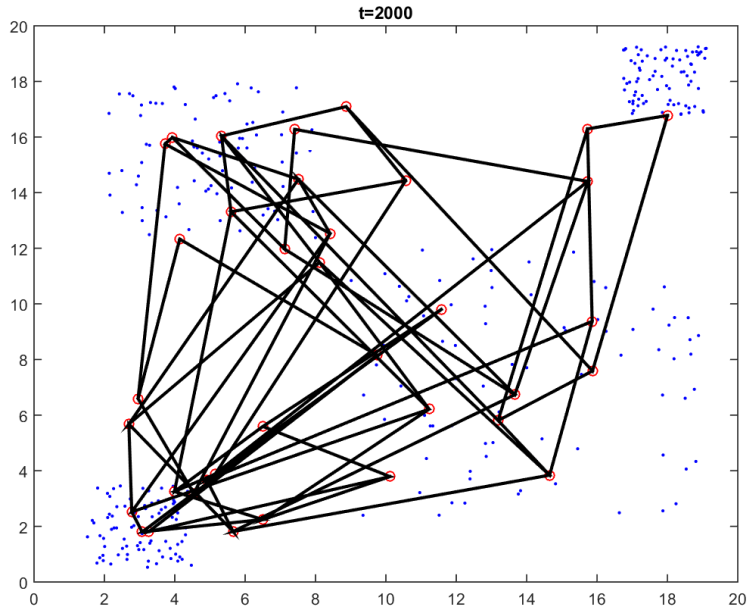
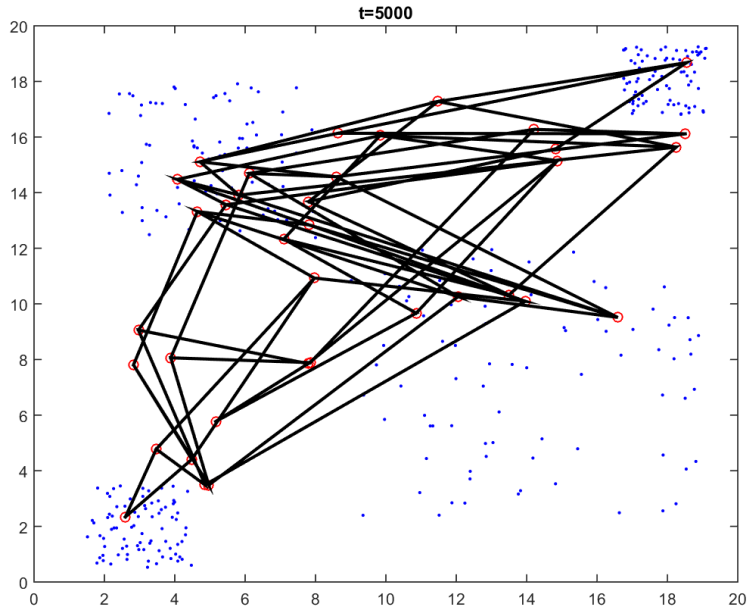


Figura 3.8: 5000 iterações



3.3 Mudando o decrescimento das taxas

Neste caso mudamos o decrescimento das taxas de aprendizagem e da atualização de vizinhança. Abaixo mostramos como o novo calculo é feito. Nesta forma obtemos um melhor traçado como podemos observar pelas figuras abaixo

Listing 3.1: Mudança dos Decrescimentos

```
dAlpha=dAlpha0*(1-dT/dMaxIter); %diminuir a taxa de aprendizagem
dSigma=round(dSigma0*(1-dT/dMaxIter)); %diminuir o tamanho da vizinhança
```

Figura 3.9: 100 iterações

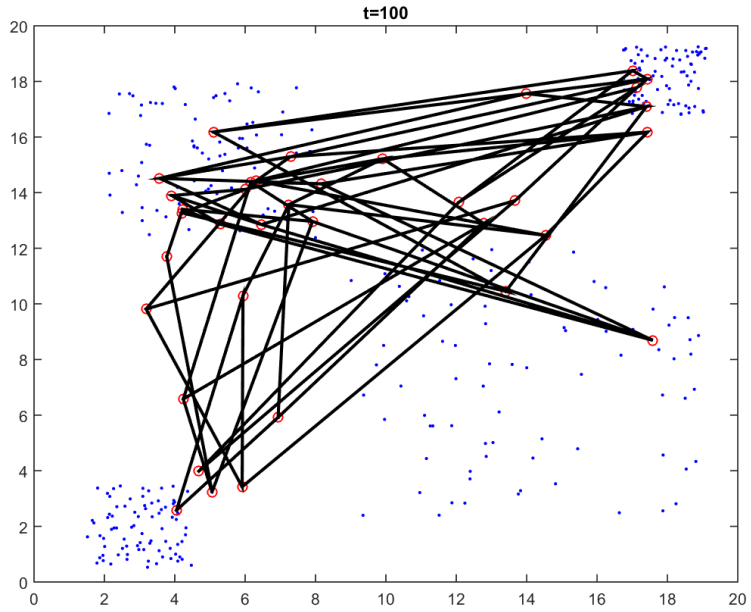


Figura 3.10: 1000 iterações

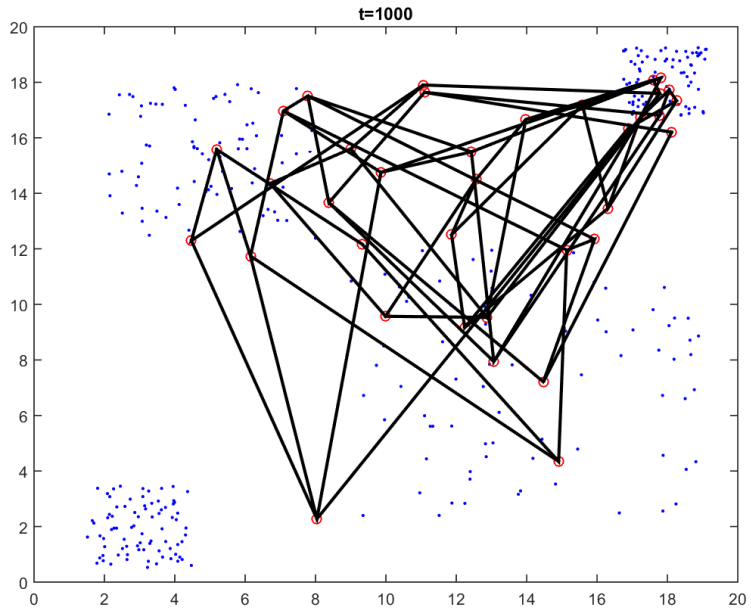


Figura 3.11: 2000 iterações

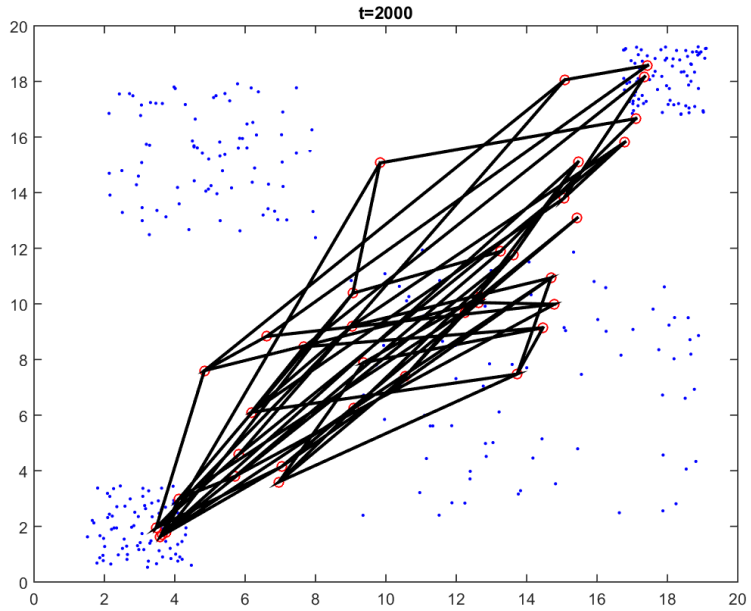
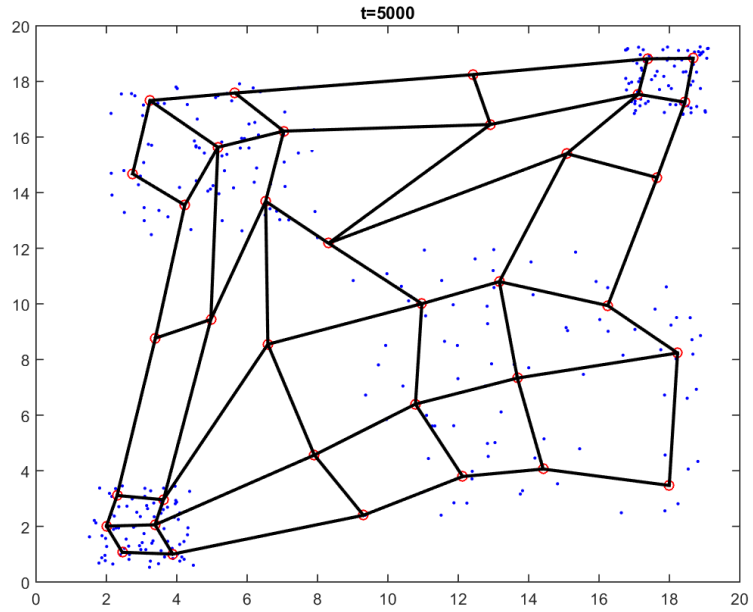


Figura 3.12: 5000 iterações



Apêndice A

Código

Listing A.1: Código Completo

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mateus Pontes Mota
% Redes Neurais Artificiais
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Clears e Imports
clear all;
clc;
load('dados.mat');
%% Declaração das Variavéis
dNeurons=6; %n de neuronios da rede
dAlpha0=0.3; %alfa inicial
dSigma0=20; %sigma inicial
dMaxIter=5000; %max de iterações
dT=0; %iteração
mtInput=pontos'; %matriz de entrada, transpor para que cada linha seja uma amostra
[dL, dC] = size(mtInput); %n de linhas e de colunas
tsW=rand(dNeurons,dNeurons,dC); %tensor dos pesos, duas matriz 6x6 concatenadas
dThal=dMaxIter/log10(dSigma0); %Thal  utilizado no calculo de sigma
%% Plot Inicial
figure(1);
plot(mtInput(:,1),mtInput(:,2),'.b')
hold on;
plot(tsW(:,:,1),tsW(:,:,2),'or')
plot(tsW(:,:,1),tsW(:,:,2),'k','linewidth',2)
plot(tsW(:,:,1)',tsW(:,:,2)','k','linewidth',2)
hold off;
title('t=0');
drawnow;
%% Treinamento
while(dT<=dMaxIter)
    dAlpha=dAlpha0*exp(-dT/dMaxIter); %diminuir a taxa de aprendizagem
    dSigma=dSigma0*exp(-dT/dThal); %diminuir o tamanho da vizinhança
    for ii= randperm(dL) % seleção aleatoria das amostras
        mtDist=(mtInput(ii,1)-tsW(:,:,1)).^2+(mtInput(ii,2)-tsW(:,:,2)).^2; %calculo da distancia euclidiana
        [vtWinV,vtWinId]=min(mtDist); %busca do vencedor por coluna com o indice da linha
        [~,dMinCol]=min(vtWinV); %coluna do menor valor
```

```
37          dMinLin=vtWinId(dMinCol); %linha do menor valor
38          %atualização do neuronio vencedor
39          tsW(dMinLin,dMinCol,:) = [tsW(dMinLin,dMinCol,1) tsW(dMinLin,dMinCol
                ,2)] + dAlpha*(mtInput(ii,:)-[tsW(dMinLin,dMinCol,1) tsW(dMinLin,
                dMinCol,2)]);
40          %atualização dos vizinhos
41          for jj=1:1:dSigma
42              dLin=dMinLin-jj;
43              dCol=dMinCol;
44              if(dLin>=1) %limita linha minima
45                  tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                        dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                        ]);
46              end
47              dLin=dMinLin+jj;
48              dCol=dMinCol;
49              if(dLin<=6) %limita linha maxima
50                  tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                        dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                        ]);
51              end
52              dLin=dMinLin;
53              dCol=dMinCol-jj;
54              if(dCol>=1) %limita coluna minima
55                  tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                        dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                        ]);
56              end
57              dLin=dMinLin;
58              dCol=dMinCol+jj;
59              if(dCol<=6) %limita coluna maxima
60                  tsW(dLin,dCol,:) = [tsW(dLin,dCol,1) tsW(dLin,dCol,2)] +
                        dAlpha*(mtInput(ii,:)-[tsW(dLin,dCol,1) tsW(dLin,dCol,2)
                        ]);
61              end
62          end
63
64      end
65      dT=dT+1;
66      if(  (dT==1) || (dT==100) || (dT==1000) || (dT==2000) || (dT==5000) )
67          figure(1);
68          plot(mtInput(:,1),mtInput(:,2),'.b')
69          hold on;
70          plot(tsW(:,:,1),tsW(:,:,2),'or')
71          plot(tsW(:,:,1),tsW(:,:,2),'k','linewidth',2)
72          plot(tsW(:,:,1)',tsW(:,:,2)','k','linewidth',2)
73          hold off;
74          title(['t=' num2str(dT)]);
75          drawnow
76          name=sprintf('T%deAlfa%d',dT,dAlpha0*10);
77          print(name,'-dpng');
78      end
79  end
```