```
classdef ALG_COMPET < handle
    properties
        xi % matriz de amostras de treinamento
        xin % vetor de entrada do k-ésimo padrão
        %xnk % vetro normalizado de entrada do k-ésimo padrão
        w % matriz do pesos
        wn % vetro normalizado de pesos
        eta % taxa de aprendizagem
        lx % nº de linhas das matizes de pesos e de treinamento (nº de entrdadas)
        cx % nº de colunas da matriz de amostras de treinamneto (nº de amostras)
        cw % nº de colunas da matriz de pesos (nº de neurônios)
    end
    methods
        function obj = ALG_COMPET(n_ent,n_amost, n_class)
            % inicialização da matriz de pesos e de amostaras
            obj.xi = -1*ones(n_ent+1, n_amost);% Sem bias
            %obj.w = 2*rand(n_ent+1,n_class)-1;
%             nw = ceil(n_amost);
%             sxi = 0;
%              for j = 1:n_class
%                  for i = 1:n_ent
%                       for k = 1:nw
%                            %sxi = sxi + obj.xi(i,k);
%                       end
%                      %obj.w(i,j) = (sxi/nw)*j;
%                  end
%              end
            obj.lx = n_ent+1;% nº de entradas sem bias
            obj.cx = n_amost;% nº de amostras
            obj.cw = n_class;% nº de neurônios
        end
        function nor(obj,entradat)
            obj.xi(2:obj.lx,:) = entradat;
            obj.w = 6*rand(obj.lx,obj.cw)-3;% Inicialização aleatória da matriz de↙
pesos
            obj.w(1,:) = -1;
            % Normalização dos vetores de pesos e de treinamento
             for xc = 1:obj.cx
                 obj.xin(:,xc) = obj.xi(:,xc)/norm(obj.xi(:,xc));
             end
             for wc = 1:obj.cw
                 obj.wn(:,wc) = obj.w(:,wc)/norm(obj.w(:,wc));
             end
            plot(obj.wn(2,:),obj.wn(3,:),'or')
            hold on
            plot(obj.xin(2,1:51),obj.xin(3,1:51),'b')
            plot(obj.xin(2,52:102),obj.xin(3,52:102),'y')
            %sphere
            grid
        end
        function [mdist, W] = trei(obj,txap,nep)
            obj.eta = txap; %Taxa de aprendizagem
            a=0;
            dist = [];
            mdist = [];
```

```
            ind = 0;
            epoca = 1;
            while a==0
                a=0;
                for k = 1:obj.cx
                    for j = 1:obj.cw
                        somatxw = 0;
                        for i = 1:obj.lx
                            somatxw = somatxw + (obj.xin(i,k)-obj.wn(i,j))^2;
                        end
                        dist(j) = sqrt(somatxw);
                    end
                    [mdist, ind] = min(dist);
                    obj.wn(:,ind) = obj.wn(:,ind)+ obj.eta*(obj.xin(:,k)-obj.wn(:,↙
ind));
                end
                if epoca == nep
                    a=1;
                end
                epoca=epoca+1;
            end
            plot(obj.wn(2,:),obj.wn(3,:),'or')
            hold on
            plot(obj.xin(2,1:51),obj.xin(3,1:51),'b')
            plot(obj.xin(2,52:102),obj.xin(3,52:102),'y')
            grid
            W = obj.wn;
        end
%         function ind = valid(obj, entradav, pesost)
%             obj.xi = entradav;
%             % Normalização dos vetores de pesos e de treinamento
%             for xc = 1:obj.cx
%                 obj.xin(:,xc) = obj.xi(:,xc)/norm(obj.xi(:,xc));
%             end
%             obj.wn = pesost;
%             plot(obj.wn(1,:),obj.wn(2,:),'*r')
%             hold on
%             plot(obj.xin(1,1:51),obj.xin(2,1:51),'*b')
%             plot(obj.xin(1,52:102),obj.xin(2,52:102),'*y')
%             grid
%             figure
%             for k = 1:obj.cx
%                     for j = 1:obj.cw
%                         somatxw = 0;
%                         for i = 1:obj.lx
%                             somatxw = somatxw + (obj.xin(i,k)-obj.wn(i,j))^2;
%                         end
%                         dist(j) = sqrt(somatxw);
%                     end
%                     [mdist ind(k)] = min(dist);
%                     if ind == 1
%                          plot(obj.xin(1,k),obj.xin(2,k),'*k')
%                     else
%                         plot(obj.xin(1,k),obj.xin(2,k),'*r')
%                     end
```

```
%                       hold on
%                   end
%                   plot3(obj.wn(1,:),obj.wn(2,:),'*y')
%                   grid
%               end
    end

end
```