```
classdef RBF < handle
    properties

        ni % Número de entradas
        nh % Numero de neuronios ocultos
        no % Numero de saidas
        n_classes % Número de classes

        % Vetores de entrada e ativação
        xi % Vetor de entradas
        yh % Saída na camada oculta
        yo % Saída da rede
        centro % Vetor com índices de cada classe retornado pelo kmeans

        % Vetores de erro e gradientes locais
        delta
        erro

        % Matrizes de centros e pesos sinápticos
        wo
        sigma2
        wc
    end

    methods (Access = private)
        function iniciar_centros(obj,met_de_ini,padroes)
            if met_de_ini == 's'
                figure;scatter(padroes(:,1),padroes(:,2))
                centros = ginput(obj.nh);
                [obj.centro,centros] =...
                kmeans(padroes(:,1:obj.ni),obj.nh,'start',obj.wc');
                close;
            else
                if isempty(obj.wc)
                    [obj.centro,centros] =...
                    kmeans(padroes(:,1:obj.ni),obj.nh);
                else
                    [obj.centro,centros] =...
                    kmeans(padroes(:,1:obj.ni),obj.nh,'start',obj.wc');
                end
            end
            obj.wc = centros';
        end
    end

    methods
        function self = RBF(ni, nh, no, n_classes)
            % Inicialização dos parâmetros da rede
            self.ni = ni;
            self.nh = nh;
            self.no = no;
            self.n_classes = n_classes;

            % Inicialização dos principais sinais
            self.xi = zeros(self.ni,1);
            self.yh = ones(self.nh+1,1);
```

```
    self.yo = zeros(self.no,1);
    self.delta = zeros(self.no,1);

    % Vetor que conterá o erro da última camada
    self.erro = zeros(self.no,1);

    % Inicialização de pesos sinápticos e mudança anterior
    % nos pesos para o momento
    self.wc = [];
    self.wo = 2*rand(nh+1,no)-1;
end

function saida = atualizar(obj,entradas)
    % Verificando a quantidade de entradas
    try
        obj.xi = entradas(:);
    catch e
        throw(e);
    end

    % Passando pelas exponenciais
    for c = 1:1:obj.nh
        obj.yh(c,1) = exp(-(obj.xi - obj.wc(:,c))'*...
        (obj.xi - obj.wc(:,c)))/(2*obj.sigma2(c));
    end

    % Passando pelo combinador linear
    obj.yo = (obj.yh'*obj.wo)';

    saida = obj.yo;
end

function erro_quad = ajustes_dos_pesos(obj, desejado, eta)
    % Verificando a quantidade de saidas
    try
        obj.erro = desejado - obj.yo;
    catch e
        throw(e);
    end

    % Calculando o delta na camada de saída
    obj.delta = obj.erro;

    % Fazendo correções nas sinapses da camada de saída
    obj.wo = obj.wo + eta*(obj.delta*obj.yh')';

    E = 0.5*(obj.erro'*obj.erro);

    erro_quad = E;
end

function treinar(obj, padroes, epocas, eta, met_de_ini)
    % padroes   --> Padrões a serem utilizados no treino
    % epocas    --> Número de épocas
    % eta       --> Taxa de aprendizado
```

```
            % met_de_ini--> Inicialização dos centros:
            %               'a' (aleatórios), 's' (supervisionado) e
            %               'as' (autosupervisionado)
            [n_pad, n_io] = size(padroes);
            J = zeros(epocas,1);

            %if (met_de_ini ~= 'a')
            obj.iniciar_centros(met_de_ini,padroes)
            %end

            'Valor de wo antes do treino'
            obj.wo
            % Calculo da variância
            for c=1:1:obj.nh
                obj.sigma2(c) =...
                mean(sum((padroes(obj.centro==c,1:obj.ni)'-
obj.wc(c)).^2));
            end

            for i = 1:1:epocas
                padroes = padroes(randperm(n_pad),:); % Misturando
entradas
                erro_quad = 0;
                for p = 1:1:n_pad
                    entrada = padroes(p,1:(obj.ni));
                    desejado = padroes(p,obj.ni+1:n_io);
                    atualizar(obj,entrada);
                    erro_quad =...
                    erro_quad + obj.ajustes_dos_pesos(desejado, eta);
                end
                J(i) = erro_quad;
            end
            figure;
            plot(1:1:epocas,J);

            'Valor de wo após o treino'
            obj.wo
        end

        function saida_funcao = testar(obj,padroes)
            n_pad = size(padroes,1);
            saida = zeros(n_pad,1);
            for p = 1:1:n_pad
                entrada = padroes(p,:);
                % Armazena apenas as saídas em saida
                saida(p,1) = obj.atualizar(entrada);
            end
            saida_funcao = saida;
        end

    end

end
```