Inteligência Artificial

Redes Neurais Artificiais
Radial Basis Function Network

José Luis Seixas Junior

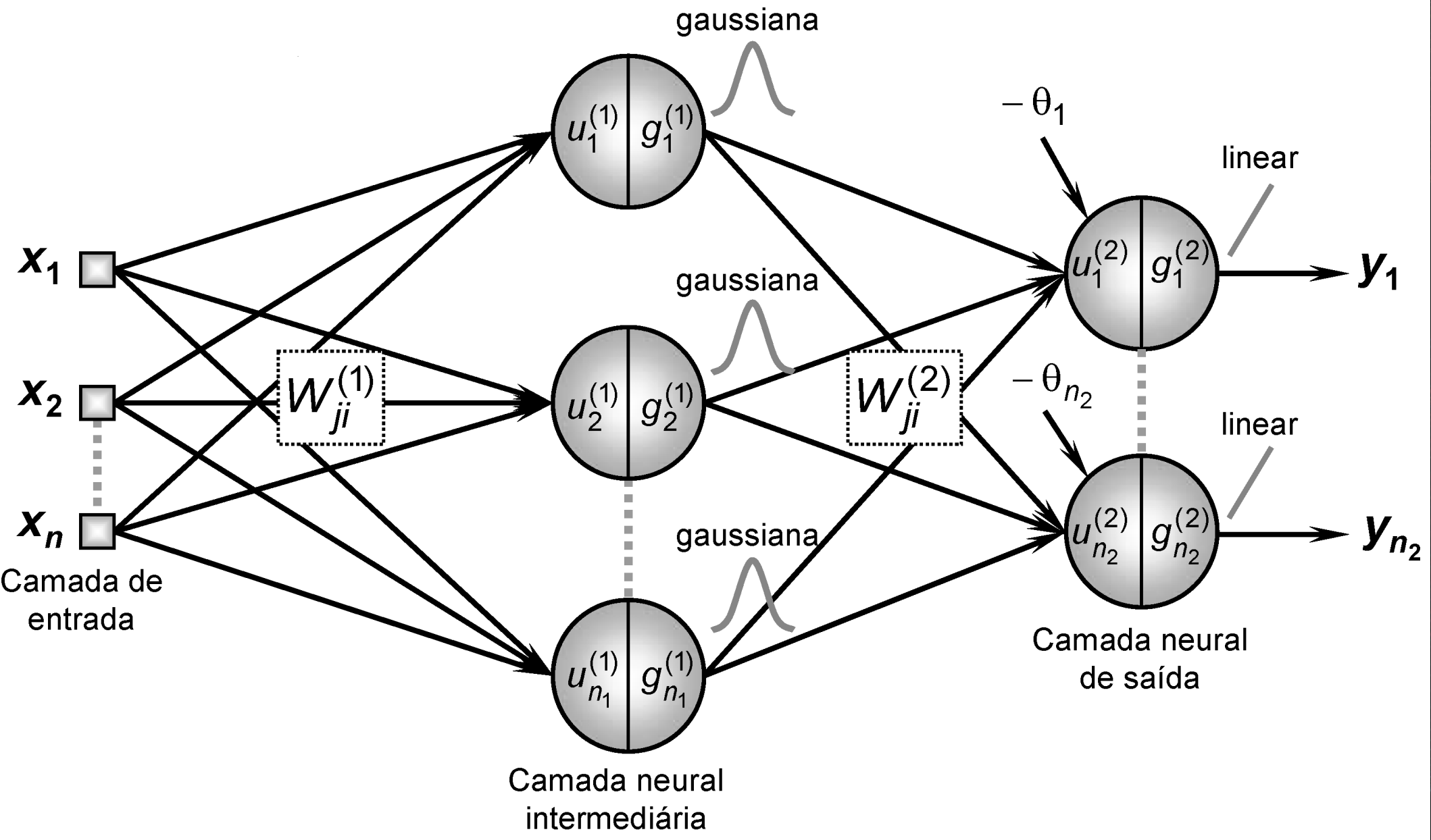
Índice

- Introdução;
- Processo de treinamento;
  - Camada Intermediária;
  - Camada de Saída;
- Aplicabilidade;
- Atividade;

Introdução

- Rede com neurônio feito com a Função de Base Radial;
- Diferentemente da MLP, possui apenas uma camada intermediária;
- Função de ativação, na maioria dos casos, do tipo Gaussiana;

Introdução

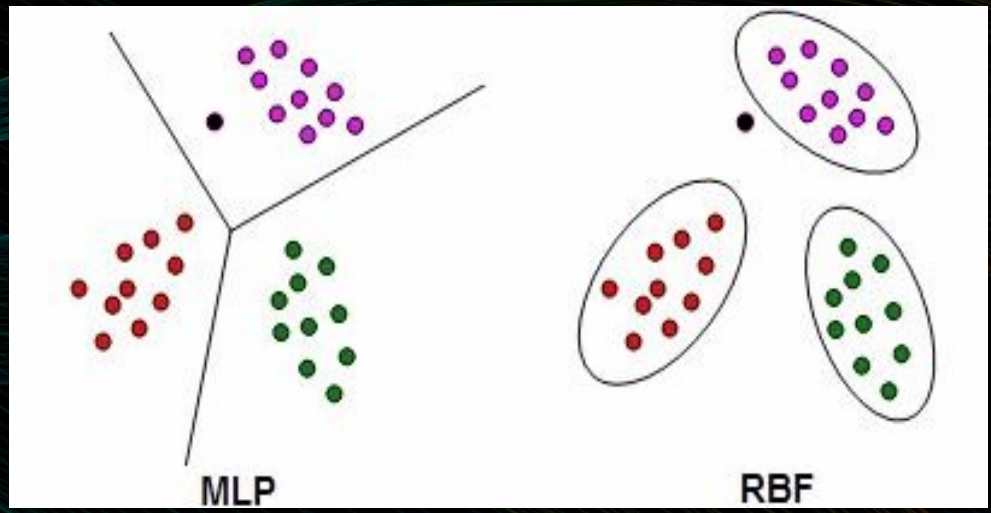

gaussiana
gaussiana
gaussiana
$u_1^{(1)}$ $g_1^{(1)}$
$u_2^{(1)}$ $g_2^{(1)}$
$u_{n_1}^{(1)}$ $g_{n_1}^{(1)}$
$W_{ji}^{(1)}$
$W_{ji}^{(2)}$
$-\theta_1$
$-\theta_{n_2}$
linear
linear
$u_1^{(2)}$ $g_1^{(2)}$
$u_{n_2}^{(2)}$ $g_{n_2}^{(2)}$
x_1
x_2
x_n
y_1
y_{n_2}
Camada de entrada
Camada neural intermediária
Camada neural de saída

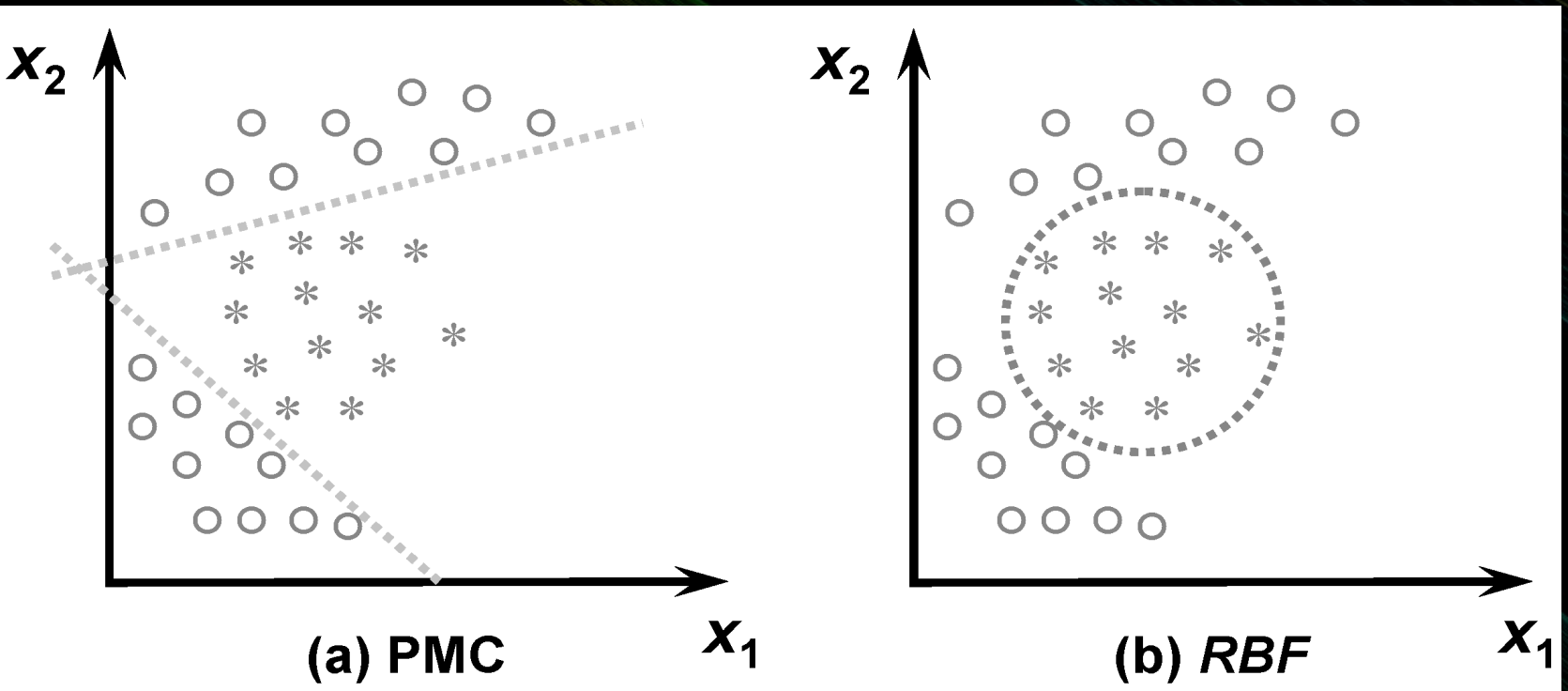
Introdução

- Estratégia de treinamento diversificada:
  - Camada intermediária é tratada como não supervisionada;
  - Camada de Saída é supervisionada;
- Segue a arquitetura *feedforward* de múltiplas camadas;

Introdução

- MLP x RBF:



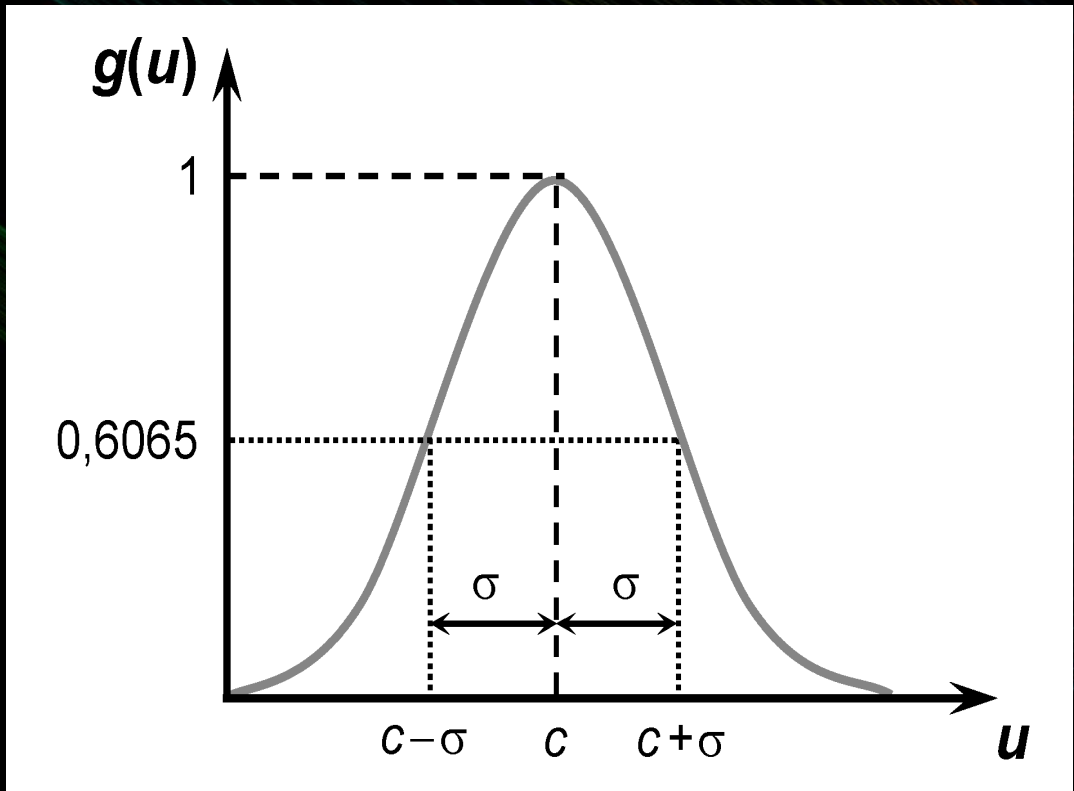


Treinamento

- Intermediária:
  - Estratégia auto-organizada, não apresentado supervisão;
  - Função de ativação Gaussiana:

$$g(u) = e^{\frac{(u-c)^2}{2\sigma^2}}$$

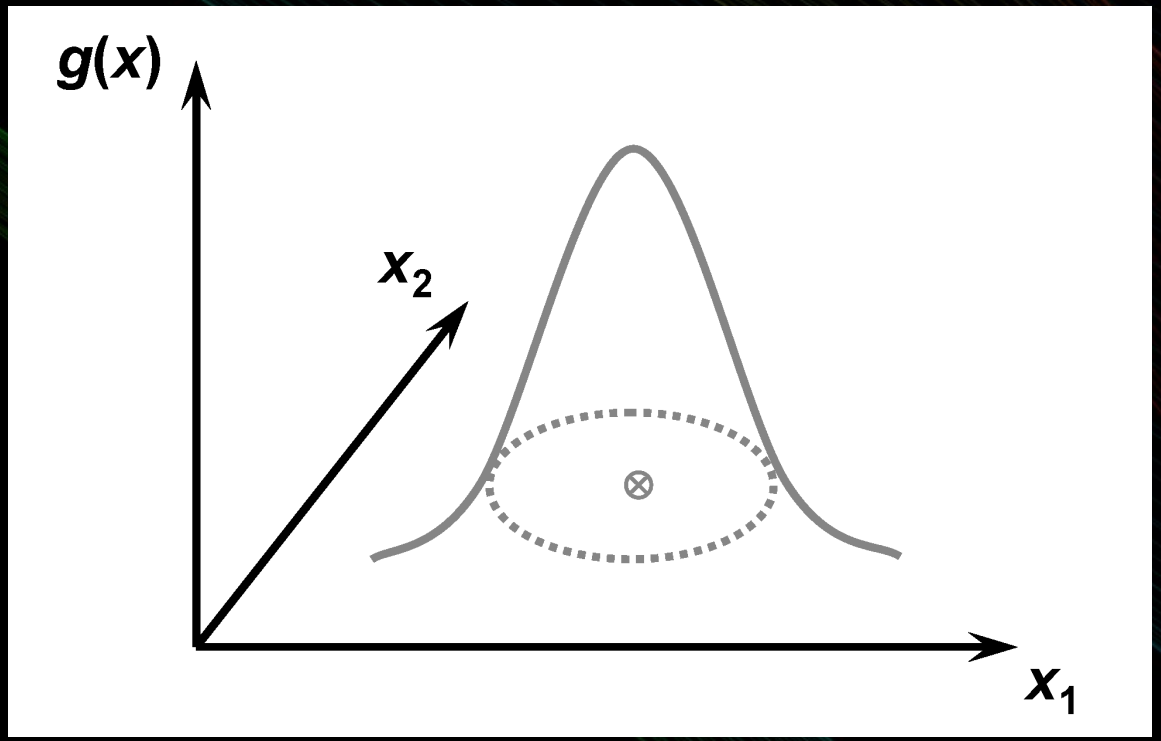
c: centro da função gaussiana
σ^2: variância
u: potencial de ativação



Treinamento

- Intermediária:
  - O centro **c** está associado ao peso, $\mathbf{u}_j$ será o próprio valor de entrada **x** indexado de *j*, assim temos:

$$g\left(u_j^{(1)}\right) = e^{\frac{\sum_{i=1}^{n} x_i - w_{ji}^{(1)}}{2\sigma^2}}$$

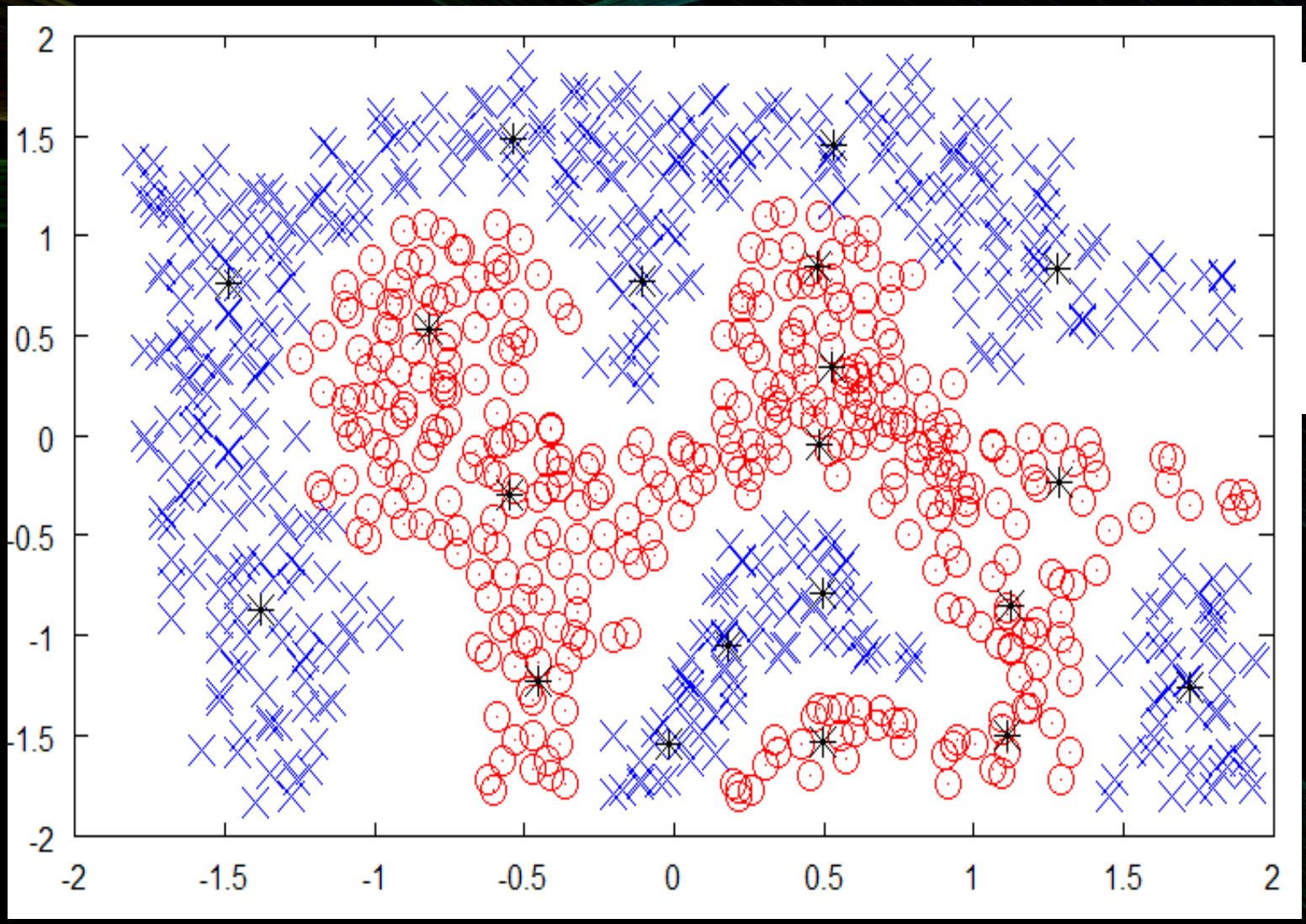


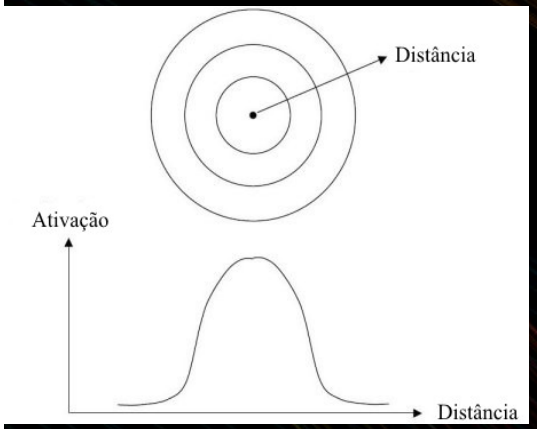
Treinamento

- Intermediária:
  - O principal objetivo dos neurônios da camada intermediária é posicionar os centro de suas gaussianas de forma mais apropriada possível, por exemplo, utilizando métodos como o **k-means**, posicionando o centro **k** em regiões onde os padrões de entrada tenderão a se agrupar;

Treinamento

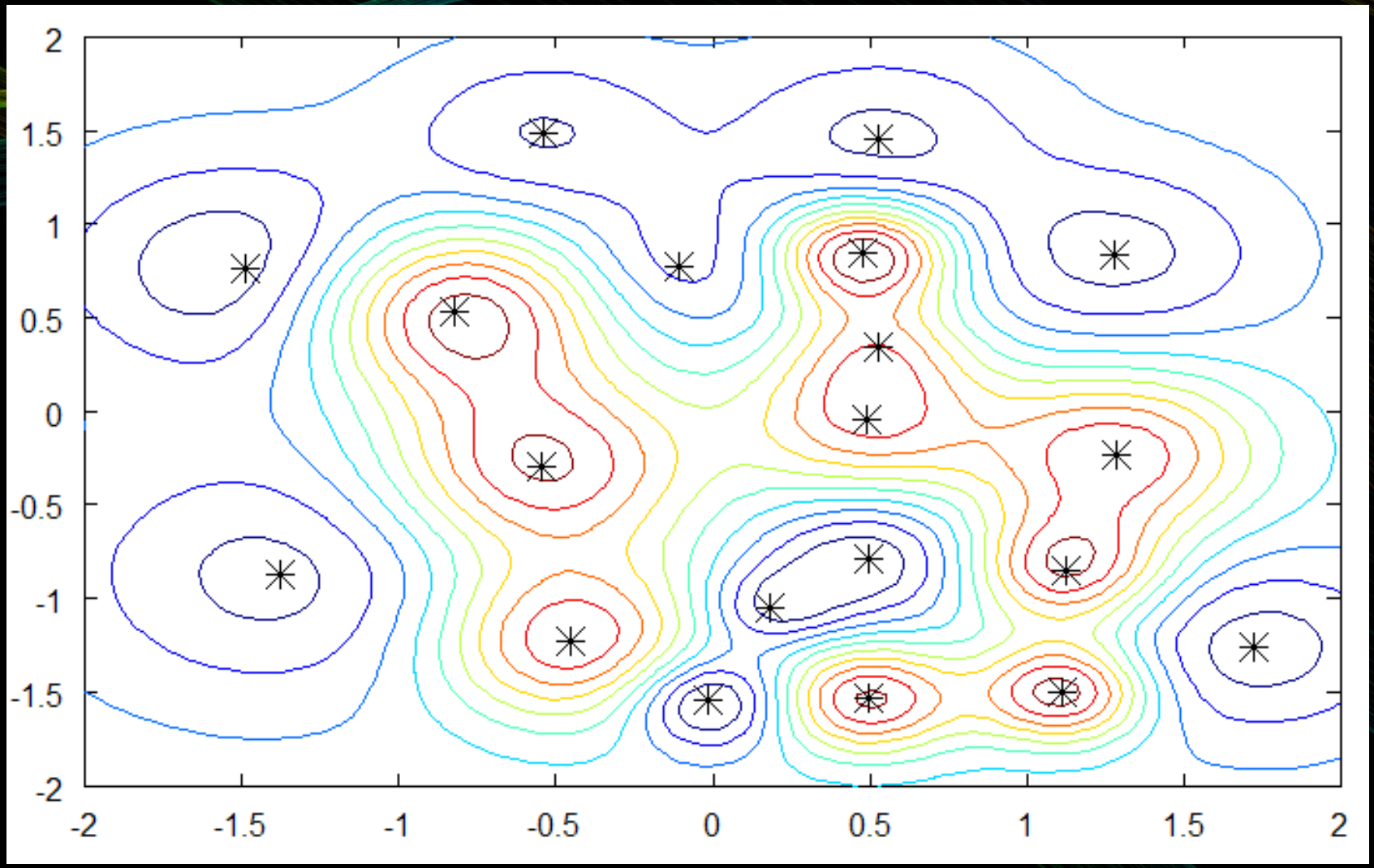
- Intermediária:





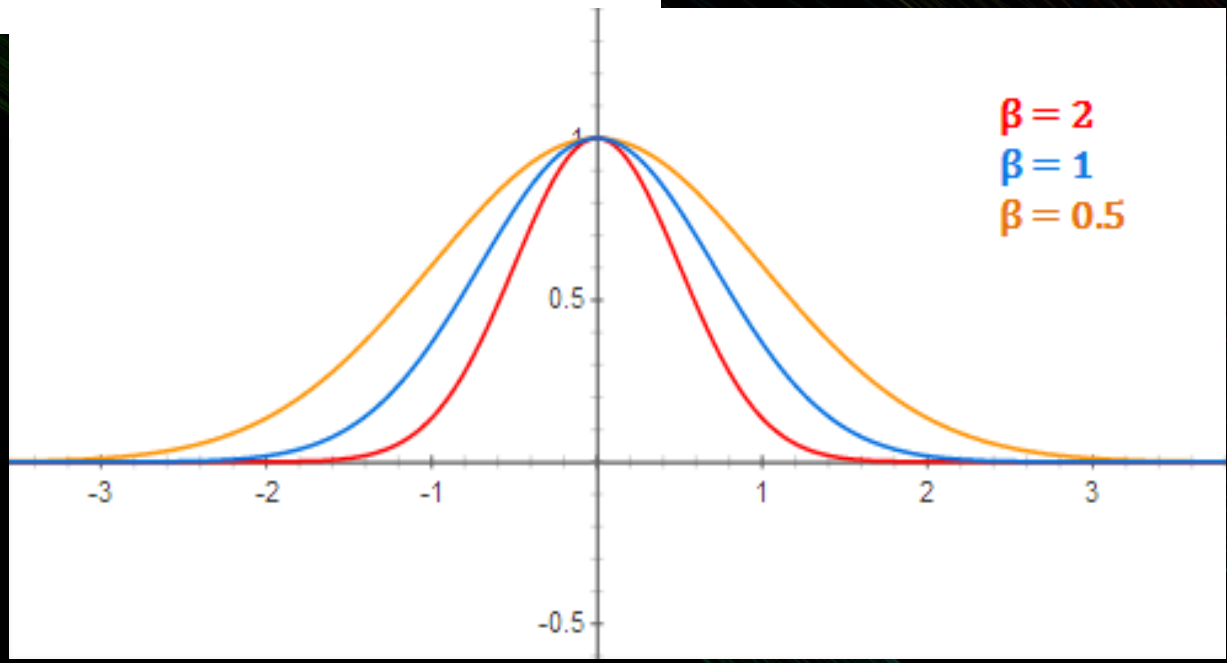
Treinamento

- Intermediária:



Trienamento

$$g(u) = e^{-\beta(u-c)^2}$$



Início {Algoritmo *RBF* – Primeiro Estágio de Treinamento}

<1> Obter o conjunto de amostras de treinamento $\{\boldsymbol{x}^{(k)}\}$;

<2> Iniciar o vetor de pesos de cada neurônio da camada intermediária com os valores das n_1 primeiras amostras de treinamento;

<3> Repetir as instruções:

<3.1> Para todas as amostras de treinamento $\{\boldsymbol{x}^{(k)}\}$, fazer:

<3.1.1> Calcular as distâncias euclidianas entre $\boldsymbol{x}^{(k)}$ e $W_{ji}^{(1)}$, considerando-se cada *j*-ésimo neurônio por vez;

<3.1.2> Selecionar o neurônio *j*, que contenha a menor distância, com o intuito de agrupar a referida amostra junto ao centro mais próximo;

<3.1.3> Atribuir a amostra $\boldsymbol{x}^{(k)}$ ao grupo $\Omega^{(j)}$;

<3.2> Para todos $W_{ji}^{(1)}$, onde $j = 1,\ldots,n_1$, fazer:

<3.2.1> Ajustar $W_{ji}^{(1)}$ de acordo com as amostras em $\Omega^{(j)}$:

$$W_{ji}^{(1)} = \frac{1}{m^{(j)}} \sum_{\boldsymbol{x}^{(k)} \in \Omega^{(j)}} \boldsymbol{x}^{(k)}$$

$\{m^{(j)}$ é o no. de amostras em $\Omega^{(j)}\}$

Até que: não haja mudanças nos grupos $\Omega^{(j)}$ entre as iterações;

<4> Para todos $W_{ji}^{(1)}$, onde $j = 1,\ldots,n_1$, fazer:

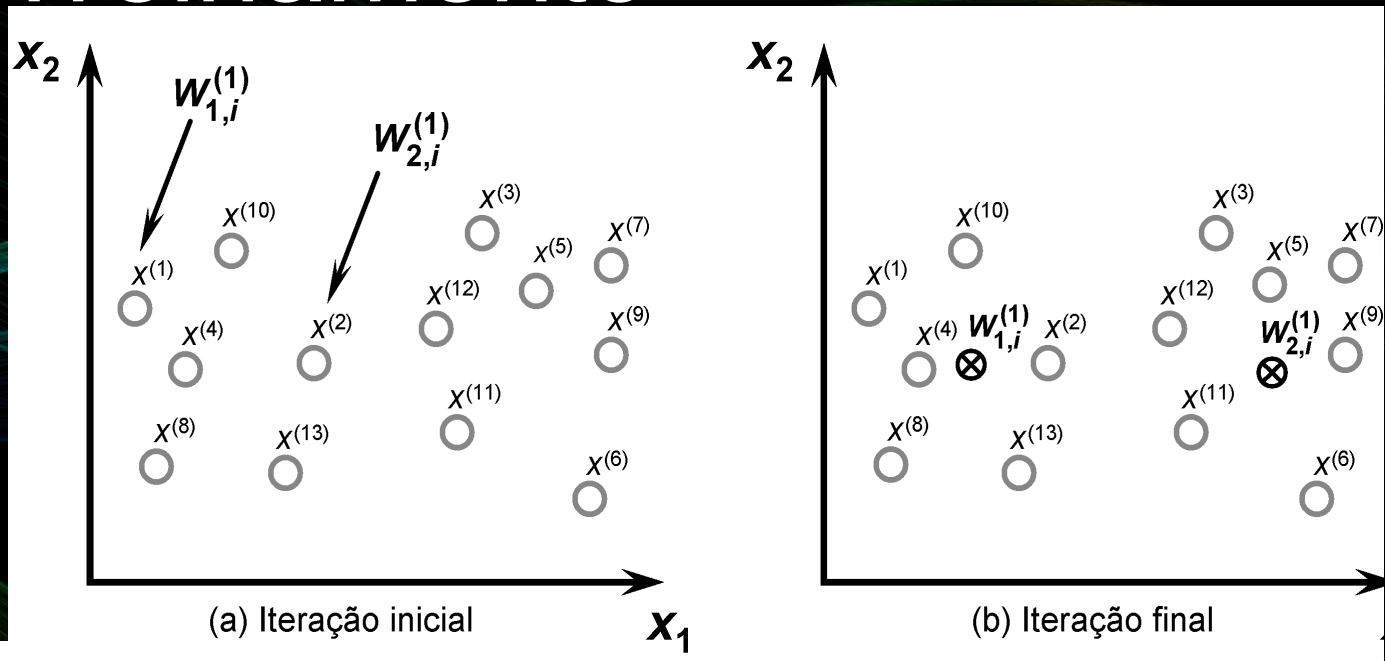
<4.1> Calcular a variância de cada uma das funções de ativação gaussianas pelo critério da distância quadrática média:

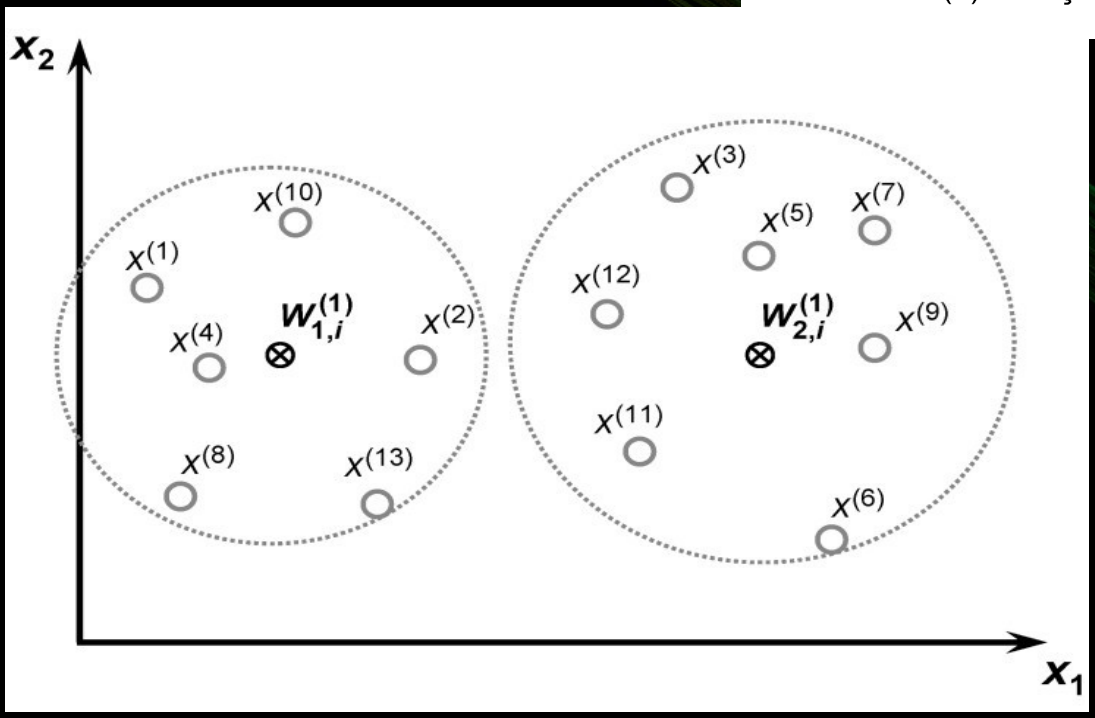
$$\sigma_j^2 = \frac{1}{m^{(j)}} \sum_{x^{(k)} \in \Omega^{(j)}} \sum_{i=1}^{n} (x_i^{(k)} - W_{ji}^{(1)})^2$$

Fim {Algoritmo *RBF* – Primeiro Estágio de Treinamento}

Treinamento

- Intermediária:





Saída

- Segue o mesmo procedimento da MLP com algoritmo da Regra Delta generalizada;

$$u_j^{(2)} = \sum_{(i=1)}^{n} w_{ji}^{(2)} \, g_i^{(1)}\left(u_i^{(1)}\right)$$

Início {Algoritmo *RBF* – Segundo Estágio de Treinamento}

<1> Obter o conjunto original de amostras de treinamento $\{ \boldsymbol{x}^{(k)} \}$;

<2> Obter o vetor de saída desejada $\{ \boldsymbol{d}^{(k)} \}$ para cada amostra;

<3> Iniciar $W_{ji}^{(2)}$ com valores aleatórios pequenos;

<4> Especificar taxa de aprendizagem $\{\eta\}$ e precisão requerida $\{\varepsilon\}$;

<5> Para todas as amostras $\{ \boldsymbol{x}^{(k)} \}$, fazer:

<5.1> Obter os valores de $g_j^{(1)}$ em relação à $\boldsymbol{x}^{(k)}$; {conforme (6.2)}

<5.2> Assumir $\boldsymbol{z}^{(k)} = [g_1^{(1)} \;\; g_2^{(1)} \ldots g_{n_1}^{(1)}]^T$; {*pseudo-amostras*}

<6> Iniciar o contador de número de épocas {*época* $\leftarrow$ 0};

<7> Repetir as instruções:

<7.1> $E_M^{anterior} \leftarrow E_M$; {conforme (5.8)}

<7.2> Para todos os pares de treinamento $\{ \boldsymbol{z}^{(k)}, \boldsymbol{d}^{(k)} \}$, fazer:

Ajustar $W_{ji}^{(2)}$ e θ_j *aplicando os mesmos passos usados na camada de saída do PMC* (Subseção 5.3.1)

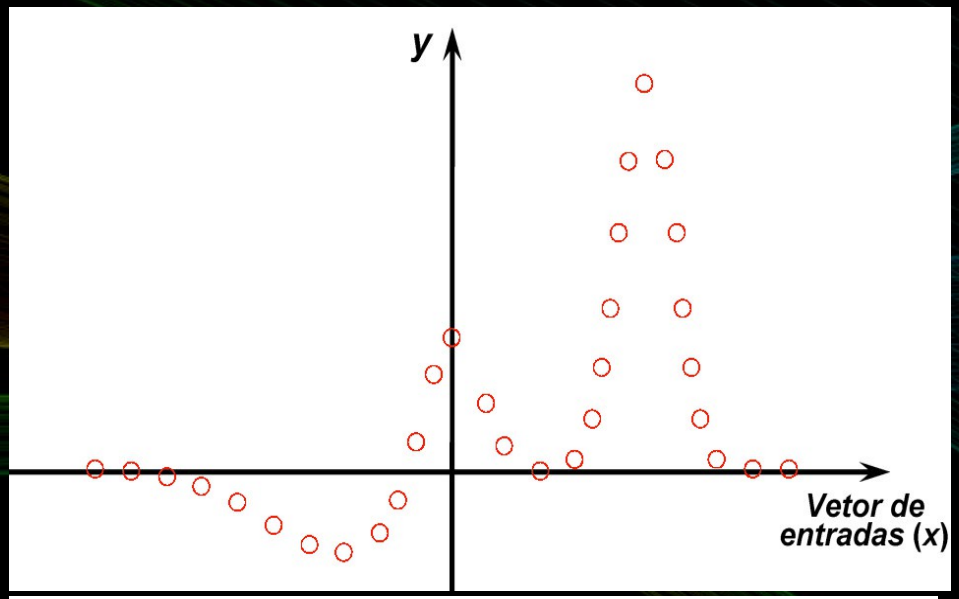
<7.3> $E_M^{atual} \leftarrow E_M$; {conforme (5.8)}

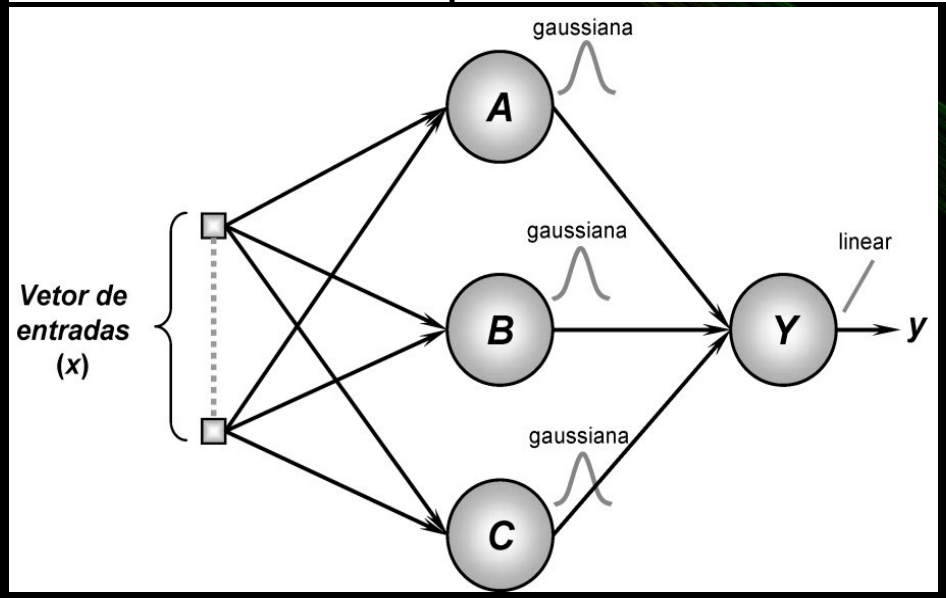
<7.4> *época* $\leftarrow$ *época* $+ 1$;

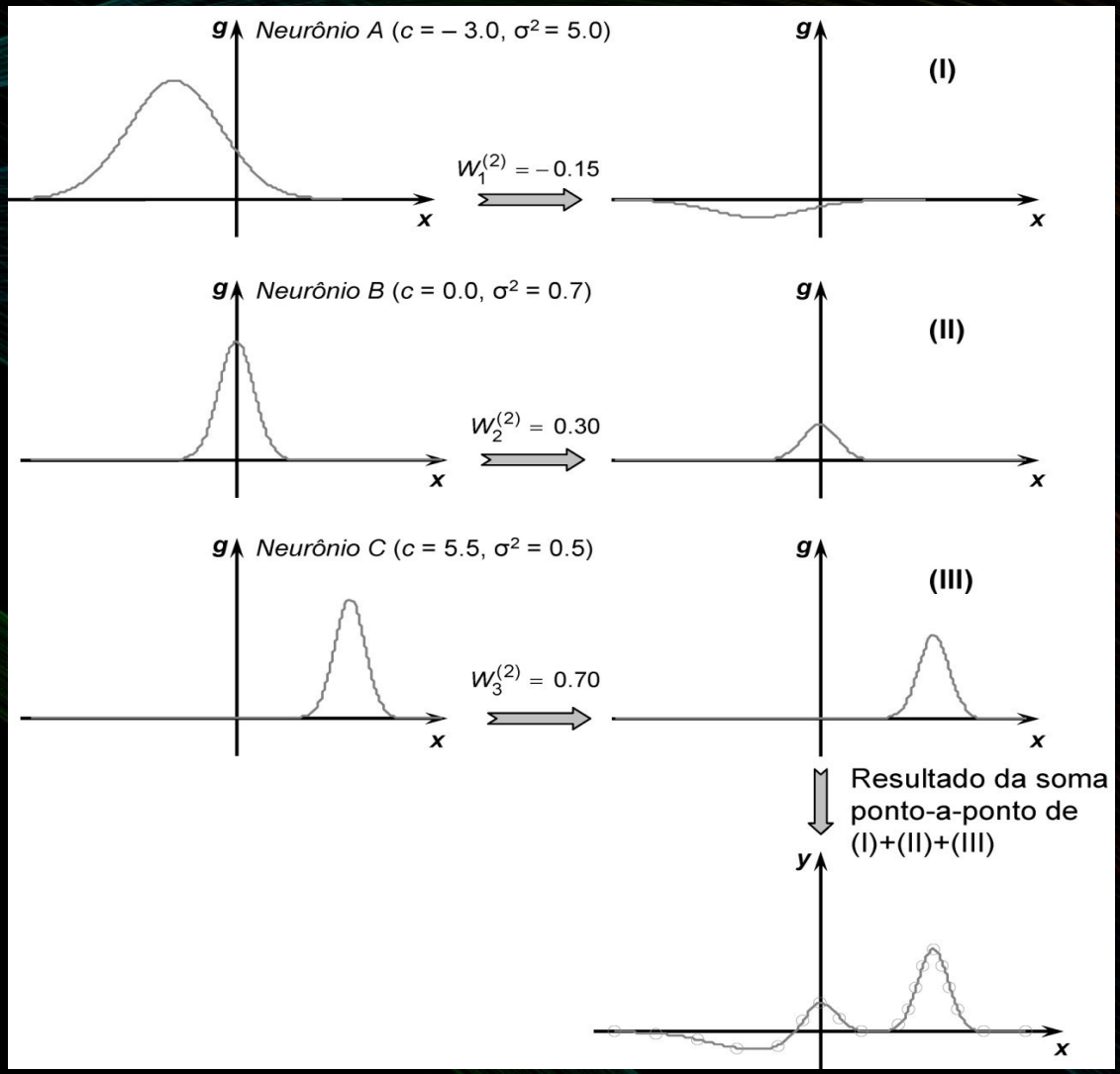
Até que: $| E_M^{atual} - E_M^{anterior} | \;\leq\; \varepsilon$

Fim {Algoritmo *RBF* – Segundo Estágio de Treinamento}

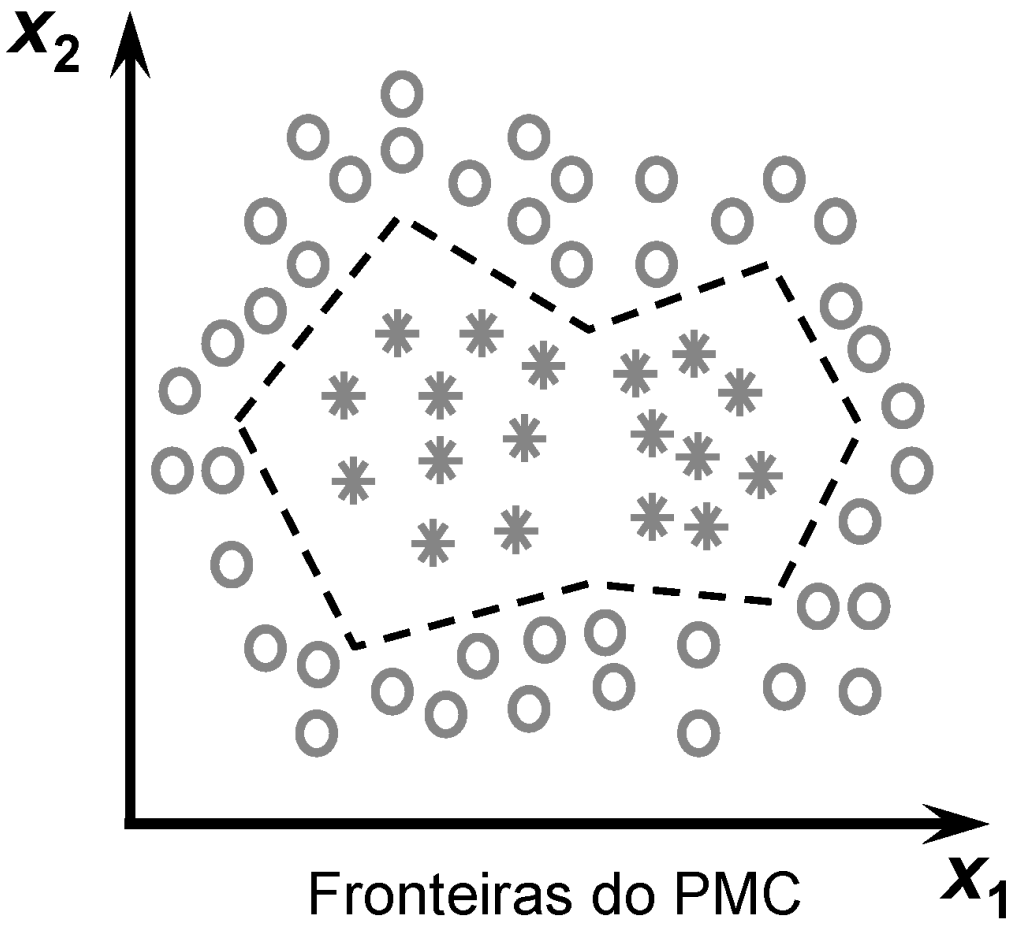
Aplicabilidade



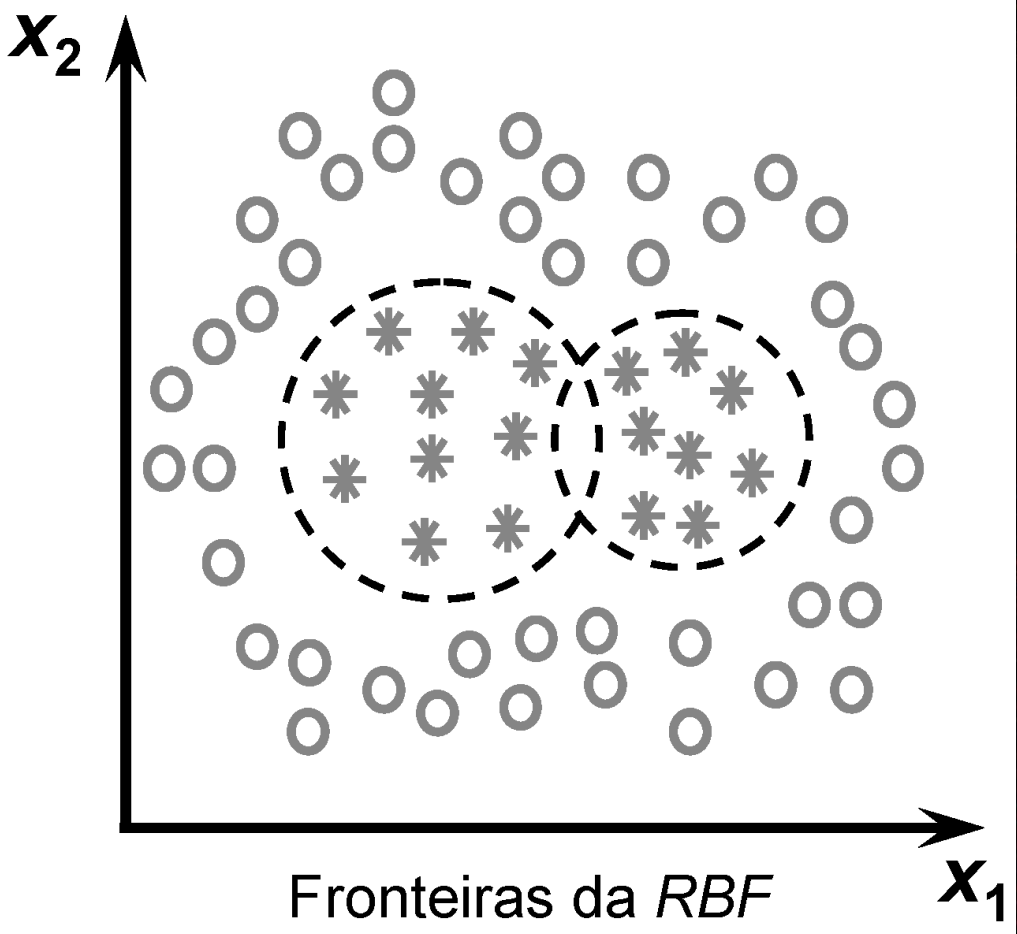




Aplicabilidade



Fronteiras do PMC



Fronteiras da *RBF*

Atividade

- Buscar e apresentar artigos que tem problemas resolvidos com a RBF;
  - Qual o problema?
  - Como é construído o conjunto de amostras?
  - Qual arquitetura da RBF?
  - Comparado com alguma outra rede ou forma de Inteligência Artificial?
  - Resultados?
  - Conclusão?

Referência

- Silva, IN da, Danilo Hernane Spatti, and Rogério Andrade Flauzino. "*Redes neurais artificiais para engenharia e ciências aplicadas*." São Paulo: Artliber (2010).

Obrigado.

Dúvidas?