Redes Neurais Não-recorrentes[1]

Índice Geral

[1]As Figuras 1, 25 e 26 foram elaboradas pelo Prof. Leandro N. de Castro, enquanto que a Figura 27 foi elaborada pela Profa. Andrea Carolina Peres Kulaif.

1. Neurônio artificial

- Modelo matemático: Simplificações da realidade com o propósito de representar aspectos relevantes de um sistema em estudo, sendo que detalhes de menor significância são descartados para viabilizar a modelagem.

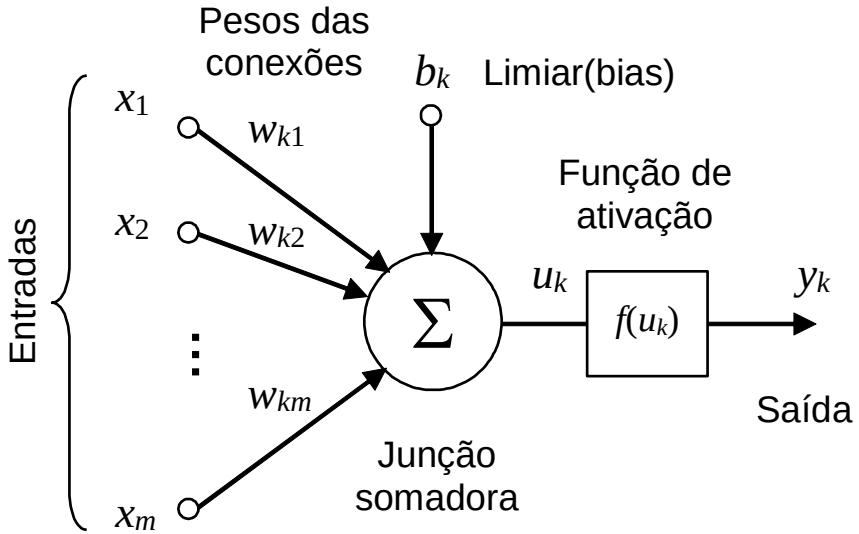


Figura 1 – Modelo matemático de um neurônio artificial.

- No modelo original de MCCULLOCH & PITTS (1943), $f(\cdot)$ é a função sinal.
- A saída do neurônio *k* pode ser descrita por: $y_k = f(u_k) = f\left(\sum_{j=1}^{m} w_{kj} x_j + b_k\right)$

- É possível simplificar a notação acima, de forma a incluir o limiar (*offset*), definindo um sinal de entrada de valor fixo $x_0 = 1$ com peso associado $w_{k0} = b_k$:

$$y_k = f(u_k) = f\left(\sum_{j=0}^{m} w_{kj} x_j\right) = f\left(\mathbf{w}^T \mathbf{x}\right)$$

1.1 Exemplos mais usuais de funções de ativação

- A função de ativação pode ser linear ou não-linear. No caso linear, vai corresponder à função identidade. No caso não-linear, ela pode evidenciar o limiar de ativação, por exemplo, ao se adotar a função sinal, ou o efeito de saturação, por exemplo, ao se adotarem as funções sigmoidais.
- Um desafio associado às funções sigmoidais é que elas são funções transcendentes (não podem ser expressas por uma combinação finita de operações algébricas), embora tenham derivadas de fácil computação. Com o advento das técnicas de *deep learning*, funções lineares por partes ganharam espaço, por apresentarem uma melhor relação custo-benefício (eficiência no ajuste de pesos × desempenho).

$$y_k = f(u_k) = \frac{e^{pu_k}}{e^{pu_k} + 1} = \frac{1}{1 + e^{-pu_k}} \qquad \frac{\partial y_k}{\partial u_k} = p y_k (1 - y_k) > 0$$

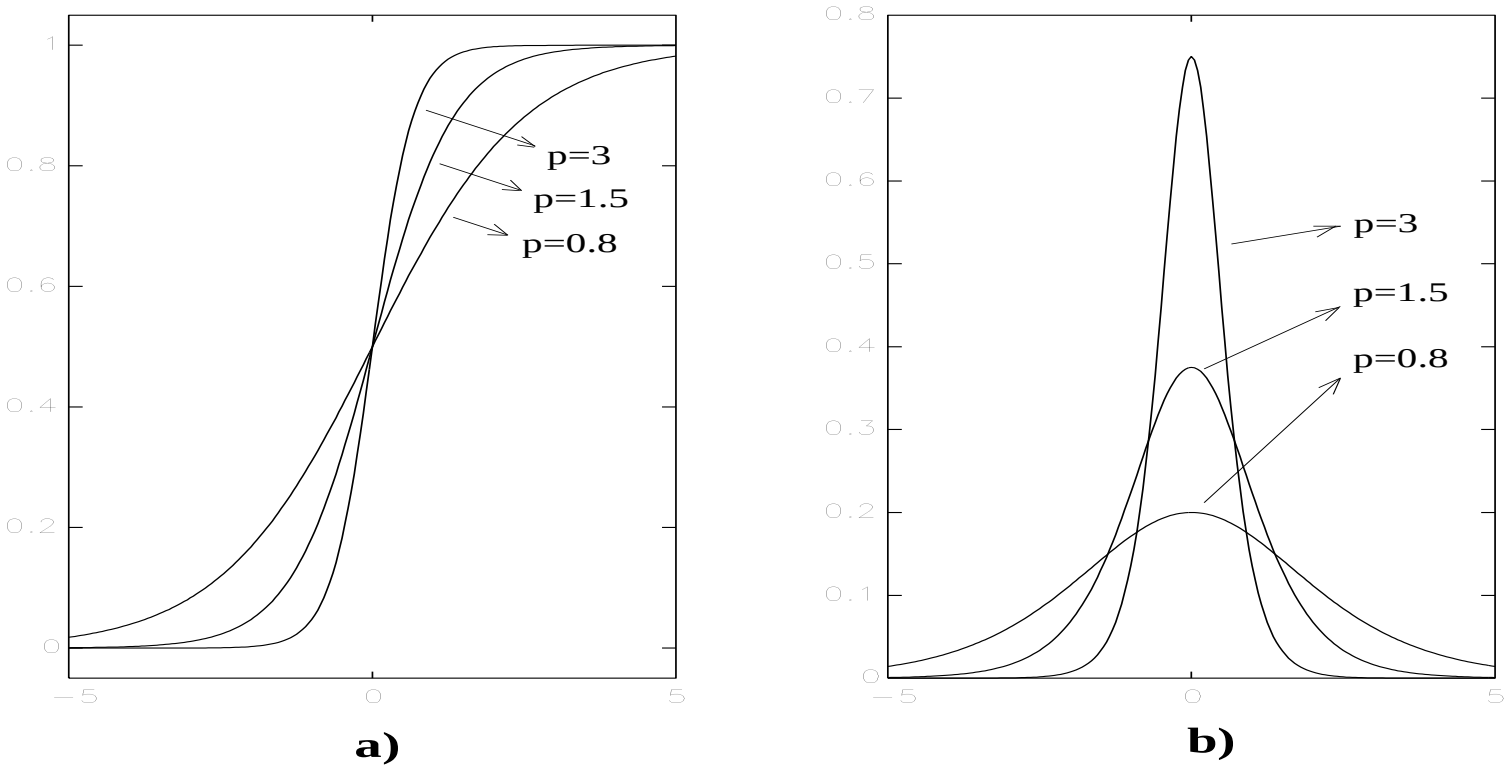


Figura 2 – Função logística (a) e sua derivada (b) em relação à entrada interna u_k.

$$y_k = f(u_k) = \tanh(pu_k) = \frac{e^{pu_k} - e^{-pu_k}}{e^{pu_k} + e^{-pu_k}} \qquad \frac{\partial y_k}{\partial u_k} = p\left(1 - y_k^2\right) > 0$$

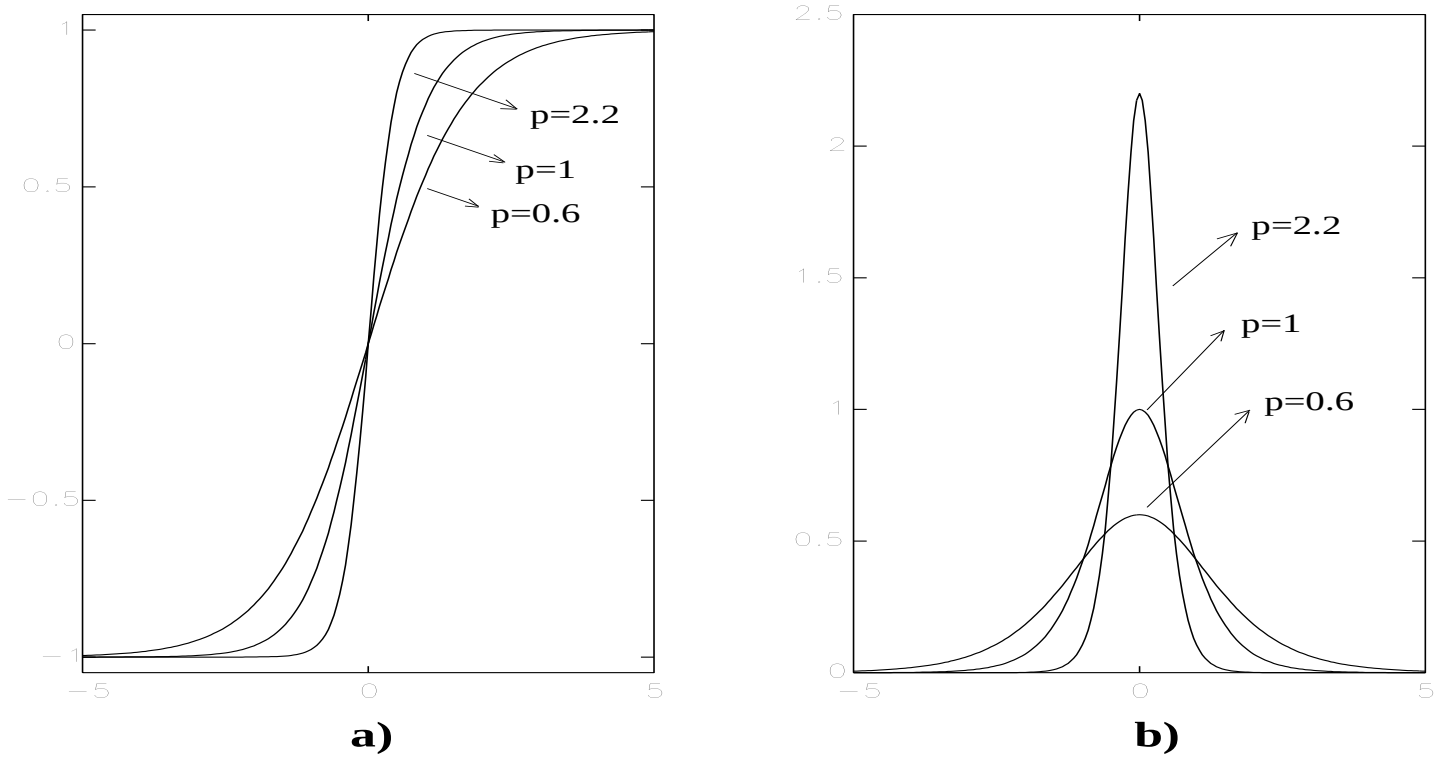


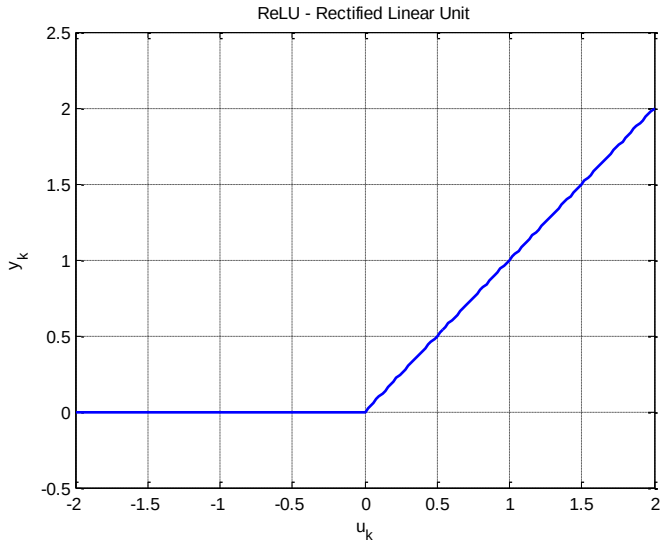
Figura 3 – Função tangente hiperbólica (a) e sua derivada (b) em relação à entrada interna u_k.

- Esta é uma das funções de ativação mais adotadas em implementações práticas do curso, com parâmetro p = 1 fixo.

- Uma função de ativação que passou a ser amplamente utilizada em *deep learning* é a Rectified Linear Unit (ReLU), apresentada a seguir e cuja equação é dada por:

$$y_k = f(u_k) = \max(0, u_k)$$

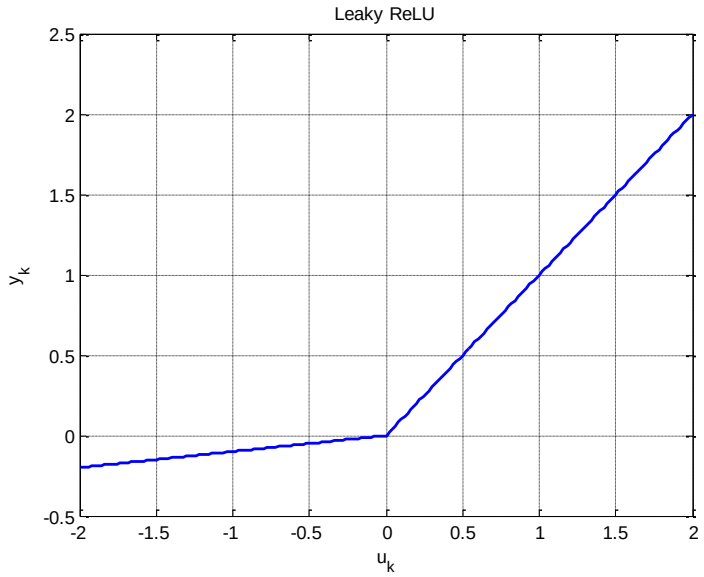
- Trata-se de uma função linear por partes, de fácil cômputo do gradiente e que opera como uma porta: a ativação interna do neurônio passa ou não passa.



- No entanto, a ReLU apresenta uma desvantagem em potencial. Caso muitos neurônios ativem com valores negativos para muitos estímulos de entrada, eles

deixam de participar do processamento da rede neural e não podem ter suas conexões sinápticas ajustadas, pois a função de ativação tem derivada nula. Com isso, uma opção válida é adotar a função Leaky ReLU, na forma:

$$y_k = \begin{cases} u_k & \text{se } u_k > 0 \\ \alpha u_k & \text{se } u_k \leq 0 \end{cases}, \text{ para } \alpha \text{ próximo de zero.}$$



- Variantes como ELU e SELU também têm sido consideradas. Para uma análise comparativa, consulte PEDAMONTI (2018).

2. Produto interno e projeção

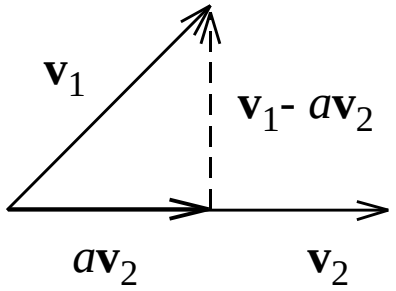


Figura 4 – Projeção realizada pelo produto interno no $\Re^2$

- Sejam $\mathbf{v}_1, \mathbf{v}_2 \in \Re^2$ elementos não-nulos. Considere um escalar a tal que $a\mathbf{v}_2$ corresponda à projeção de $\mathbf{v}_1$ na direção de $\mathbf{v}_2$. Então, pode-se afirmar que

$$a\mathbf{v}_2 \perp \mathbf{v}_1 - a\mathbf{v}_2,$$

  conduzindo a

$$\langle a\mathbf{v}_2, \mathbf{v}_1 - a\mathbf{v}_2 \rangle = 0 .$$

- Logo,

$$a\langle \mathbf{v}_2, \mathbf{v}_1 \rangle - a^2 \langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 0,$$

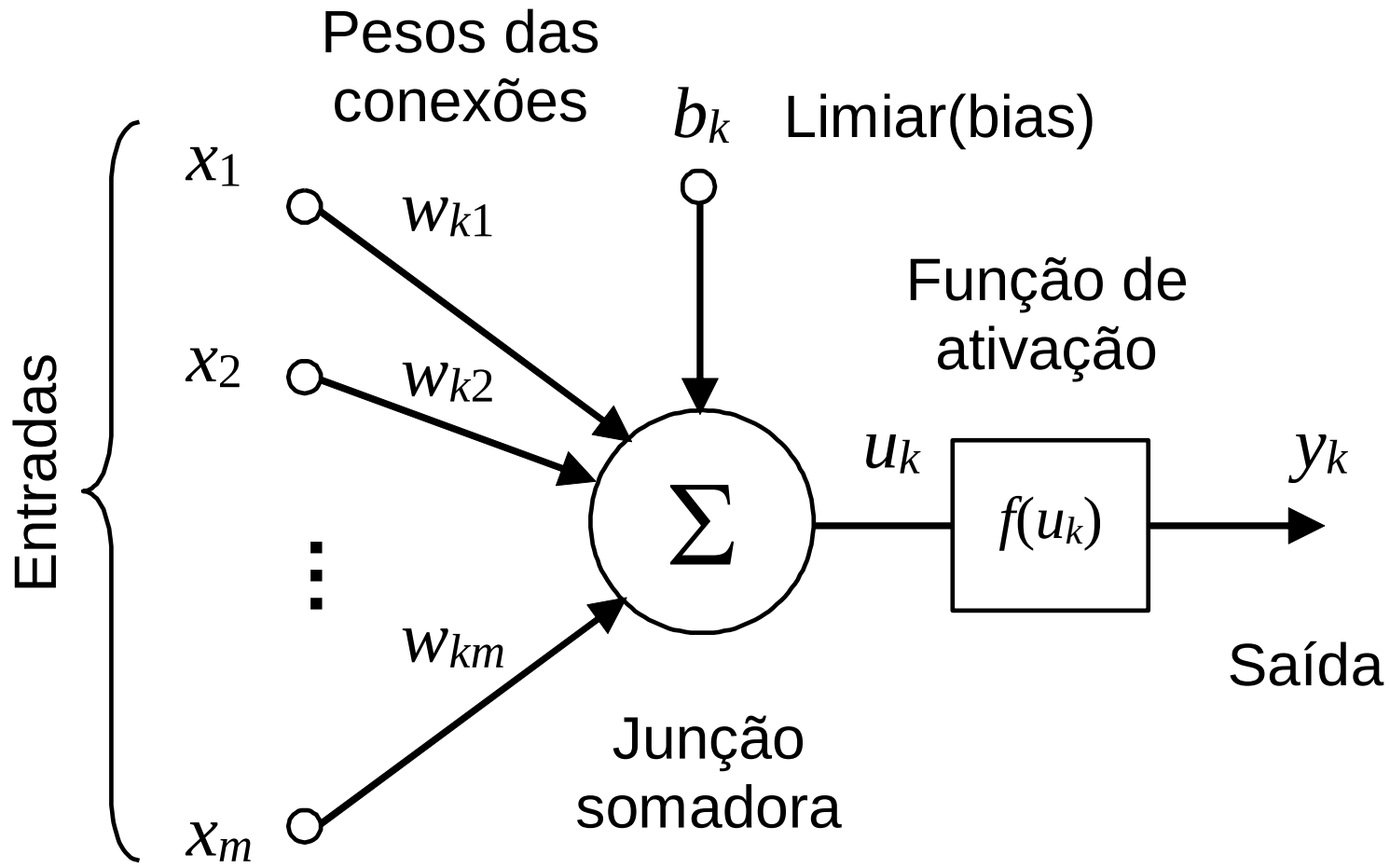
permitindo obter *a* na forma

$$a = \frac{\langle \mathbf{v}_2, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle}.$$

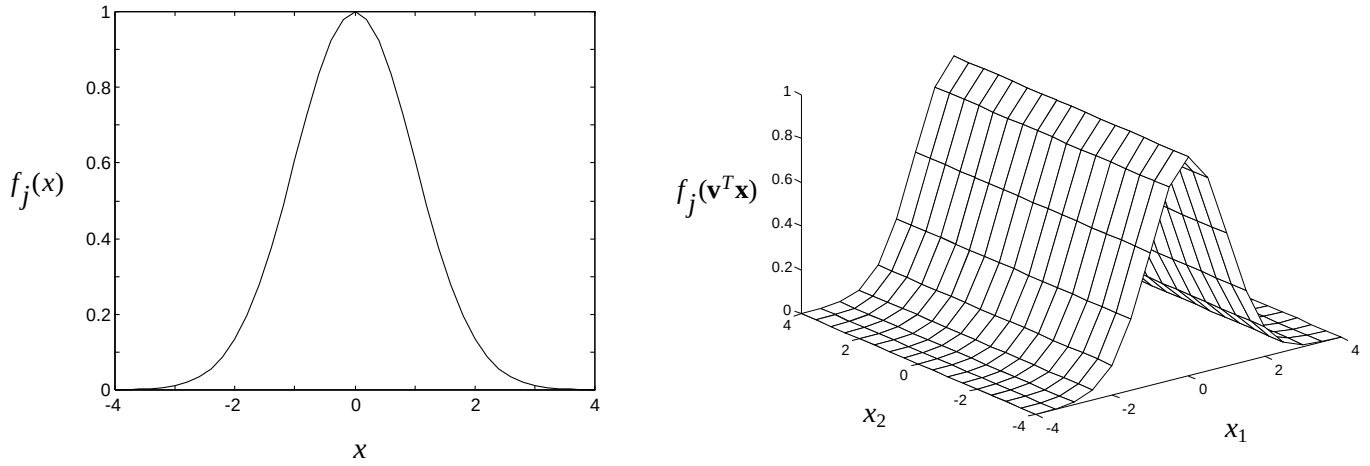
- Isto significa que a projeção de $\mathbf{v}_1$ na direção de $\mathbf{v}_2$ ($\mathbf{v}_2 \neq \mathbf{0}$) assume a forma:

$$\text{proj}_{\mathbf{v}_2}(\mathbf{v}_1) = \frac{\langle \mathbf{v}_2, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle} \mathbf{v}_2$$

- Mantendo constante o módulo de $\mathbf{v}_1$, a sua projeção na direção de $\mathbf{v}_2$ é tão maior quanto mais colineares forem esses dois vetores.
- O produto interno, portanto, permite o estabelecimento de uma associação bem definida entre o vetor de estímulos de entrada de um neurônio e o seu vetor de pesos sinápticos, de modo que o neurônio ativa mais quanto mais colineares forem esses dois vetores, supondo norma constante para esses vetores.


Pesos das conexões
b_k
Limiar(bias)
x_1
w_{k1}
Função de ativação
x_2
w_{k2}
Entradas
u_k
y_k
Σ
$f(u_k)$
w_{km}
Saída
Junção somadora
x_m

3. Função de expansão ortogonal



(a) $f_j(x) = e^{-0,5 \cdot x^2}$ (b) $f_j(\mathbf{v}^T\mathbf{x}) = e^{-0,5 \cdot \left([1\ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right)^2}$

Figura 5 – Função de expansão ortogonal em que $\mathbf{v} = [1\ 0]^T$ e $\mathbf{x} = [x_1\ \ x_2]^T$.

- A função de expansão ortogonal é conhecida na literatura em língua inglesa como *ridge function*.

4. Redes neurais e perceptron com uma camada intermediária

- O processo de conexão entre neurônios artificiais leva à geração de sinapses e à construção de redes neurais artificiais.

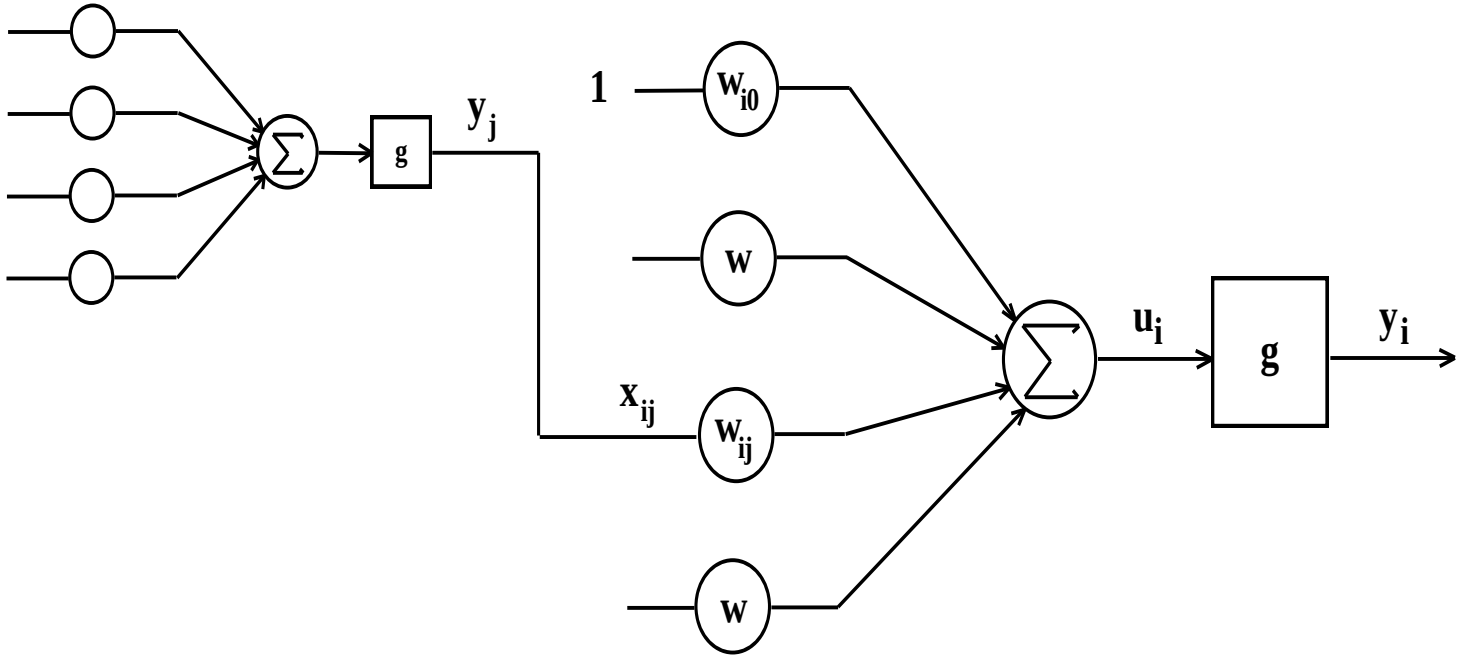


Figura 6 – Estabelecimento de conexão entre dois neurônios artificiais.

- Em estruturas com camadas plenamente conectadas, a saída de cada neurônio de uma camada precedente é entrada para todos os neurônios da camada seguinte.

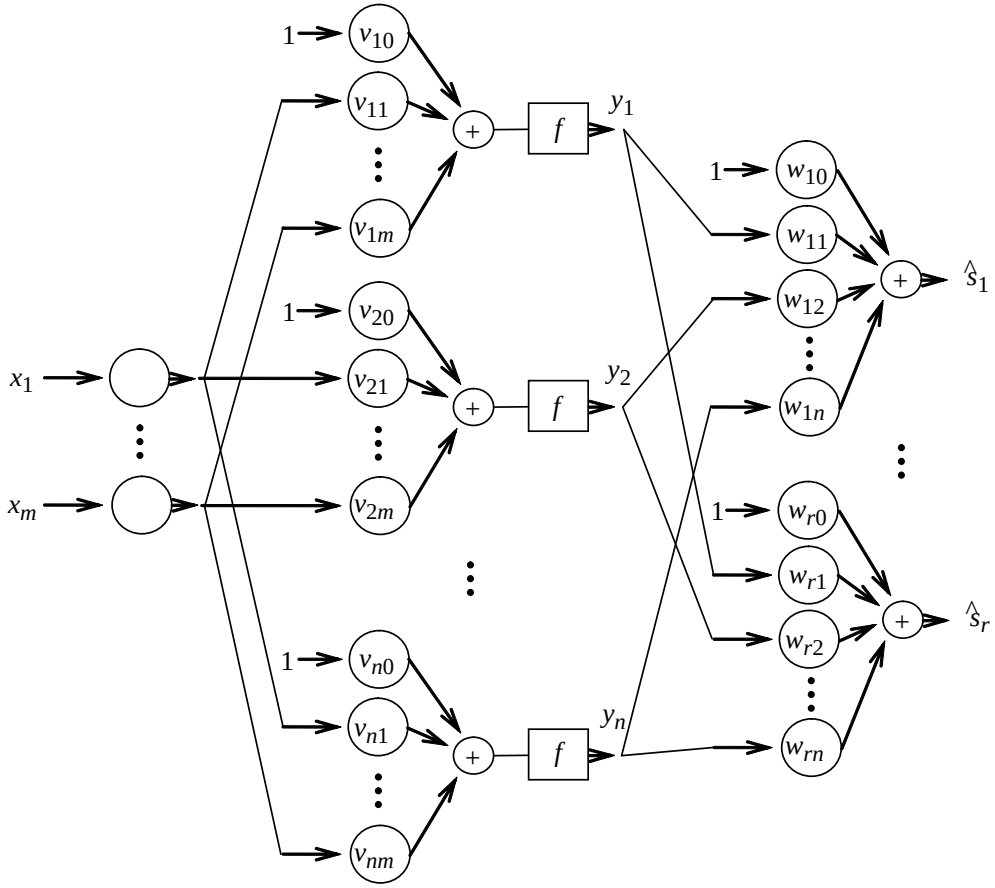
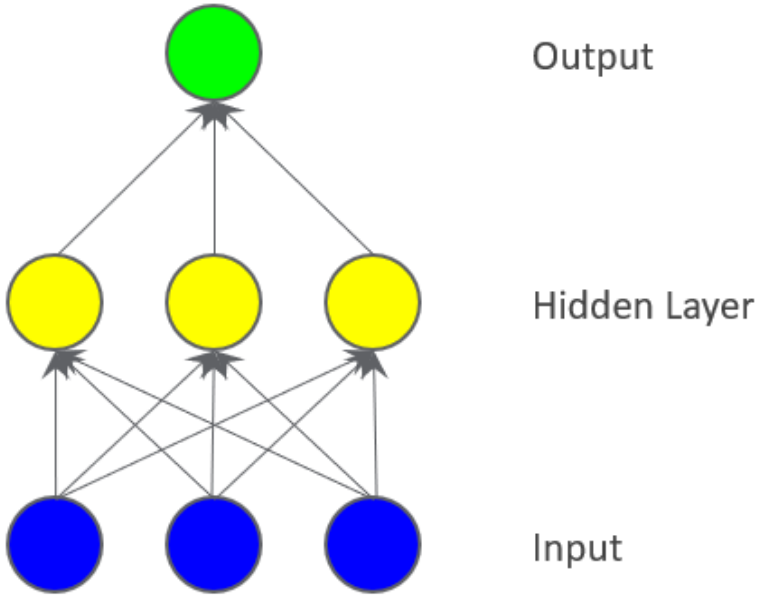


Figura 7 – Rede neural perceptron com uma camada intermediária.
Do inglês *Multilayer Perceptron* (MLP).

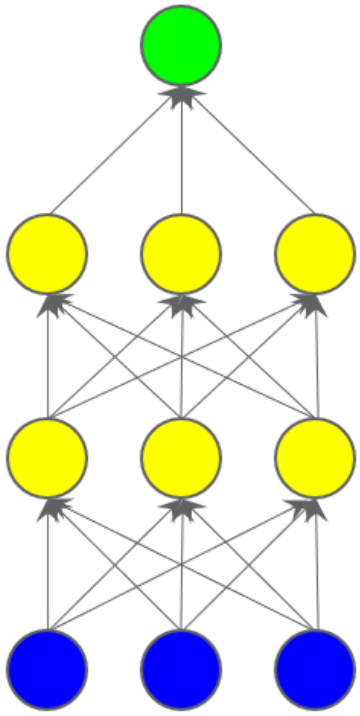
$$\hat{s}_k = \sum_{j=0}^{n} w_{kj} f\left(\sum_{i=0}^{m} v_{ji} x_i \right) = \sum_{j=0}^{n} w_{kj} f\left(\mathbf{v}_j^T \mathbf{x}\right) = \hat{g}_k\left(\mathbf{x}, \theta\right), k = 1,...,r$$

5. Múltiplas camadas com função de ativação linear

- Iremos mostrar aqui que, na eventualidade de se considerar apenas funções de ativação lineares, o uso de múltiplas camadas intermediárias não traz nenhum benefício ao processamento.



$$y = Wx + b$$



$$y = Wx + b$$

$$z = Vy + c$$

$$z = V(Wx+b) + c$$

$$= VWx + (Vb + c)$$

which is just another linear operation

6. Contribuição de cada neurônio em uma rede MLP

- O mapeamento não-linear realizado por uma rede neural do tipo perceptron de uma camada intermediária é uma **combinação linear de funções de expansão ortogonal**, ou seja, funções que têm a forma de tangente hiperbólica em uma direção e são constantes nas demais direções ortogonais a esta única direção em que a forma da função se manifesta.
- Como um exemplo, vamos tomar amostras de um mapeamento do $\Re^2$ para o $\Re^1$, e utilizar uma rede neural com 5 neurônios na camada intermediária para buscar aproximar este mapeamento, o qual pode ser visualizado no $\Re^3$.
- Os pesos sinápticos resultantes do processo de treinamento estão apresentados na sequência, sendo que a rede neural tem ao todo $3 \times 5 + 6 \times 1 = 21$ pesos ajustáveis. São 2 entradas, 5 neurônios na camada intermediária e 1 saída, mais as entradas constantes (entradas de limiar ou *offset*) de todos os 6 neurônios da rede neural.

- Pesos sinápticos da camada intermediária (cada coluna representa os pesos de um neurônio):

-0.20008939714462 -0.70051908010040 0.39699221844113 -0.10003863267278 0.69606262467282

0.70018168528932 0.10015860417667 0.19860028823484 -0.29996195303800 0.29869112235480

-0.30006398146599 0.80022209855791 0.49372400421686 0.50005427222963 0.89515012131364

- Pesos sinápticos da camada de saída:

0.99989340388393

0.79971888341317

0.90007841696146

0.38564988369799

0.79996881679466

0.71442550587375

- Obs: O peso de limiar é o primeiro peso de cada neurônio.

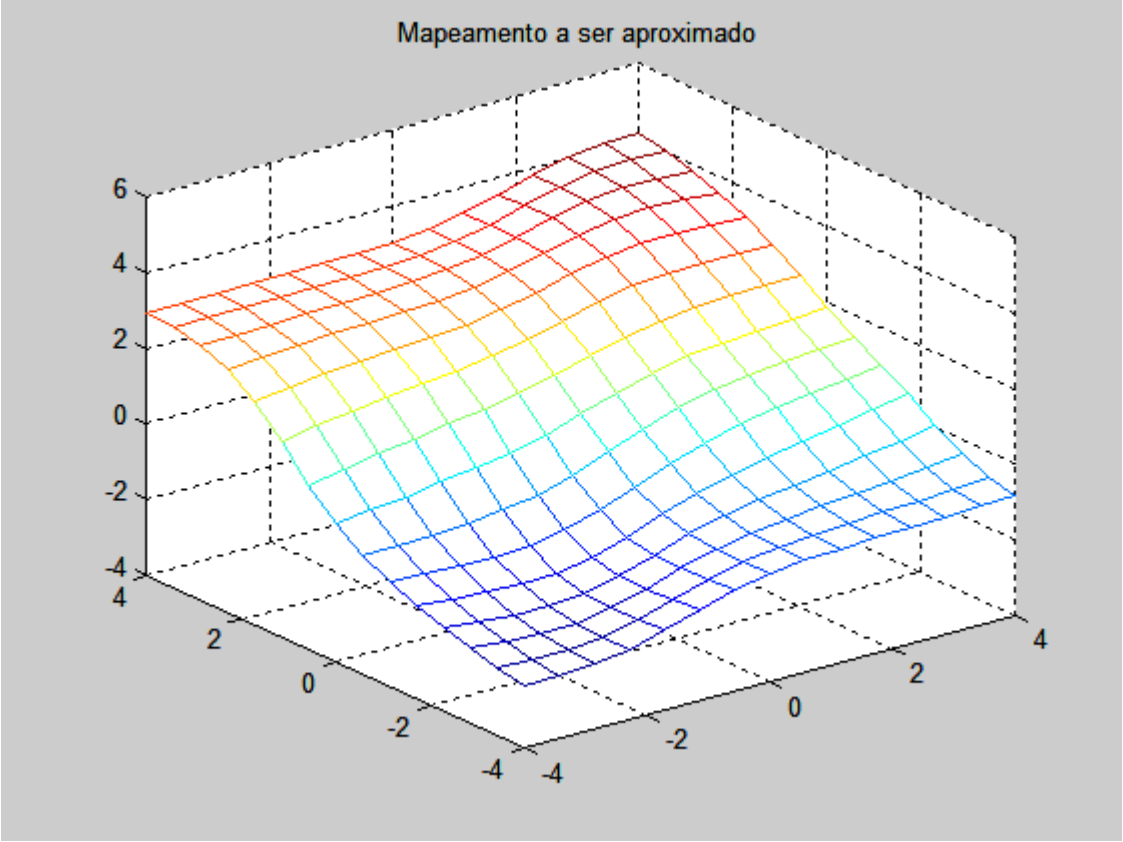


Figura 8 – Mapeamento a ser aproximado.

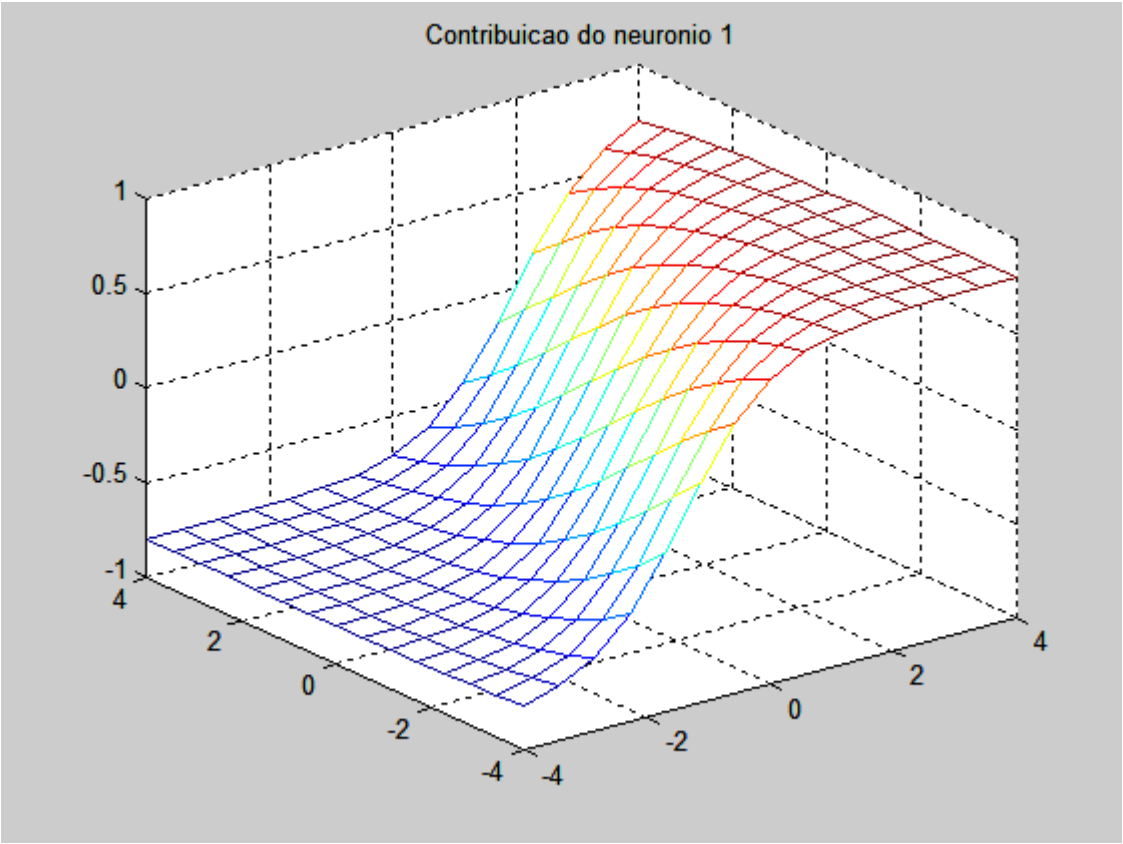


Figura 9 – Contribuição do neurônio 1, já multiplicada pelo peso do neurônio de saída.

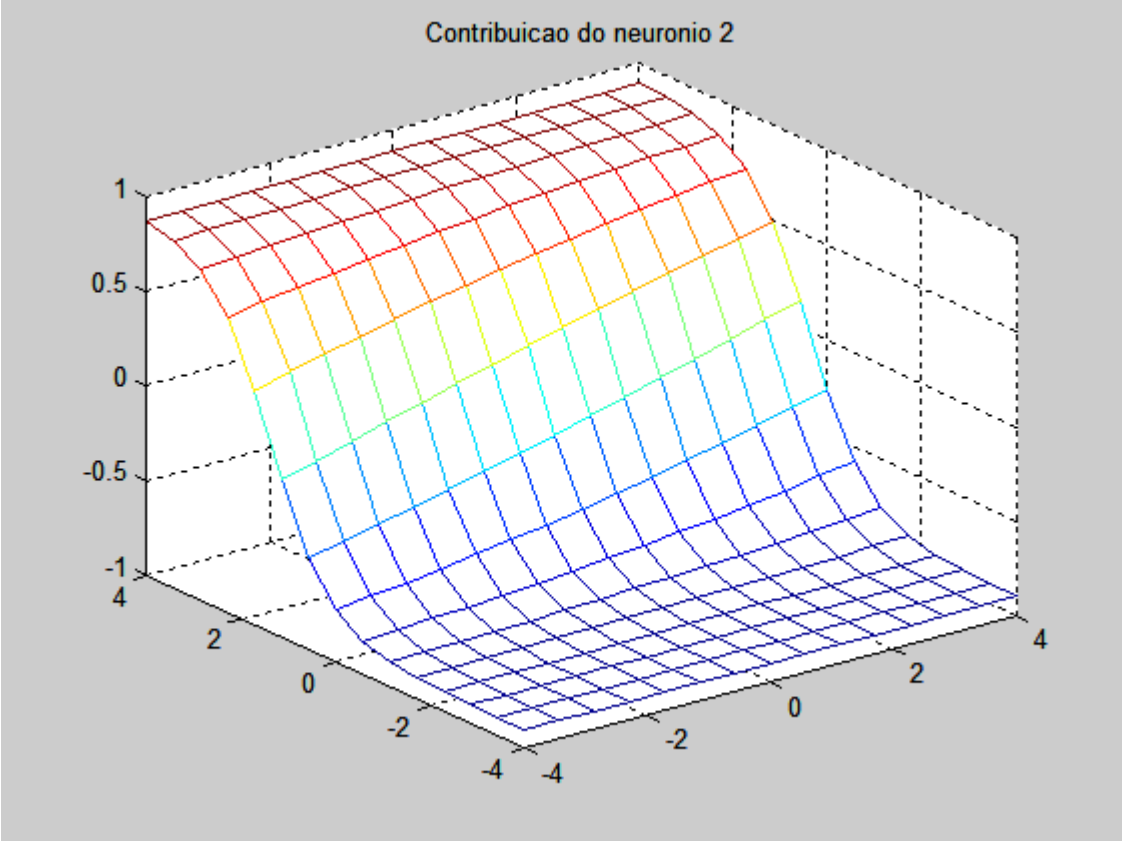


Figura 10 – Contribuição do neurônio 2, já multiplicada pelo peso do neurônio de saída.

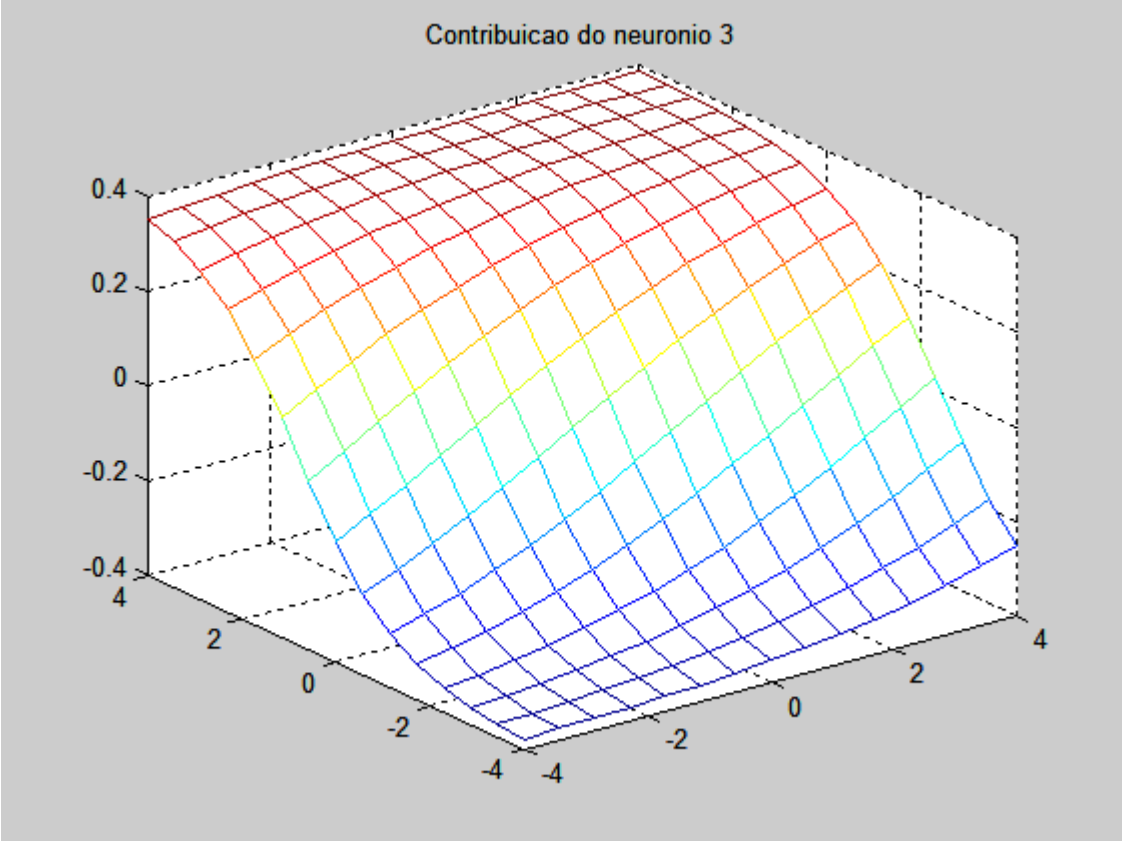


Figura 11 – Contribuição do neurônio 3, já multiplicada pelo peso do neurônio de saída.

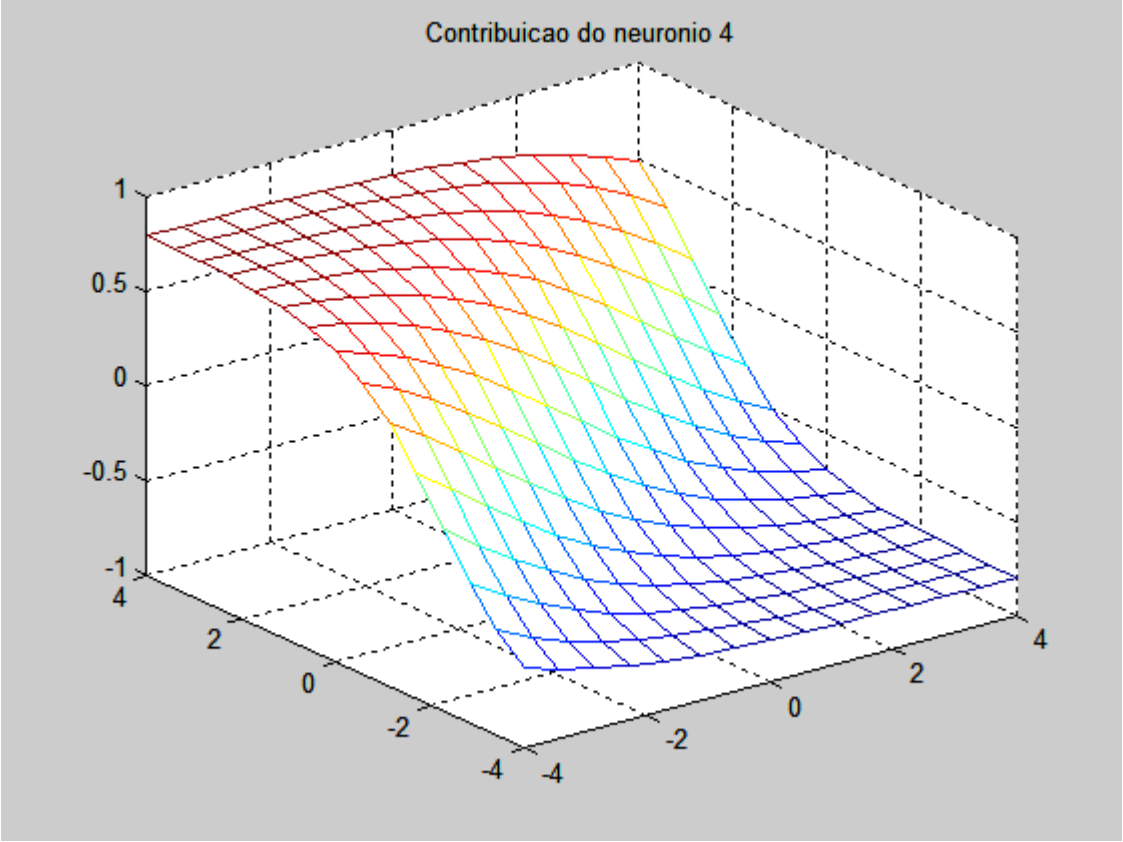


Figura 12 – Contribuição do neurônio 4, já multiplicada pelo peso do neurônio de saída.

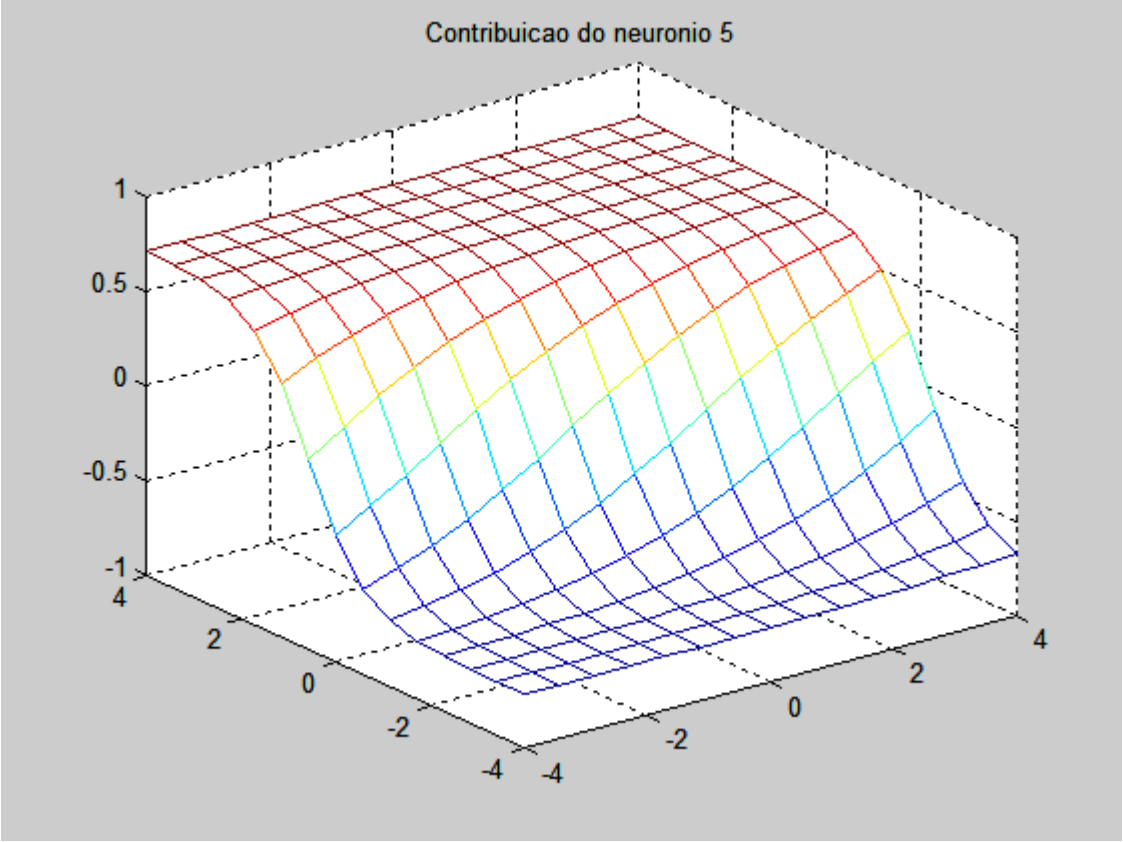
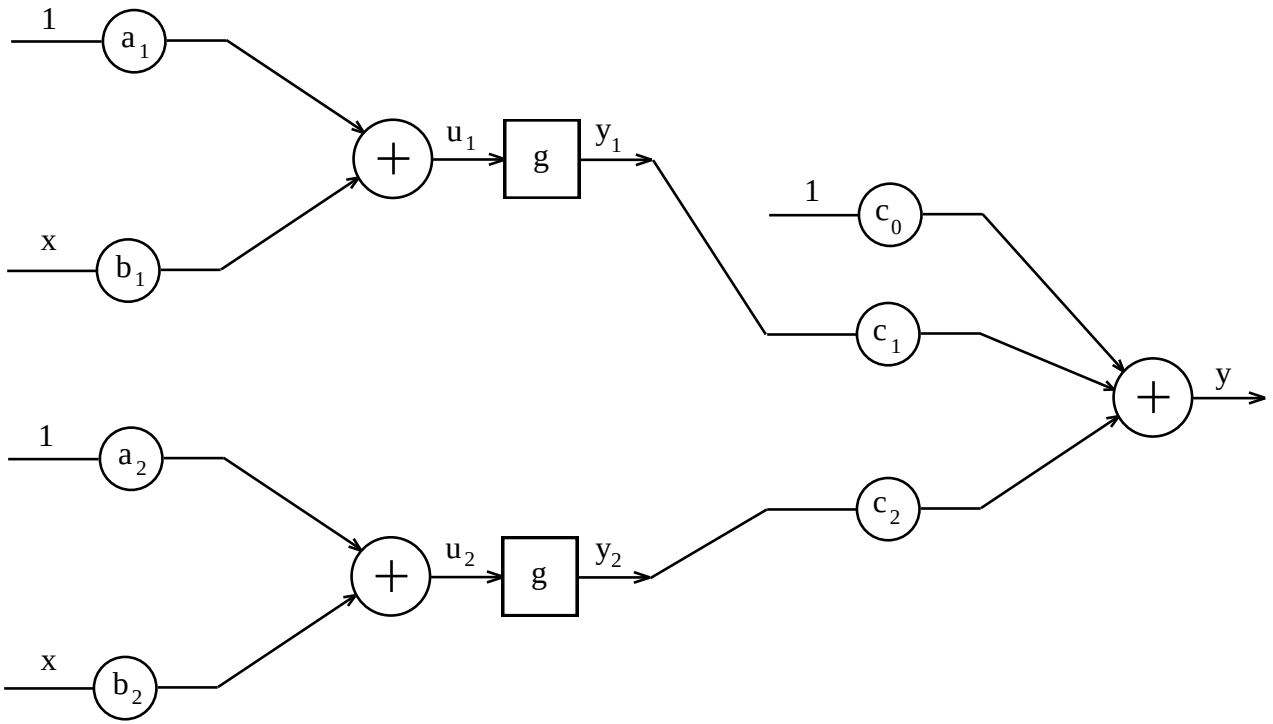


Figura 13 – Contribuição do neurônio 5, já multiplicada pelo peso do neurônio de saída.

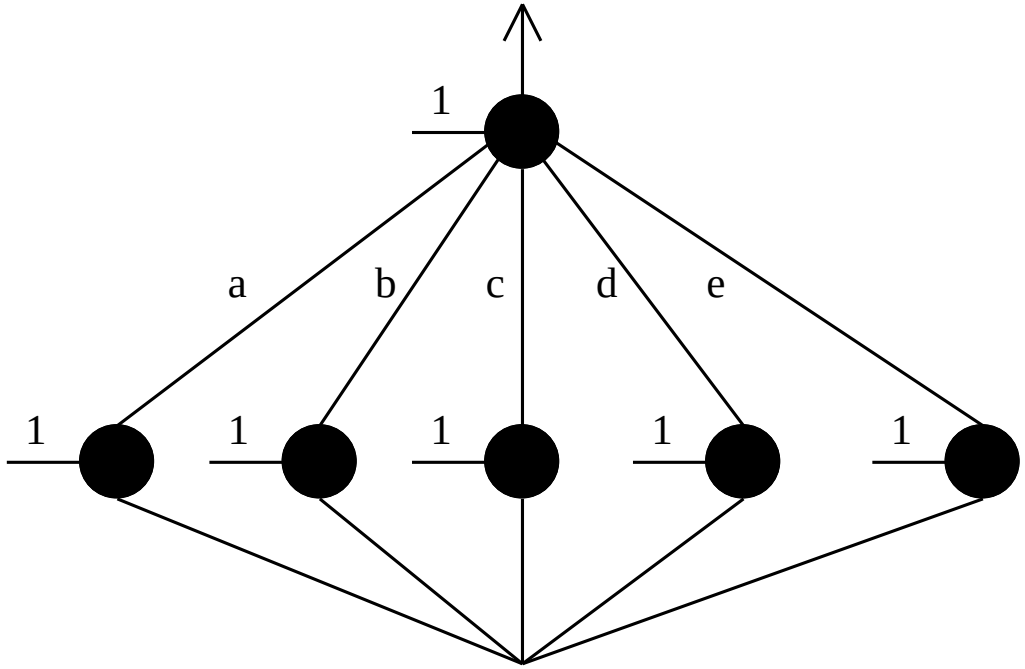
7. O papel dos pesos sinápticos

$$y = c_0 + \sum_{n=1}^{p} c_n g\left(b_n x + a_n\right)$$



$$y = c_0 + c_1 g\left(b_1 x + a_1\right) + c_2 g\left(b_2 x + a_2\right) \Rightarrow \begin{cases} \text{a : deslocamento no eixo x} \\ \text{b : inclinação da sigmóide} \\ \text{c : amplitude da sigmóide} \end{cases}$$

Exemplo: Forma "construtiva" de aproximação de um mapeamento não-linear empregando neurônios com função de ativação do tipo tangente hiperbólica. Exemplo considerando um único estímulo de entrada.



$$f(\mathbf{w}) = \underbrace{c_1 g(b_1 x + a_1)}_{a} + \underbrace{c_2 g(b_2 x + a_2)}_{b} + \underbrace{c_3 g(b_3 x + a_3)}_{c} + \underbrace{c_4 g(b_4 x + a_4)}_{d} + \underbrace{c_5 g(b_5 x + a_5)}_{e} + \underbrace{c_0}_{bias}$$

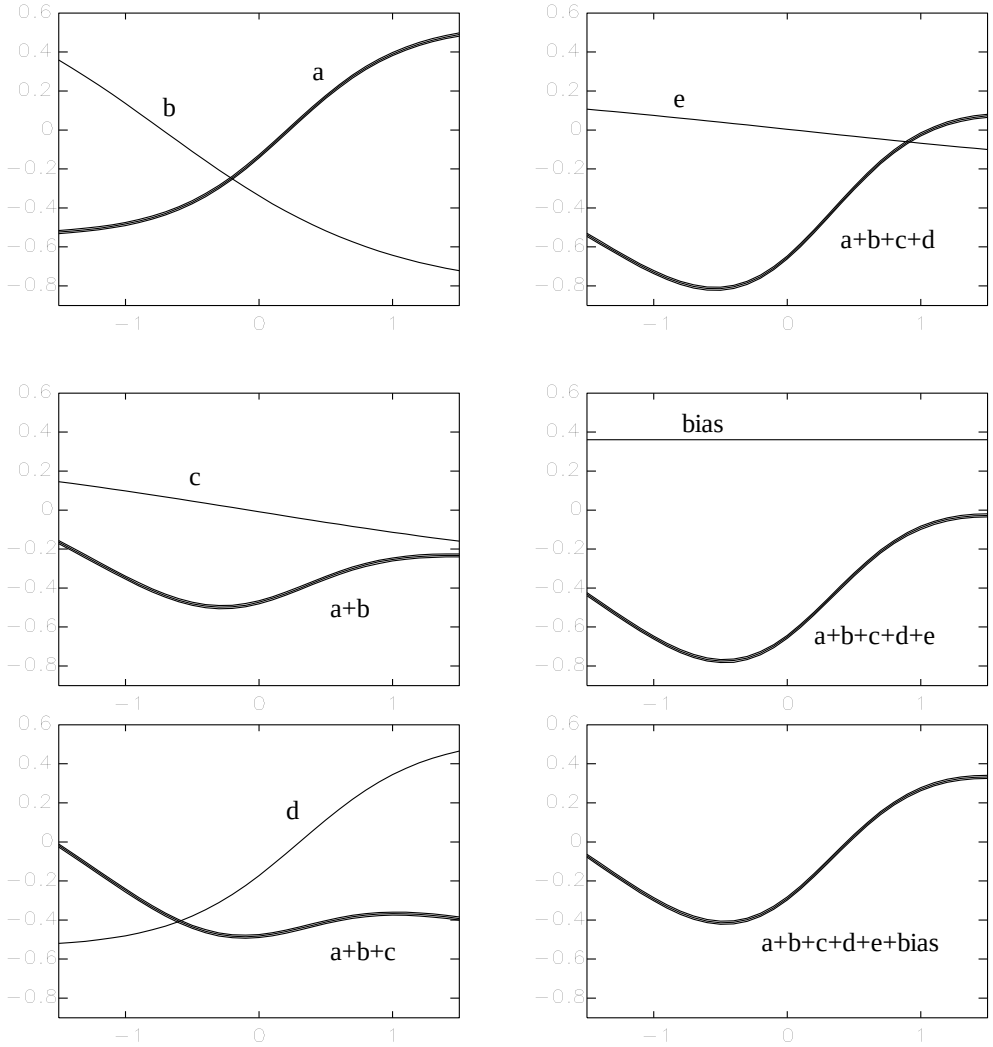


Figura 14 – Composição aditiva de ativações na reprodução de um mapeamento não-linear.

O mapeamento a ser aproximado se encontra na última figura à direita. Ler por coluna.

8. Superfície de erro

- Seja X uma região compacta do $\Re^m$ e seja $g: X \subset \Re^m \rightarrow \Re$ a função a ser aproximada (formulação para uma única saída, $r = 1$).
- Os dados de aproximação $\{(\mathbf{x}_l, s_l) \in \Re^m \times \Re\}_{l=1}^N$ são gerados considerando que os vetores de entrada $\mathbf{x}_l$ estão distribuídos na região compacta $X \subset \Re^m$ conforme uma função densidade de probabilidade $d_P: X \subset \Re^m \rightarrow [0,1]$ e que os vetores de saída s_l são produzidos pelo mapeamento definido pela função g na forma:

$$s_l = g(\mathbf{x}_l) + \varepsilon_l, \quad l = 1,\ldots,N,$$

onde $\varepsilon_l \in \Re$ é uma variável aleatória de média zero e variância fixa.

- A função g que associa a cada vetor de entrada $\mathbf{x} \in X$ uma saída escalar $s \in \Re$ pode ser aproximada com base no conjunto de dados de aproximação $\{(\mathbf{x}_l, s_l) \in \Re^m \times \Re\}_{l=1}^N$ por uma composição aditiva de funções de expansão ortogonal na forma:

$$\hat{s}_l = \hat{g}(\mathbf{x}_l, \theta) = \sum_{j=0}^{n} w_j \, f\left(\sum_{i=0}^{m} v_{ji} x_{li} \right) = \sum_{j=0}^{n} w_j \, f\left(\mathbf{v}_j^T \mathbf{x}_l \right)$$

onde θ é o vetor contendo todos os pesos da rede neural.

- Logo, o erro quadrático médio produzido na saída da rede neural, considerando as *N* amostras, assume a forma:

$$J(\theta) = \frac{1}{N} \sum_{l=1}^{N} (\hat{s}_l - s_l)^2 = \frac{1}{N} \sum_{l=1}^{N} (\hat{g}(\mathbf{x}_l, \theta) - s_l)^2 =$$

$$= \frac{1}{N} \sum_{l=1}^{N} \left(\sum_{j=0}^{n} w_j \, f\left(\sum_{i=0}^{m} v_{ji} x_{li} \right) - s_l \right)^2 = \frac{1}{N} \sum_{l=1}^{N} \left(\sum_{j=0}^{n} w_j \, f\left(\mathbf{v}_j^T \mathbf{x}_l \right) - s_l \right)^2$$

- Sendo *P* a dimensão do vetor θ, então tem-se que: $J : \Re^P \rightarrow \Re^1$.
- A superfície de erro definida por $J(\theta)$ reside no espaço $\Re^{P+1}$, sendo que se deve buscar em $\Re^P$ um ponto que minimiza $J(\theta)$, supondo que se queira minimizar o erro entre a saída produzida pelo rede neural e a saída desejada.

9. Aprendizado a partir de dados amostrados

- O aprendizado supervisionado pode então ser visto como um problema de otimização não-linear, onde a função-objetivo (critério de desempenho a ser otimizado) depende do vetor $\theta \in \Re^P$ de parâmetros ajustáveis:

$$\min_{\theta \in \Re^P} J(\theta)$$

- Repare que resultou um problema de otimização não-linear irrestrita, pois se admite qualquer valor real para os elementos do vetor θ. Como a função-objetivo $J(\theta)$ depende não-linearmente (ao menos de parte) dos elementos do vetor θ, não é possível produzir uma solução na forma fechada, devendo-se recorrer a um processo de ajuste iterativo, na forma:

$$\theta_{k+1} = \theta_k + \text{passo}_k * \text{direção}_k, \text{ com } \theta_0 \text{ fornecido.}$$

- A função objetivo, portanto, representa a formalização matemática do que se quer obter com o treinamento, e a expressão acima aponta para um método de solução.

- Evidentemente, é necessário definir um procedimento para a obtenção do passo e da direção de ajuste a cada iteração *k*, além da condição inicial e de um critério de parada do processo iterativo.
- Partindo-se, portanto, do conjunto de dados de treinamento $\{(\mathbf{x}_l, s_l) \in \Re^m \times \Re\}_{l=1}^N$, que na verdade contém pontos amostrados do mapeamento de entrada-saída que se quer sintetizar, existem 3 mapeamentos envolvidos no processo de treinamento supervisionado:
    1. O mapeamento a ser aproximado (do qual se conhece apenas pontos amostrados);
    2. O mapeamento resultante do processo de aproximação, fornecido pela rede neural após o treinamento;
    3. O mapeamento entre o vetor θ e $J(\theta)$, denominado superfície de erro.

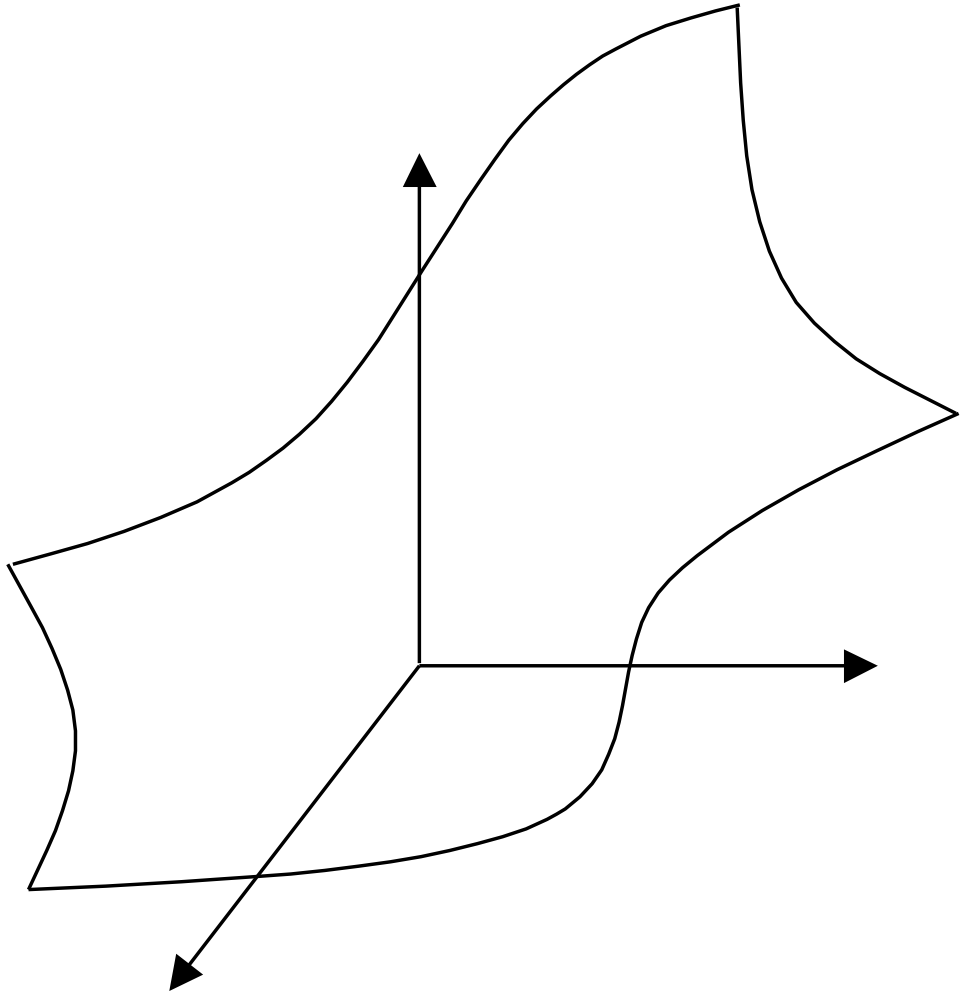


Figura 15– Mapeamento desconhecido a ser aproximado. Duas dimensões estão sendo consideradas, ou seja, $m = 2$, apenas para efeito de visualização.

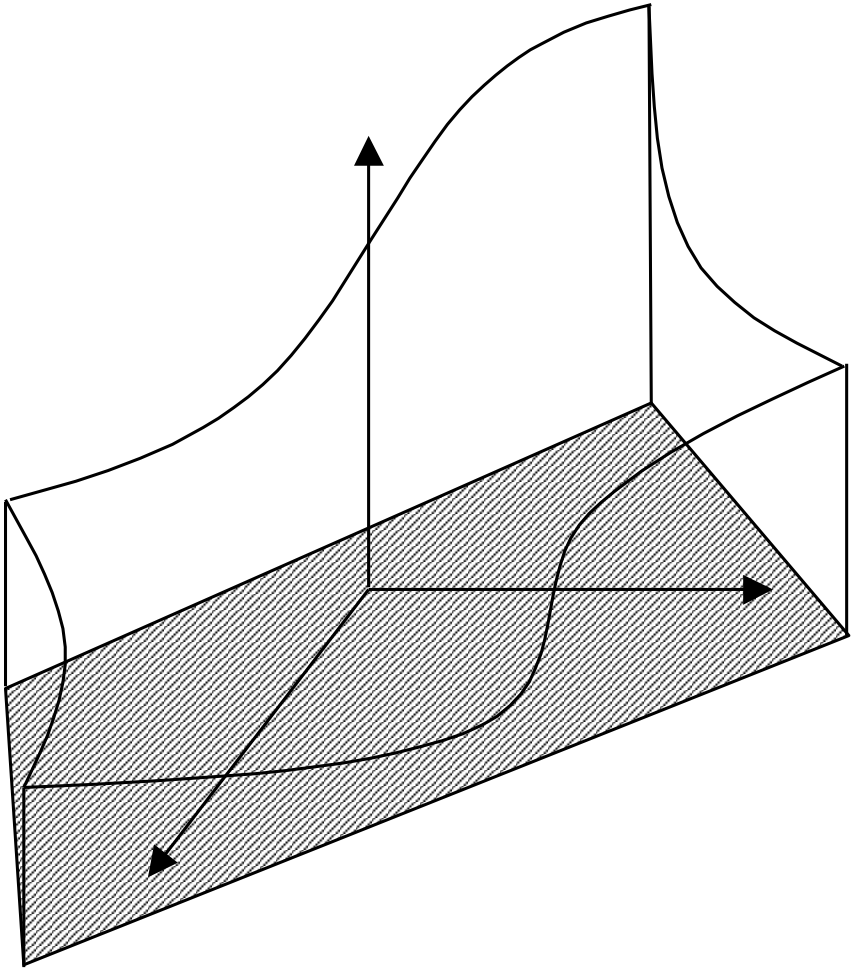


Figura 16 – Exemplo de região de operação. É uma região compacta (fechada e limitada).

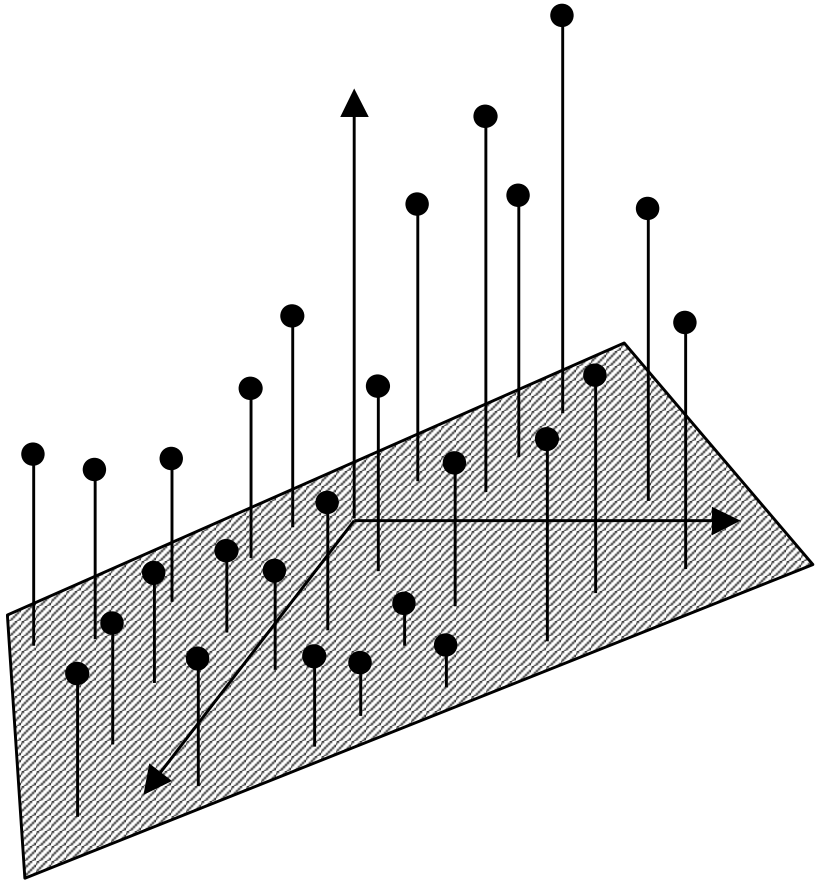


Figura 17 – Amostras expressando o comportamento da função para pontos específicos da região de operação. Essas amostras comporão os conjuntos de treinamento e validação (sendo que os dois conjuntos são independentes entre si).

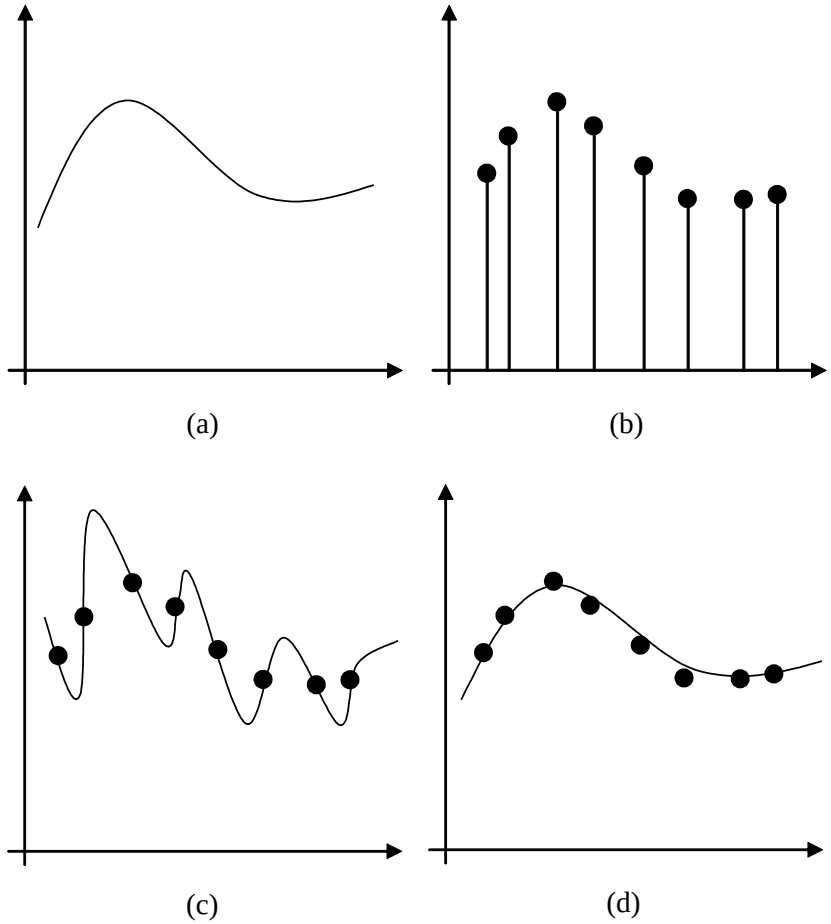


Figura 18 – (a) Mapeamento a ser aproximado (agora considerando apenas uma entrada); (b) Amostras disponíveis; (c) Resultado de um processo de aproximação com sobretreinamento; (d) Resultado de um processo de aproximação sem sobretreinamento.

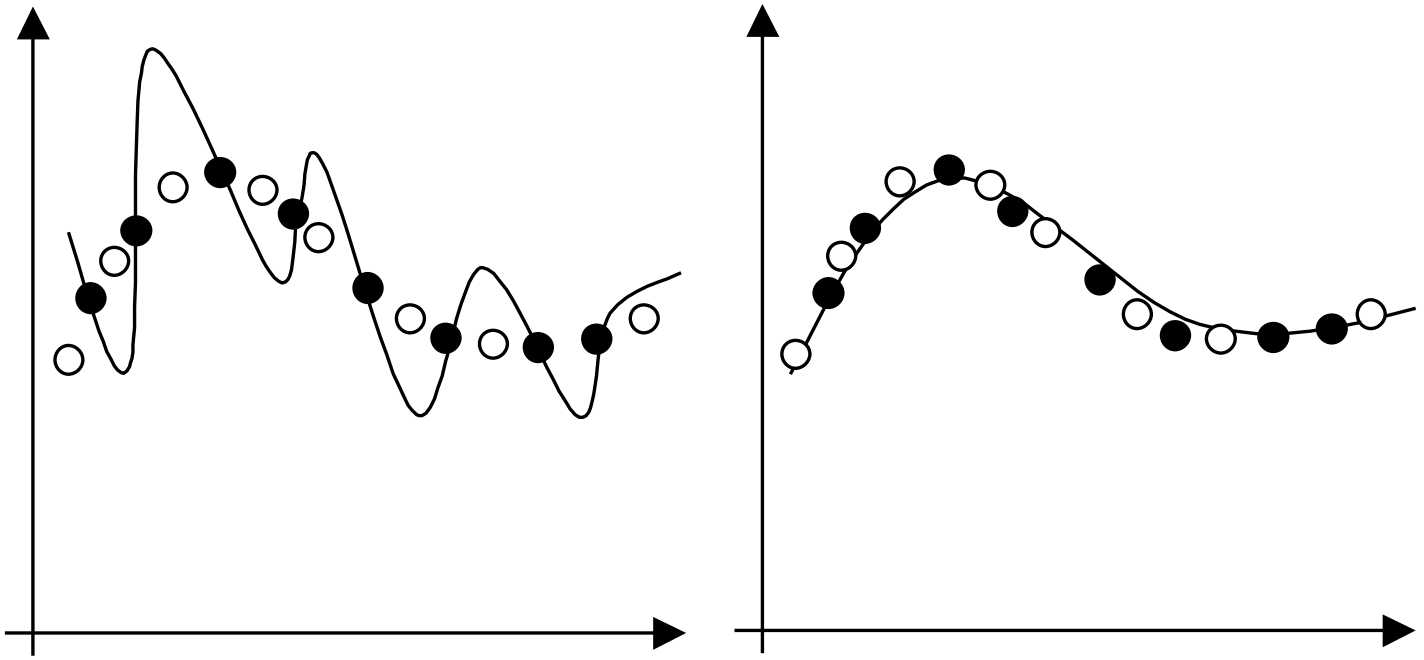


Figura 19 – Comparação de desempenho para dados de treinamento (pontos em preto) e validação (pontos em branco), de modo a medir a capacidade de generalização dos mapeamentos produzidos.

- O mapeamento da esquerda apresenta um erro de treinamento muito baixo, mas um erro de validação bastante elevado, quando comparado ao mapeamento da direita.

10. O problema do OU-exclusivo em MLP

- Considere os pontos (0,0),(0,1),(1,0) e (1,1) no plano $\Re^2$, conforme apresentado na Figura 20. O objetivo é determinar uma rede com duas entradas $\mathbf{x}_i \in \{0,1\}$ (i=1,2), e uma saída $\mathbf{y} \in \{0,1\}$ de maneira que: $\begin{cases} (\mathbf{x_1}, \mathbf{x_2}) = (0,0) \text{ ou } (1,1) \Rightarrow \mathbf{y} = 0 \\ (\mathbf{x_1}, \mathbf{x_2}) = (1,0) \text{ ou } (0,1) \Rightarrow \mathbf{y} = 1 \end{cases}$

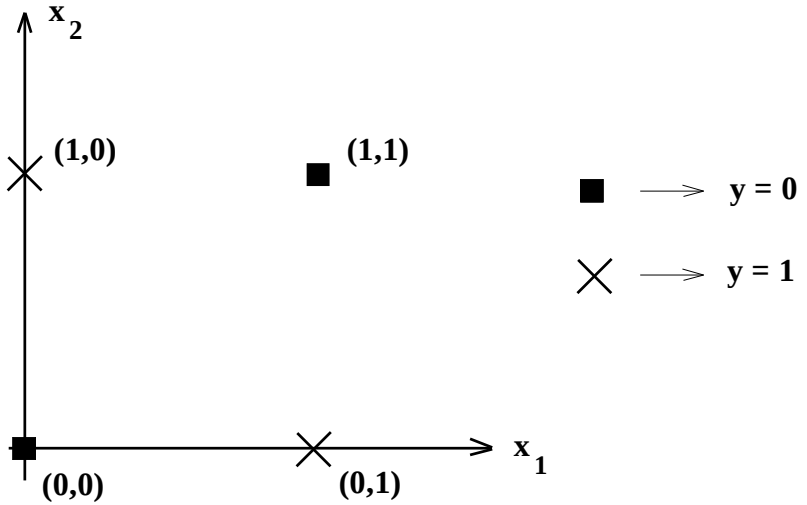


Figura 20 – O problema do OU-exclusivo

- Inicialmente será analisado o comportamento de um neurônio tipo perceptron (veja Figura 21) no processo de solução do problema exposto acima. A saída **y** pode ser representada na forma:

$$\mathbf{y} = \mathbf{g}(\mathbf{w_1 x_1} + \mathbf{w_2 x_2} + \mathbf{w_0}) \quad \text{onde} \quad \begin{cases} g(\mathbf{u}) = 1 \ \text{se} \ \mathbf{u} \geq 0 \\ g(\mathbf{u}) = 0 \ \text{se} \ \mathbf{u} < 0 \end{cases}$$

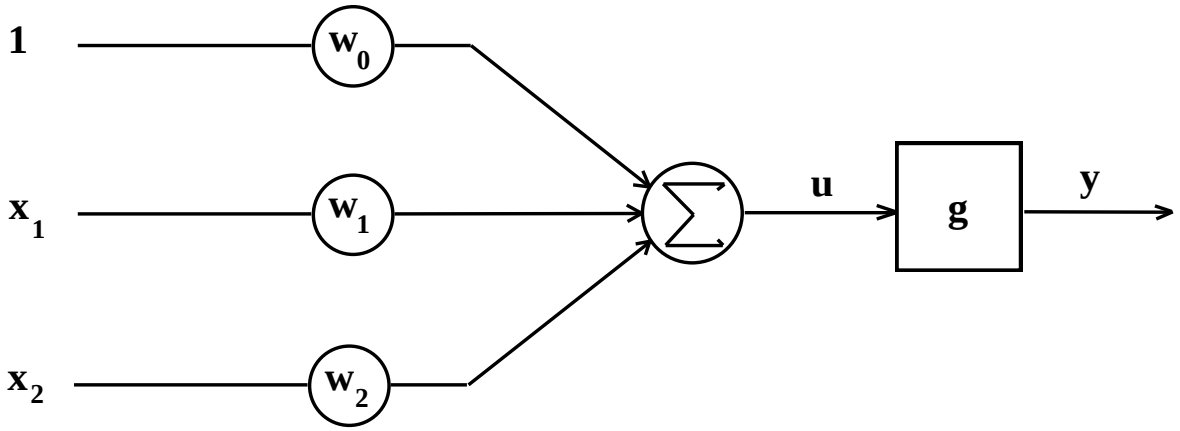


Figura 21 – Neurônio tipo perceptron, com duas entradas (mais a polarização)

- Para qualquer valor dos parâmetros $\mathbf{w_0}$, $\mathbf{w_1}$ e $\mathbf{w_2}$, a função $\mathbf{g(u)}$ separa o espaço de entradas em duas regiões, sendo que a curva de separação é uma linha reta.

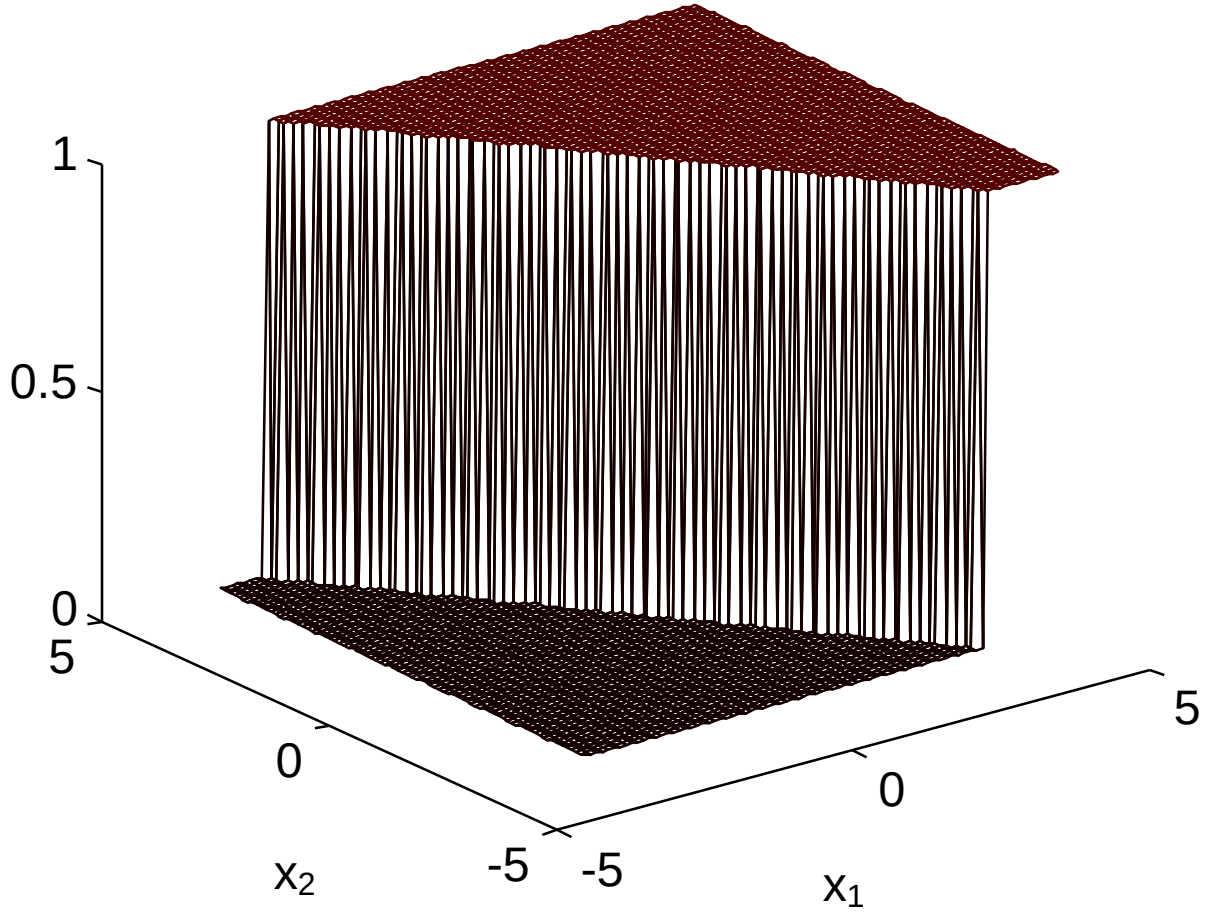


Figura 22 – Mapeamento de entrada-saída para o perceptron da Figura 21, com $\mathbf{w_0} = -6$, $\mathbf{w_1} = 4$ e $\mathbf{w_2} = 3$

- Aqui se tomou a função g(·) como sendo a função sinal, pois as saídas são binárias.

- No problema do OU-exclusivo (Figura 20), pode-se constatar que não existe uma única linha reta divisória de forma que os pontos (0,0) e (1,1) se posicionem de um lado enquanto que (0,1) e (1,0) permaneçam do outro lado da linha.
- Logo, pode-se imediatamente concluir que um neurônio tipo perceptron não apresenta grau de liberdade suficiente para resolver o problema proposto, o que foi corretamente constatado por MINSKY & PAPERT (1969).
- No entanto, esses autores também acreditavam que não havia razão para supor que redes multicamadas pudessem conduzir a uma solução para o problema proposto. Esta hipótese só foi definitivamente rejeitada com o desenvolvimento do algoritmo de retro-propagação (*back-propagation*), já nos anos 80 (RUMELHART & MCCLELLAND, 1986), o qual permite o ajuste automático de pesos para redes neurais multicamadas, arquitetura necessária para a realização de mapeamentos não-lineares arbitrários.

- Considere o problema de mapeamento de uma rede neural tipo perceptron, com uma camada intermediária (Figura 23), aplicada ao problema do OU-exclusivo.

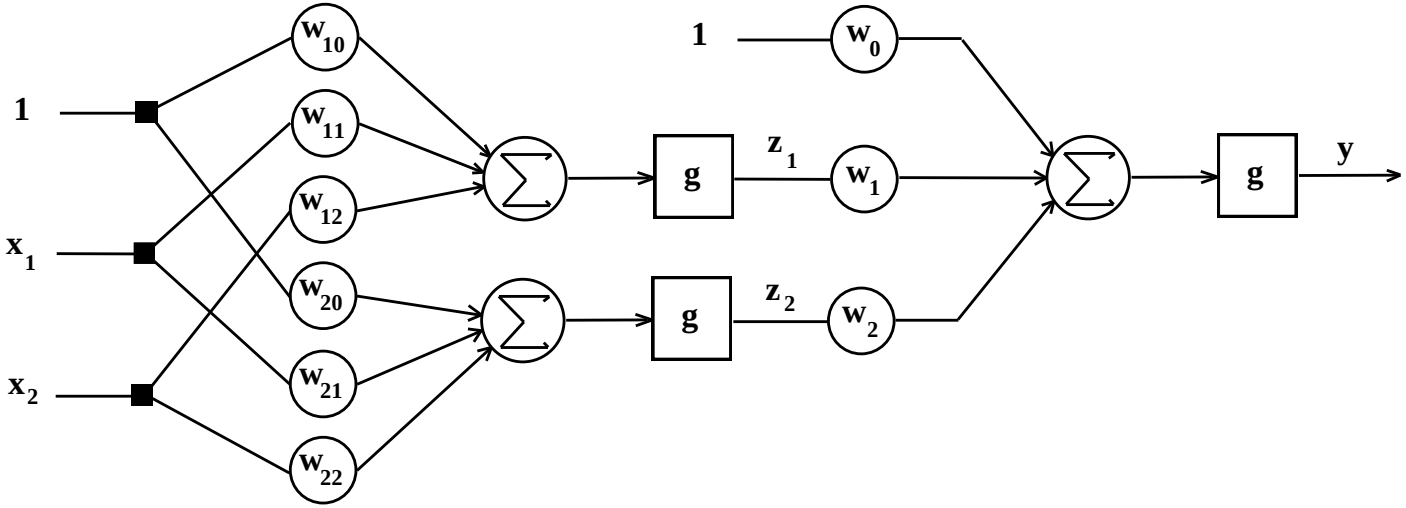


Figura 23 – Perceptron de três camadas (uma camada intermediária).

- A camada de entrada fornece um vetor de entrada ($\mathbf{x_1},\mathbf{x_2}$) para a camada intermediária, enquanto que a camada intermediária produz duas saídas $\mathbf{z_1}=\text{sgn}(\mathbf{w_{10}}+\mathbf{w_{11}x_1}+\mathbf{w_{12}x_2})$ e $\mathbf{z_2}=\text{sgn}(\mathbf{w_{20}}+\mathbf{w_{21}x_1}+\mathbf{w_{22}x_2})$. Na camada de saída, o sinal de saída da rede neural é dado por $\mathbf{y}=\text{sgn}(\mathbf{w_0}+\mathbf{w_1z_1}+\mathbf{w_2z_2})$.

- Surge uma questão: existem parâmetros $\mathbf{w_{ij}}$ (i=1,2; j=0,1,2) e $\mathbf{w_k}$ (k = 0,1,2) tais que **y** = 0 para as entradas (0,0) e (1,1) e **y** = 1 para as entradas (1,0) e (0,1)?
- As saídas da primeira camada ($\mathbf{z_1}$ e $\mathbf{z_2}$) podem ser consideradas como variáveis intermediárias utilizadas na geração da saída **y**. As variáveis $\mathbf{z_1}$ e $\mathbf{z_2}$ formam o que será denominado mais à frente no curso de espaço de características.
- Do que já foi visto a respeito de um neurônio tipo perceptron, sabe-se que existem pesos $\mathbf{w_{1j}}$ (j=0,1,2) tais que (veja curva de separação $\mathbf{L_1}$ na Figura 24(a)):

(0,1) produza $\mathbf{z_1} = 1$

(0,0),(1,0),(1,1) produza $\mathbf{z_1} = 0$.

- De forma similar, existem pesos $\mathbf{w_{2j}}$ (j=0,1,2) tais que (veja curva de separação $\mathbf{L_2}$ na Figura 24(a)):

(0,1),(0,0),(1,1) produza $\mathbf{z_2} = 1$

(1,0) produza $\mathbf{z_2} = 0$

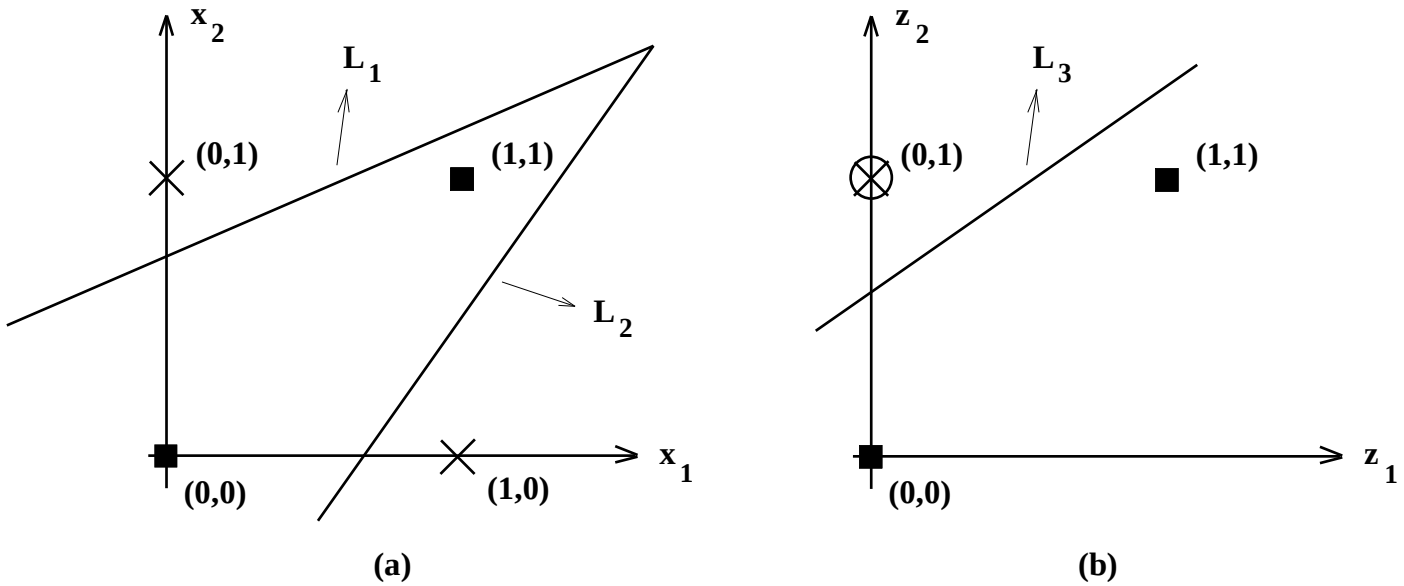


Figura 24 – Realização da função OU-exclusivo

- A discussão acima mostra que existem pesos $\mathbf{w_{ij}}$ (i=1,2; j=0,1,2) de maneira que a entrada (0,1) resulte em $\mathbf{z_1} = 1$, $\mathbf{z_2} = 1$, e a entrada (1,0) resulte em $\mathbf{z_1} = 0$, $\mathbf{z_2} = 0$, enquanto que (0,0) e (1,1) produzam $\mathbf{z_1} = 0$, $\mathbf{z_2} = 1$. Já que (0,0) e (1,1) podem ser separados linearmente de (0,1), como mostrado na Figura 24(b) pela curva de separação $\mathbf{L_3}$, pode-se concluir que a função booleana desejada pode ser obtida utilizando-se perceptrons em cascata, ou seja, 3 neurônios do tipo perceptron.

11. Otimização não-linear e capacidade de generalização

- Diversos tipos de parâmetros da rede neural poderiam ser submetidos a processos de ajuste durante o treinamento, como (*i*) pesos sinápticos; (*ii*) parâmetros da função de ativação de cada neurônio; (*iii*) número de neurônios na camada intermediária; (*iv*) número de camadas intermediárias.
- Iremos nos restringir aqui ao ajuste dos pesos sinápticos. Neste caso, o processo de treinamento supervisionado de redes neurais artificiais multicamadas é equivalente a um problema de otimização não-linear irrestrita, em que a superfície de erro é minimizada a partir do ajuste dos pesos sinápticos.
- Iremos nos restringir também a redes MLP com uma única camada intermediária, visto que com apenas uma camada intermediária a rede neural já apresenta **capacidade de aproximação universal** (CYBENKO, 1989; HORNIK *et al.*, 1989; HORNIK *et al.*, 1990; HORNIK *et al.*, 1994).

- Um problema comum a todos os modelos de aproximação de funções que possuem capacidade de aproximação universal, não apenas redes neurais artificiais do tipo MLP, é a necessidade de controlar adequadamente o seu grau de flexibilidade.
- Como o conjunto de amostras disponível para treinamento supervisionado é finito, infinitos mapeamentos podem produzir o mesmo desempenho de aproximação, independente do critério de desempenho adotado. Esses mapeamentos alternativos vão diferir justamente onde não há amostras disponíveis para diferenciá-los.
- Visando maximizar a **capacidade de generalização** do modelo de aproximação (no caso, uma rede neural MLP), ou seja, buscando encontrar o grau de flexibilidade adequado para o modelo de aproximação (dada a demanda da aplicação), um procedimento recomendado é dividir o conjunto de amostras disponível para treinamento em dois: um conjunto que será efetivamente empregado no ajuste dos pesos (conjunto de treinamento) e um conjunto que será

empregado para definir o momento de interromper o treinamento (conjunto de validação).

- Deve-se assegurar que ambos os conjuntos sejam suficientemente representativos do mapeamento que se pretende aproximar. Assim, minimizar o erro junto ao conjunto de validação implica em maximizar a capacidade de generalização. Logo, espera-se que a rede neural que minimiza o erro junto ao conjunto de validação (não usado para o ajuste dos pesos) tenha o melhor desempenho possível junto a novas amostras.
- A figura alto/esquerda a seguir mostra um mapeamento unidimensional a ser aproximado (desconhecido pela rede neural) e amostras sujeitas a ruído de média zero (única informação disponível para o treinamento da rede neural). A figura alto/direita mostra o resultado da aproximação produzida por uma rede neural com poucos neurônios, a qual foi incapaz de realizar a aproximação (tem baixa flexibilidade).

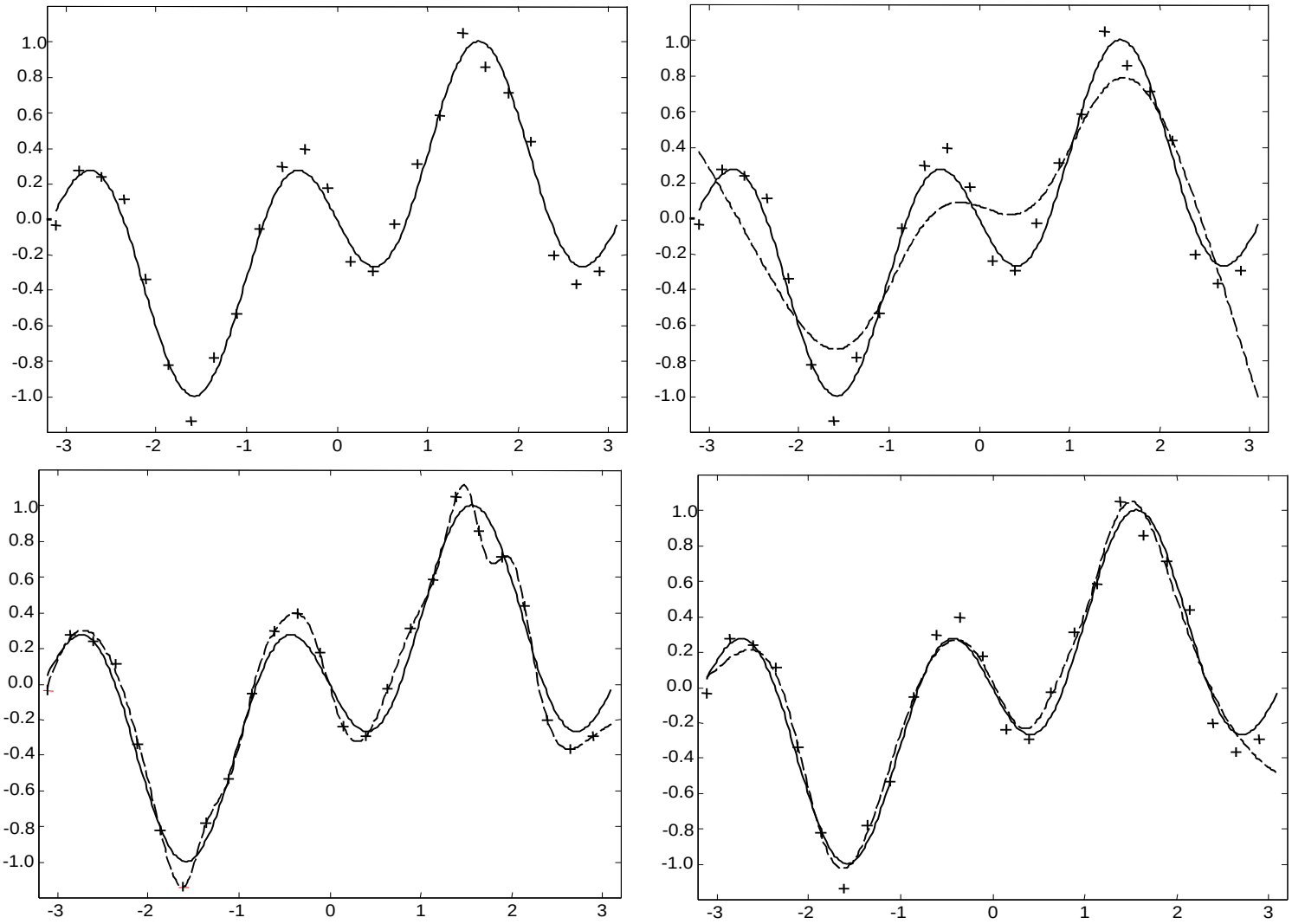


Figura 25 – Dados amostrados, função a ser aproximada e modelos de aproximação com diferentes capacidades de generalização

- Já as figuras baixo/esquerda e baixo/direita mostram o resultado de uma mesma rede neural (com número suficiente de neurônios), mas à esquerda ocorreu sobre-treinamento, enquanto à direita o treinamento foi interrompido quando se minimizou o erro junto a dados de validação (não apresentados).
- Curvas típicas de erro de treinamento e validação são apresentadas a seguir.

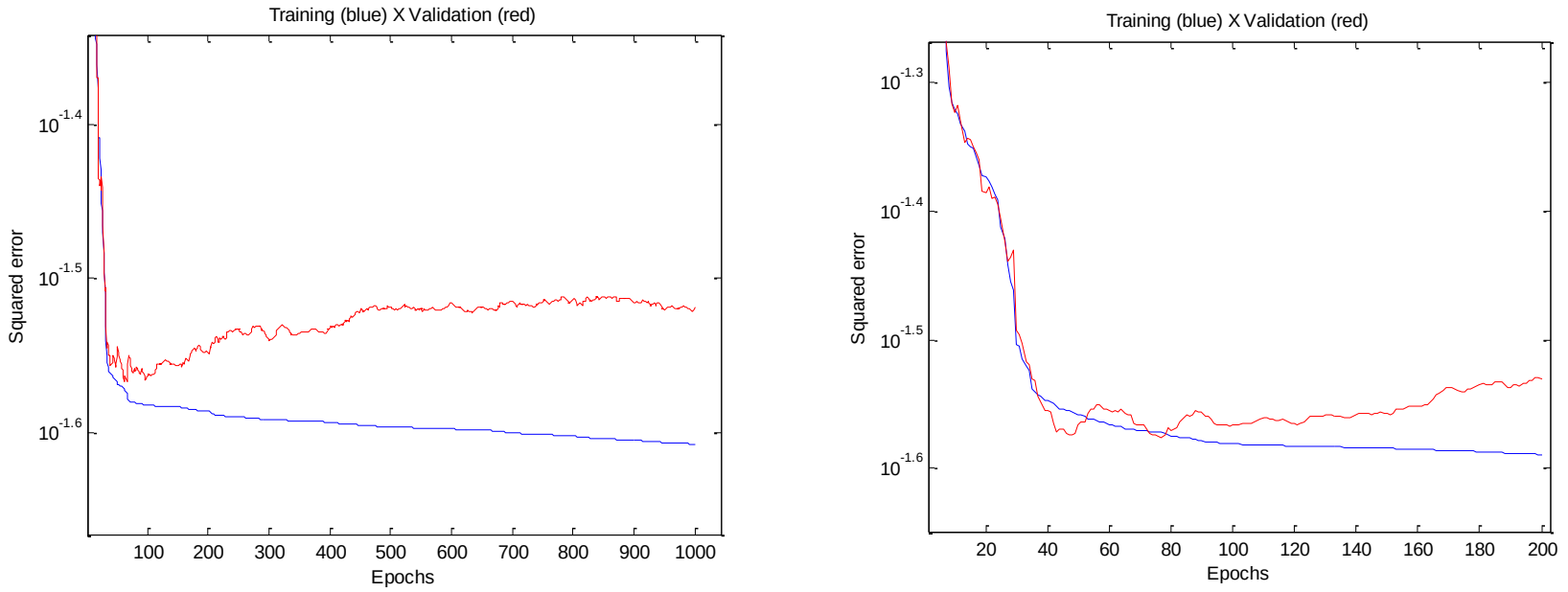


Figura 26 – Ilustrações típicas da evolução, ao longo das épocas de treinamento, dos erros de treinamento (azul) e de validação (vermelho) em treinamento supervisionado de redes MLP.

- No entanto, nem sempre o erro de validação apresenta este comportamento, e cada caso deve ser analisado isoladamente.
- Este mecanismo de regularização do treinamento de redes neurais é denominado *holdout*, por manter de fora do treinamento um subconjunto das amostras.
- Como a curva do erro de validação oscila bastante e esboça um comportamento pouco previsível, não é indicado desenvolver detectores automáticos de mínimos e encerrar o treinamento ali. O mais indicado é sobretreinar a rede e armazenar os pesos associados ao mínimo do erro de validação.
- Não existe um consenso sobre como fazer o melhor particionamento do conjunto de dados, ou seja, sobre como dividi-lo de forma que possamos encontrar uma rede com a melhor capacidade de generalização em todos os casos. Uma sugestão de partida pode ser 80% das amostras para treinamento e 20% para validação.
- No caso em que as amostras correspondem a dados rotulados em problemas de classificação de padrões, procure respeitar a distribuição junto a cada classe.

- Quando a quantidade de dados é elevada, é comum também realizar uma terceira partição, chamada de conjunto de teste. Neste caso, o objetivo do treinamento continua sendo a minimização do erro junto ao conjunto de validação, mas o desempenho final da rede neural é medido junto ao conjunto de teste.

11.1 Validação cruzada com *k* pastas

- A técnica *holdout* que acabou de ser descrita tem uma desvantagem importante: alguns dados disponíveis sempre serão usados no ajuste dos pesos e outros nunca serão usados para este fim, pois compõem o conjunto de validação.
- Uma técnica mais elaborada é dividir o conjunto de amostras disponíveis para treinamento em k pastas e realizar k treinamentos, cada um considerando k–1 pastas para o ajuste de pesos (equivale ao conjunto de treinamento da seção anterior) e 1 pasta para a validação (equivale ao conjunto de validação da seção anterior).

- Com este procedimento, toda amostra disponível vai aparecer k–1 vezes no conjunto de treinamento e 1 vez no conjunto de validação. Repare que os k conjuntos de treinamento terão composição distinta, assim como os k conjuntos de validação. O desempenho da rede neural é geralmente tomado como a média do desempenho das k redes neurais obtidas.

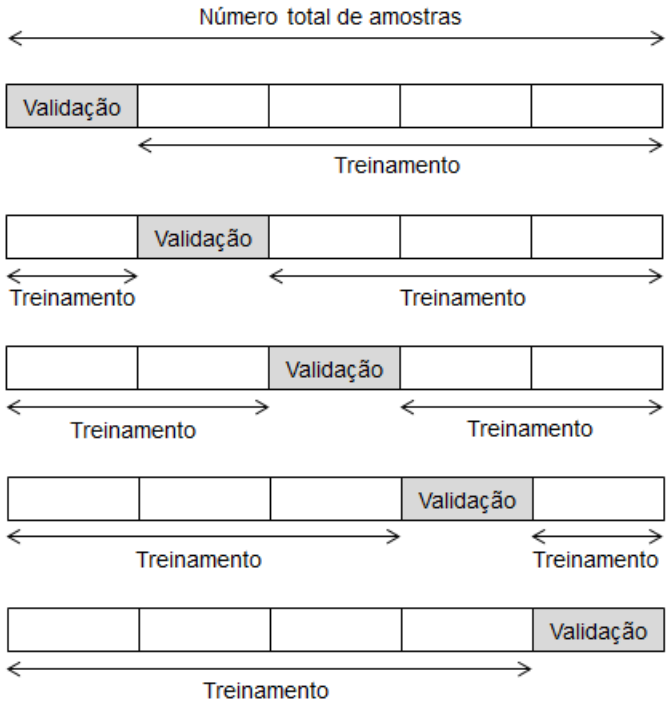


Figura 27 – Esquema de divisão das amostras disponíveis para treinamento na abordagem de validação cruzada com k-pastas (*k-fold cross-validation*), com k=5.

11.2 Rede neural MLP como um modelo instável

- Como todos os modelos que apresentam capacidade de aproximação universal, a rede neural MLP é considerada um modelo instável, no sentido de que a proposta de mapeamento de entrada-saída pode sofrer mudanças não-esperadas com a simples inclusão de uma amostra adicional junto ao conjunto de dados de treinamento.
- Essa instabilidade simplesmente é o reflexo do grau de flexibilidade e do poder de aproximação e não necessariamente se apresenta como uma desvantagem em aplicações práticas. De fato, esta propriedade é fundamental para o sucesso de abordagem de comitê de máquinas, denominada *ensemble*, e também da técnica de *dropout*, a ser vista mais adiante.
- As técnicas de regularização *holdout*, *k-fold cross-validation* e *dropout* podem ser interpretadas, então, como uma forma de "robustecer" o resultado do processo de aproximação do mapeamento de entrada-saída a partir de modelos instáveis.

11.3 Gradiente, hessiana e algoritmos de otimização

- Considere uma função contínua e diferenciável até 2a. ordem em todos os pontos do domínio de interesse, tal que: f: $\Re^n \rightarrow \Re$ e $\mathbf{x} \in \Re^n$
- Expansão em série de Taylor em torno do ponto $\mathbf{x}^* \in \Re^n$:

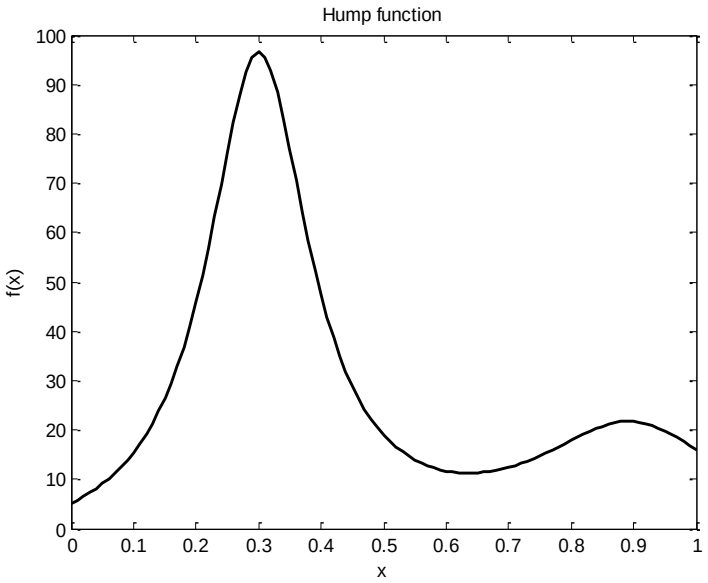
$$f(\mathbf{x}) = f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^T (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \nabla^2 f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + O(3)$$

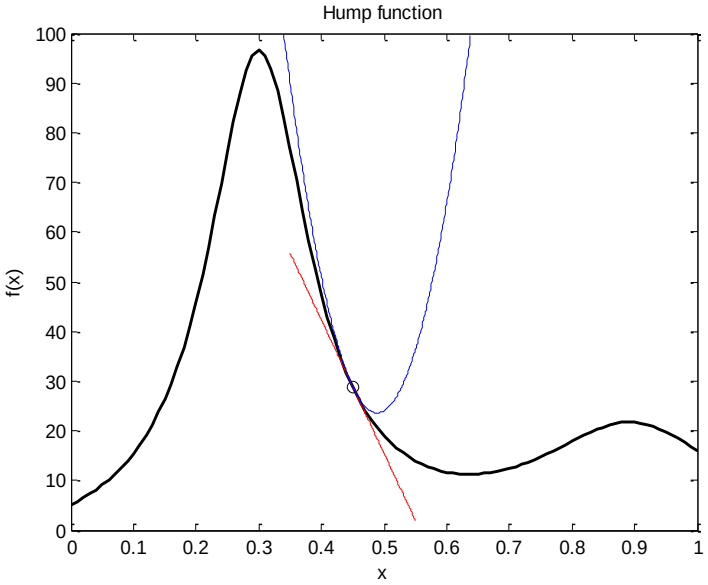
$$\nabla f(x^*) = \begin{bmatrix} \frac{\partial f(\mathbf{x}^*)}{\partial x_1} \\ \frac{\partial f(\mathbf{x}^*)}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x}^*)}{\partial x_n} \end{bmatrix} \qquad \nabla^2 f(\mathbf{x}^*) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_2^2} & & \vdots \\ & & \ddots & \\ \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_n \partial x_1} & \cdots & & \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_n^2} \end{bmatrix}$$

Vetor gradiente **Matriz hessiana**

- O algoritmo de retropopagação do erro (do inglês *backpropagation*) é empregado para obter o vetor gradiente, onde cada elemento do vetor gradiente está associado a um peso da rede neural e indica o quanto a saída é influenciada por uma variação incremental neste peso sináptico.

- Função $y = f(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$, conhecida como *hump function*, e suas aproximações de 1a. e 2a. ordem no ponto $x = 0.45$.





- Existem várias técnicas para obter exatamente ou aproximadamente a informação de 2a. ordem junto à superfície de erro produzida por redes neurais MLP (BATTITI, 1992; BISHOP, 1992).

- O processo de otimização não-linear envolvido no ajuste de pesos de uma rede neural vai realizar aproximações locais de primeira ordem ou de primeira e segunda ordem junto à superfície de erro e realizar ajustes incrementais e recursivos na forma:

$$\theta_{k+1} = \theta_k + \text{passo}_k * \text{direção}_k$$

- Parte-se de uma condição inicial θ_0 e se aplica iterativamente a fórmula acima, sendo que a direção depende da informação local de primeira e segunda ordem. Como já comentado, cada proposta de algoritmo de otimização vai diferir na forma de computar o *passo* e a *direção* de ajuste, a cada iteração *k*.
- No Tópico 8 do curso iremos retomar os conceitos de otimização não-linear irrestrita, com enfoque em *deep learning*.

11.4 Mínimos locais

- Como o processo de ajuste é iterativo e baseado apenas em informações locais, os algoritmos de otimização geralmente convergem para o mínimo local mais próximo de onde se encontra a busca, que pode representar uma solução inadequada (com nível de erro acima do aceitável).

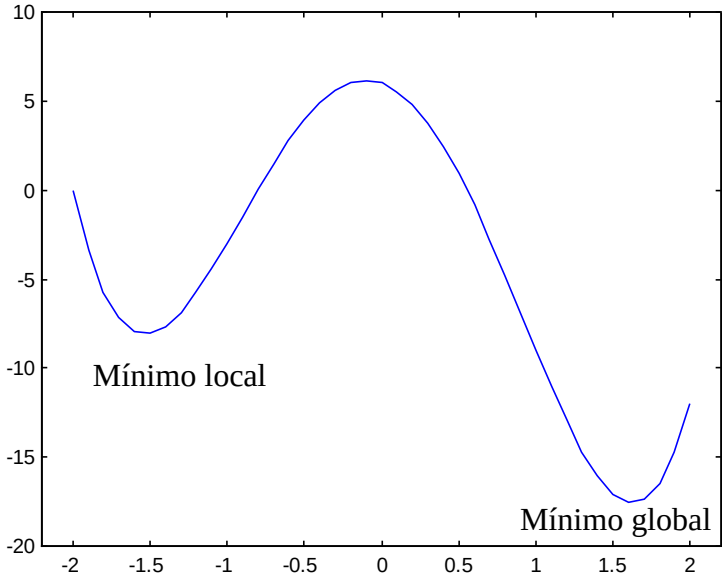


Figura 28 – Exemplo ilustrativo de mínimos local e global (considerando uma única variável).

- Note que algoritmos de 2a. ordem tendem a convergir mais rápido para os mínimos locais, mas não se pode afirmar que eles convergem para mínimos de melhor qualidade que aqueles produzidos pelos algoritmos de primeira ordem.
- Um conceito relevante aqui é o de **bacia de atração**. Todo mínimo local tem uma bacia de atração e o mínimo local a ser fornecido pelo algoritmo de otimização tende a ser aquele em cuja bacia de atração se encontra a condição inicial (ponto de partida da busca iterativa).
- Isso é bem provável no caso de buscas iterativas que dão passos sempre minimizantes, mas podem ocorrer passos capazes de deslocar a busca para bacias de atração vizinhas, associadas a outros mínimos locais.
- A trajetória da condição inicial até o ótimo local resultante será certamente distinta para algoritmos de 1a. e 2a. ordem, pois eles diferem a cada iteração da busca, mas o resultado final tende a ser o mesmo, a menos que haja deslocamentos casuais para outras bacias de atração vizinhas.

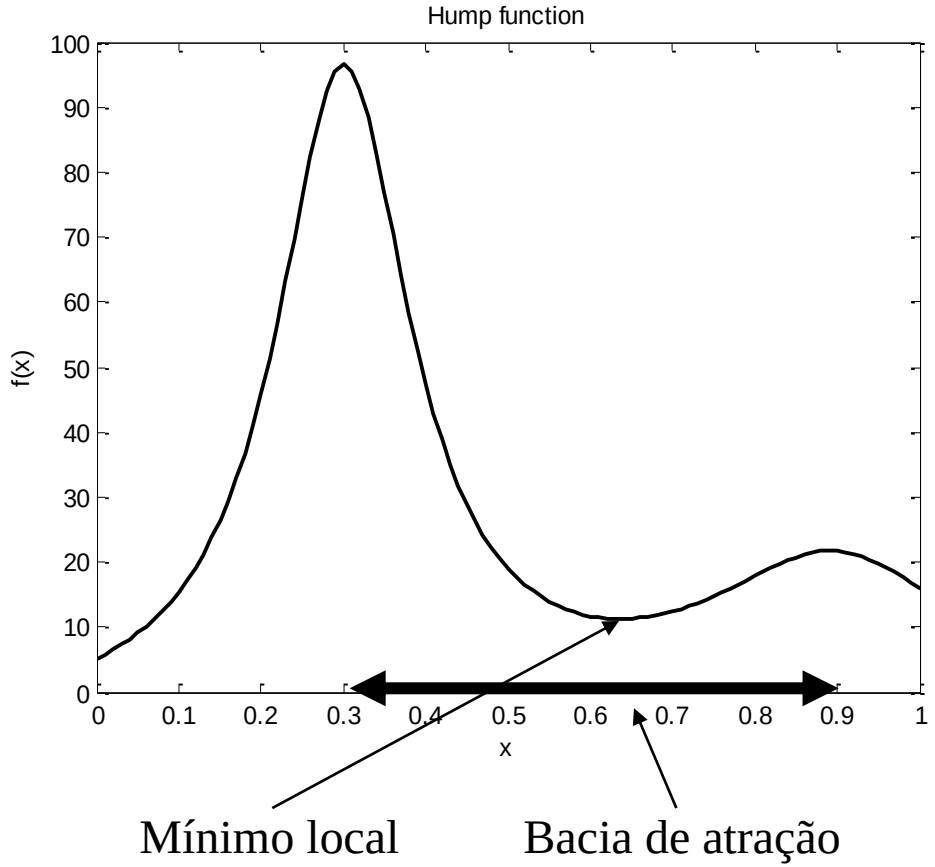


Figura 29 – Exemplo de bacia de atração de um mínimo local

11.5 Condição inicial para os pesos da rede neural

- Embora existam técnicas mais elaboradas (GLOROT & BENGIO, 2010), os pesos da rede neural podem ser inicializados com valores pequenos e aleatoriamente distribuídos em torno de zero.
- Esta inicialização promove as seguintes propriedades da rede neural inicial:
  - ✓ O mapeamento realizado pela rede neural tende a se aproximar de um hiperplano, não apresentando, assim, nenhuma tendência definida, em termos de comportamento não-linear;
  - ✓ A ativação de todos os neurônios vai se encontrar fora da região de saturação (no caso de funções sigmoidais), facilitando o processo de ajuste de pesos, a ser iniciado.
- Em termos práticos, com este procedimento de inicialização, o mapeamento produzido pela rede neural começa sem nenhuma contorção expressiva, mas com máxima capacidade de se contorcer de acordo com a demanda da aplicação.

11.6 Critério de parada

- O critério de parada mais empregado é o número máximo de épocas de treinamento, onde uma época é dada pela apresentação de todas as amostras do conjunto de treinamento à rede neural.
- Mas existem outros critérios que podem ser considerados conjuntamente ou em substituição ao número de épocas:

  ✓ Módulo do vetor gradiente abaixo de um certo limiar;

  ✓ Progresso do erro de treinamento abaixo de um certo limiar;

  ✓ Grau de ajuste dos pesos sinápticos abaixo de um certo limiar.

- Esses limiares mencionados nos três itens acima devem ser definidos pelo usuário, havendo a necessidade de realização de alguns testes junto a cada aplicação pretendida.

12. Processo Iterativo para MLP – Método Padrão-a-Padrão

Obs1: Está sendo considerada a aplicação de um método de 1a. ordem para ajuste dos pesos.

Obs2: Este método e variações dele são conhecidas como *stochastic gradient descent* (DUCHI et al. 2011).

- Defina uma condição inicial para o vetor de pesos **w** e um passo α pequeno;
- Faça $k = 0$ e calcule $J(\mathbf{w}(k))$;
- Enquanto o critério de parada não for atendido, faça:
    - ♦ Ordene aleatoriamente os N padrões de entrada-saída;
    - ♦ Para l variando de 1 até N, faça:

        Apresente o padrão l de entrada à rede;

        Calcule $J_l(\mathbf{w}(k))$ e $\nabla J_l(\mathbf{w}(k))$;

        $\mathbf{w}(k+1) = \mathbf{w}(k) - \alpha \nabla J_l(\mathbf{w}(k))$;
    - ♦ $k = k + 1$;
    - ♦ Calcule $J(\mathbf{w}(k))$;

13. Processo Iterativo para MLP – Método em Lote ou Batelada

Obs. 1: Está sendo considerada a aplicação de um método de 1a. ordem para ajuste dos pesos.

Obs. 2: Versões intermediárias entre padrão-a-padrão e batelada têm sido bastante empregadas, sendo denominadas de *mini-batch* (https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/).

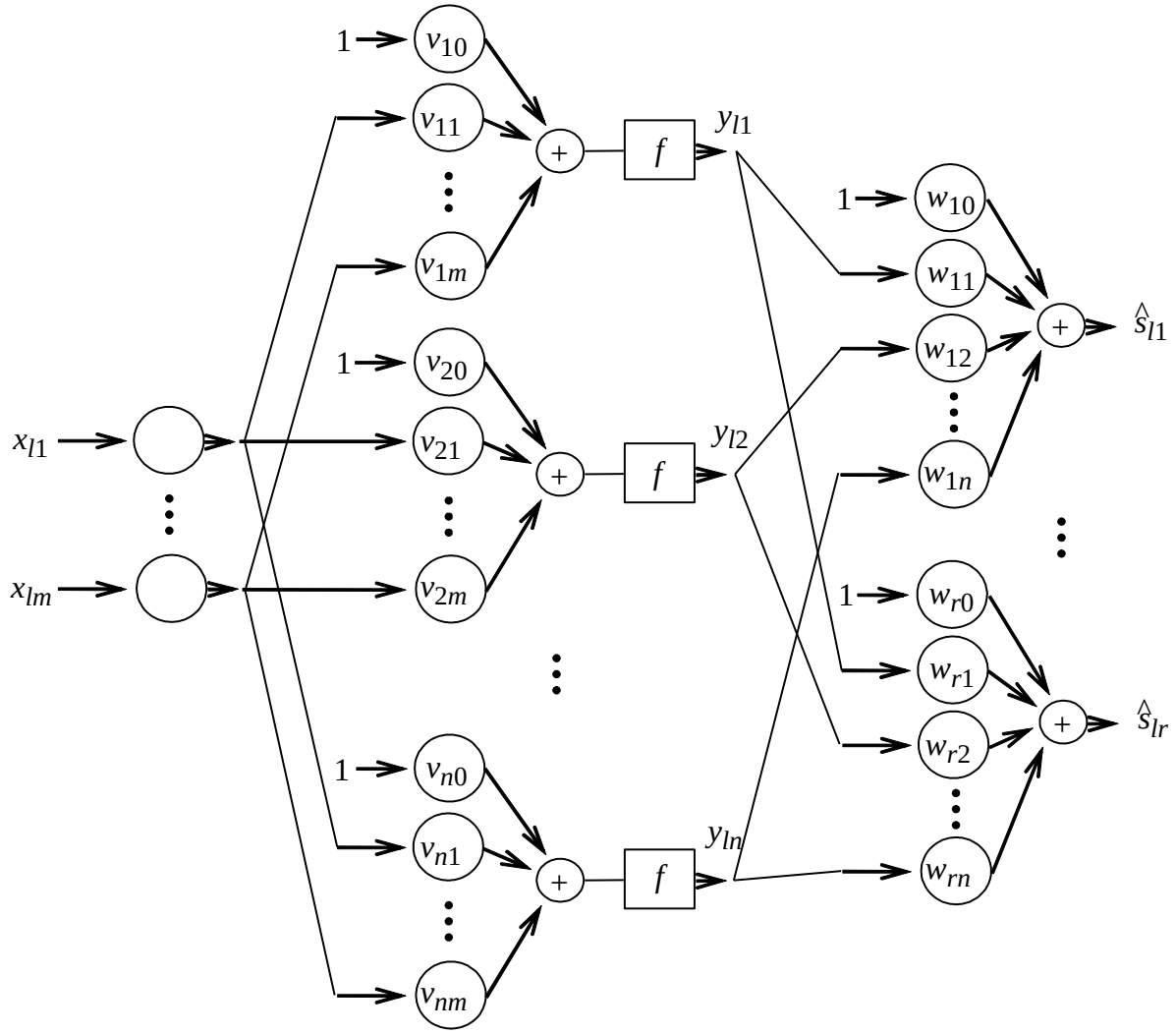
- Defina uma condição inicial para o vetor de pesos **w** e escolha um passo α pequeno;
- Faça $k = 0$ e calcule $J(\mathbf{w}(k))$;
- Enquanto o critério de parada não for atendido, faça:
    - ♦ Para l variando de 1 até N, faça:
        Apresente o padrão l de entrada à rede;
        Calcule $J_l(\mathbf{w}(k))$ e $\nabla J_l(\mathbf{w}(k))$;
    - ♦ $\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{\alpha}{N}\sum_{l=1}^{N}\nabla J_l(\mathbf{w}(k))$;
    - ♦ $k = k + 1$;
    - ♦ Calcule $J(\mathbf{w}(k))$;

14. Vetor gradiente para redes MLP

- Nesta seção, são apresentados todos os passos até a obtenção das derivadas parciais da função $J(\theta)$ em relação aos pesos v_{ji} e w_{kj}, com *i*, *j* e *k* quaisquer, na rede neural MLP apresentada a seguir, a qual tem múltiplas saídas, num total de *r* saídas.
- O vetor θ contém todos os pesos sinápticos da rede neural, ordenados de forma arbitrária, mas fixa.
- A função-objetivo a ser minimizada é dada por:

$$J(\theta) = \frac{1}{2}\sum_{l=1}^{N}\sum_{k=1}^{r}\left(s_{lk} - \hat{s}_{lk}\right)^2 = \frac{1}{2}\sum_{l=1}^{N}\sum_{k=1}^{r}\left\{s_{lk} - \left[\sum_{j=1}^{n} w_{kj}\, f\left(\sum_{i=0}^{m} v_{ji} x_{li}\right) + w_{k0}\right]\right\}^2$$

- Vamos obter $\frac{\partial J}{\partial v_{ji}}$, com j=1,…,n e i=0,…,m, e obter $\frac{\partial J}{\partial w_{kj}}$, com k=1,…,r e j=0,…,n. Está sendo considerado que x_{l0} = 1, l=1,...,N.


1
v_{10}
v_{11}
v_{1m}
y_{l1}
f
+
1
w_{10}
w_{11}
w_{12}
w_{1n}
$\hat{s}_{l1}$
x_{l1}
x_{lm}
1
v_{20}
v_{21}
v_{2m}
y_{l2}
1
w_{r0}
w_{r1}
w_{r2}
w_{rn}
$\hat{s}_{lr}$
1
v_{n0}
v_{n1}
v_{nm}
y_{ln}

Resolução:

Para evitar ambiguidade de sub-índices, os índices *i*, *j* e *k* de $J(\theta)$ serão renomeados, respectivamente, para *I*, *J* e *K*, produzindo:

$$J(\theta)=\frac{1}{2}\sum_{l=1}^{N}\sum_{K=1}^{r}\left(s_{lK}-\hat{s}_{lK}\right)^2=\frac{1}{2}\sum_{l=1}^{N}\sum_{K=1}^{r}\left\{s_{lK}-\left[\sum_{J=1}^{n}w_{KJ}\,f\left(\sum_{I=0}^{m}v_{JI}\,x_{lI}\right)+w_{K0}\right]\right\}^2$$

Três variáveis auxiliares serão consideradas:

$$e_{lK}=s_{lK}-\hat{s}_{lK}$$

$$u_{lJ}=\sum_{I=0}^{m}v_{JI}\,x_{lI}$$

$$y_{lJ}=f\left(u_{lJ}\right)$$

As propriedades de operadores de diferenciação que iremos aplicar são a regra da cadeia e equivalência entre a derivada da soma e a soma das derivadas.

Parte 1 da Solução:

$$\frac{\partial J(\theta)}{\partial v_{ji}} = \frac{\partial}{\partial v_{ji}}\left[\frac{1}{2}\sum_{l=1}^{N}\sum_{K=1}^{r}\left(s_{lK}-\hat{s}_{lK}\right)^2\right] = \frac{\partial}{\partial v_{ji}}\left[\frac{1}{2}\sum_{l=1}^{N}\sum_{K=1}^{r}e_{lK}^2\right] = \frac{1}{2}\sum_{l=1}^{N}\sum_{K=1}^{r}\frac{\partial\left(e_{lK}^2\right)}{\partial v_{ji}} =$$

$$= \frac{1}{2}\sum_{l=1}^{N}\sum_{K=1}^{r}\frac{\partial\left(e_{lK}^2\right)}{\partial e_{lK}}\frac{\partial\left(e_{lK}\right)}{\partial v_{ji}} = \sum_{l=1}^{N}\sum_{K=1}^{r}e_{lK}\frac{\partial\left(e_{lK}\right)}{\partial \hat{s}_{lK}}\frac{\partial\left(\hat{s}_{lK}\right)}{\partial v_{ji}} = -\sum_{l=1}^{N}\sum_{K=1}^{r}e_{lK}\frac{\partial\left(\hat{s}_{lK}\right)}{\partial v_{ji}}$$

Sabendo que $\hat{s}_{lK} = \sum_{J=1}^{n} w_{KJ} f\left(\sum_{I=0}^{m} v_{JI} x_{lI}\right) + w_{K0}$, então tem-se:

$$\frac{\partial\left(\hat{s}_{lK}\right)}{\partial v_{ji}} = \frac{\partial}{\partial v_{ji}}\left[\sum_{J=1}^{n} w_{KJ} f\left(\sum_{I=0}^{m} v_{JI} x_{lI}\right) + w_{K0}\right] = \sum_{J=1}^{n}\frac{\partial}{\partial v_{ji}}\left[w_{KJ} f\left(\sum_{I=0}^{m} v_{JI} x_{lI}\right) + w_{K0}\right] =$$

$$= \frac{\partial}{\partial v_{ji}}\left[w_{Kj} f\left(\sum_{I=0}^{m} v_{jI} x_{lI}\right)\right] = \frac{\partial}{\partial y_{lj}}\left[w_{Kj} f\left(\sum_{I=0}^{m} v_{jI} x_{lI}\right)\right]\frac{\partial y_{lj}}{\partial v_{ji}} = w_{Kj}\frac{\partial}{\partial v_{ji}}\left[f\left(u_{lj}\right)\right] =$$

$$= w_{Kj}\frac{\partial}{\partial u_{lj}}\left[f\left(u_{lj}\right)\right]\frac{\partial u_{lj}}{\partial v_{ji}} = w_{Kj} f'\left(u_{lj}\right)\frac{\partial}{\partial v_{ji}}\left(\sum_{I=0}^{m} v_{jI} x_{lI}\right) = w_{Kj} f'\left(u_{lj}\right)\sum_{I=0}^{m}\frac{\partial}{\partial v_{ji}}\left(v_{jI} x_{lI}\right) =$$

$$= w_{Kj} f'\left(u_{lj}\right) \frac{\partial}{\partial v_{ji}} \left(v_{ji} x_{li}\right) = w_{Kj} f'\left(u_{lj}\right) x_{li}$$

$$\frac{\partial J(\theta)}{\partial v_{ji}} = -\sum_{l=1}^{N} \sum_{K=1}^{r} e_{lK} w_{Kj} f'\left(u_{lj}\right) x_{li} \Rightarrow \boxed{\frac{\partial J(\theta)}{\partial v_{ji}} = -\sum_{l=1}^{N} \sum_{K=1}^{r} \left(s_{lK} - \hat{s}_{lK}\right) w_{Kj} f'\left(u_{lj}\right) x_{li}}$$

Parte 2 da Solução:

$$\frac{\partial J(\theta)}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}} \left[\frac{1}{2} \sum_{l=1}^{N} \sum_{K=1}^{r} \left(s_{lK} - \hat{s}_{lK}\right)^2\right] = \frac{\partial}{\partial w_{kj}} \left[\frac{1}{2} \sum_{l=1}^{N} \sum_{K=1}^{r} e_{lK}^2\right] = \frac{1}{2} \sum_{l=1}^{N} \sum_{K=1}^{r} \frac{\partial\left(e_{lK}^2\right)}{\partial w_{kj}} =$$

$$= \frac{1}{2} \sum_{l=1}^{N} \frac{\partial\left(e_{lk}^2\right)}{\partial w_{kj}} = \frac{1}{2} \sum_{l=1}^{N} \frac{\partial\left(e_{lk}^2\right)}{\partial e_{lk}} \frac{\partial\left(e_{lk}\right)}{\partial w_{kj}} = \sum_{l=1}^{N} e_{lk} \frac{\partial\left(e_{lk}\right)}{\partial w_{kj}} = \sum_{l=1}^{N} e_{lk} \frac{\partial\left(e_{lk}\right)}{\partial \hat{s}_{lk}} \frac{\partial\left(\hat{s}_{lk}\right)}{\partial w_{kj}} = -\sum_{l=1}^{N} e_{lk} \frac{\partial\left(\hat{s}_{lk}\right)}{\partial w_{kj}}$$

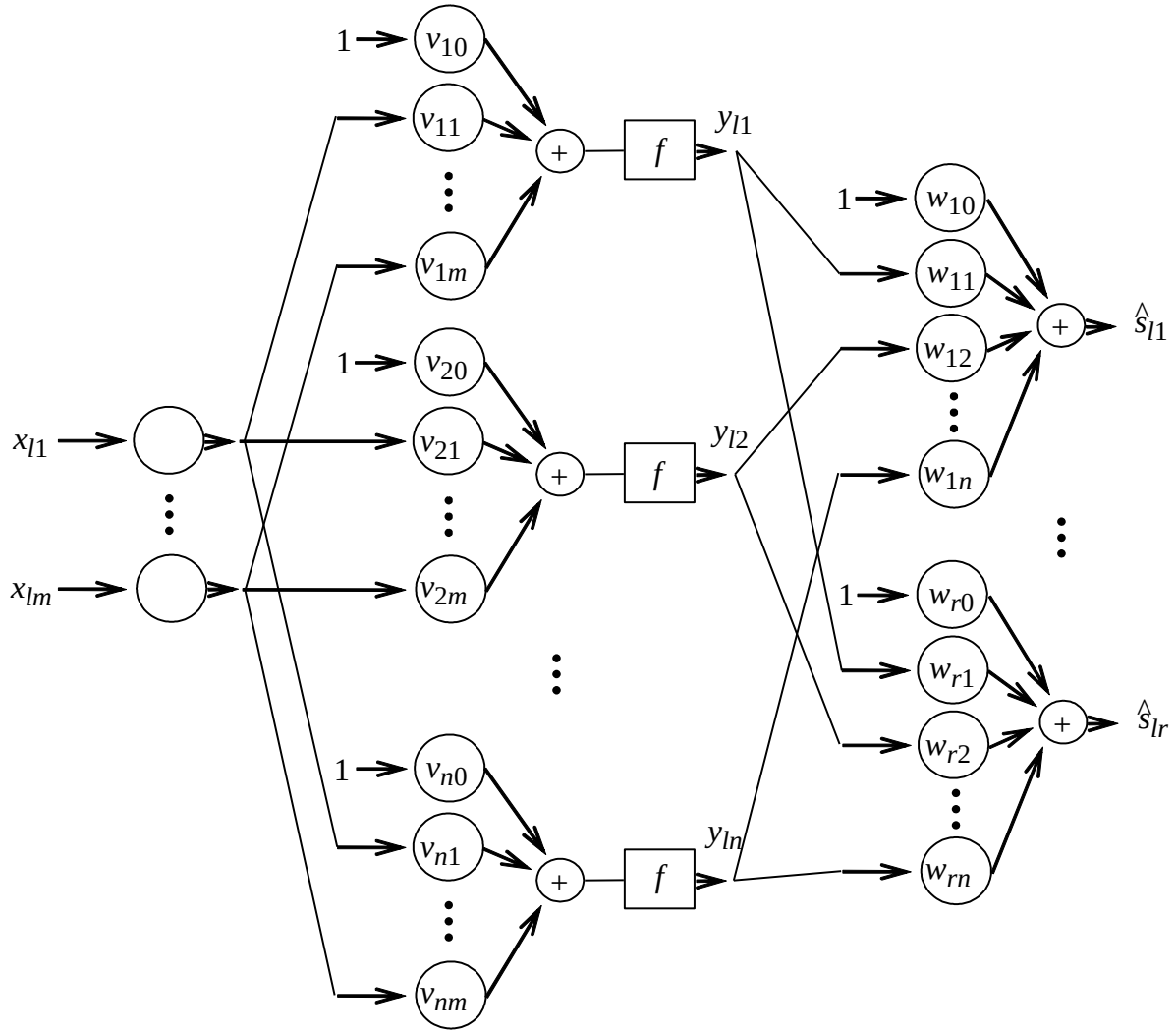
Sabendo que $\hat{s}_{lk} = \sum_{J=1}^{n} w_{kJ} f\left(\sum_{I=0}^{m} v_{JI} x_{lI}\right) + w_{k0}$, então tem-se:

$$\frac{\partial\left(\hat{s}_{lk}\right)}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}} \left[\sum_{J=1}^{n} w_{kJ} f\left(\sum_{I=0}^{m} v_{JI} x_{lI}\right) + w_{k0}\right] = \sum_{J=1}^{n} \frac{\partial}{\partial w_{kj}} \left[w_{kJ} f\left(\sum_{I=0}^{m} v_{JI} x_{lI}\right) + w_{k0}\right] =$$

$$= \begin{cases} \dfrac{\partial}{\partial w_{kj}} \left[w_{kj} f\left(\sum_{I=0}^{m} v_{jI} x_{lI} \right) \right] & \text{se } j \neq 0 \\ \dfrac{\partial}{\partial w_{kj}} \left[w_{k0} \right] & \text{se } j = 0 \end{cases} = \begin{cases} f\left(\sum_{I=0}^{m} v_{jI} x_{lI} \right) & \text{se } j \neq 0 \\ 1 & \text{se } j = 0 \end{cases}$$

$$\frac{\partial J(\theta)}{\partial w_{kj}} = -\sum_{l=1}^{N} e_{lk} \frac{\partial(\hat{s}_{lk})}{\partial w_{kj}} = \begin{cases} -\sum_{l=1}^{N} e_{lk} f\left(\sum_{I=0}^{m} v_{jI} x_{lI} \right) & \text{se } j \neq 0 \\ -\sum_{l=1}^{N} e_{lk} & \text{se } j = 0 \end{cases}$$

$$\boxed{\frac{\partial J(\theta)}{\partial w_{kj}} = \begin{cases} -\sum_{l=1}^{N} (s_{lk} - \hat{s}_{lk}) f\left(\sum_{I=0}^{m} v_{jI} x_{lI} \right) & \text{se } j \neq 0 \\ -\sum_{l=1}^{N} (s_{lk} - \hat{s}_{lk}) & \text{se } j = 0 \end{cases}}$$


1
v_{10}
v_{11}
v_{1m}
y_{l1}
f
+
1
w_{10}
w_{11}
w_{12}
w_{1n}
$\hat{s}_{l1}$
x_{l1}
x_{lm}
1
v_{20}
v_{21}
v_{2m}
y_{l2}
1
w_{r0}
w_{r1}
w_{r2}
w_{rn}
$\hat{s}_{lr}$
1
v_{n0}
v_{n1}
v_{nm}
y_{ln}

15. Máquinas de aprendizado extremo (ELMs)

- Todas as propostas de redes neurais não-recorrentes (*feedfoward*) com uma camada intermediária, como o perceptron de múltiplas camadas (MLP) e a rede neural com função de ativação de base radial (RBF) (veja Seção 17), produzem a sua saída (podendo ser múltiplas saídas) como uma combinação linear das ativações dos neurônios da camada anterior.
- Tomando uma única camada intermediária, pode-se afirmar, portanto, que redes neurais MLP e RBF sintetizam mapeamentos multidimensionais de entrada-saída por meio de uma composição aditiva de funções-base, na forma:

$$\hat{s}_{kl} = \sum_{j=1}^{n} w_{kj} f\left(\mathbf{v}_j, b_j, \mathbf{x}_l\right) + w_{k0}$$

onde

  - $\hat{s}_{kl}$ é a k-ésima saída da rede neural para o l-ésimo padrão de entrada $\mathbf{x}_l$;
  - $f\left(\mathbf{v}_j, b_j, \bullet\right)$ é a j-ésima função da base de funções-base.

- No caso da rede neural MLP, as funções-base são funções de expansão ortogonal (*ridge functions*), enquanto no caso da rede neural RBF as funções-base têm um comportamento radial em relação a um centro de ativação máxima ou mínima.
- Nos dois casos, como em outros casos de composição aditiva de funções-base, há demonstração teórica da capacidade de aproximação universal. A capacidade de aproximação universal é uma **propriedade existencial**. Ela afirma que existe um número *n* finito de neurônios e uma certa configuração de pesos sinápticos que permitem obter um erro de aproximação arbitrariamente baixo para os dados de treinamento, supondo que se considera uma região compacta do espaço de entrada e que o mapeamento original, que é amostrado para produzir os dados de treinamento, é contínuo.
- É intuitivo concluir, também, que quanto maior o número *n* de neurônios na camada intermediária, maior é a flexibilidade do modelo matemático resultante, ou seja, maiores são as "possibilidades de contorção" do mapeamento a ser

sintetizado. Ao espaço de hipóteses, é associado o número *n* de neurônios da camada intermediária.

- Por outro lado, é sabido também que há o risco de sobreajuste aos dados, produzindo modelos que generalizam mal frente a novos dados de entrada-saída.
- A máxima capacidade de generalização está associada a modelos otimamente regularizados, ou seja, que se contorcem na medida certa (exibem grau adequado de flexibilidade), de acordo com as demandas de cada aplicação.
- Com isso, uma definição adequada do número de neurônios e dos pesos sinápticos é fundamental para garantir uma boa capacidade de generalização.
- Um resultado fundamental da literatura, restrito a problemas de classificação de padrões, foi apresentado por BARTLETT (1997; 1998). Nesses trabalhos, como o próprio título deles indica, conclui-se que controlar a norma dos pesos sinápticos é mais relevante para a capacidade de generalização do que controlar o tamanho da rede neural, ou seja, o número *n* de neurônios na camada intermediária.

- De fato, pode-se introduzir o conceito de ⟨número efetivo de neurônios na camada intermediária⟩, o qual é determinado pela configuração dos pesos da camada de saída da rede neural.
- As máquinas de aprendizado extremo exploram este resultado "de forma extrema", ou seja, jogam toda a responsabilidade por garantir uma boa capacidade de generalização aos pesos da camada de saída, permitindo que os pesos da camada intermediária, responsáveis por definir as funções-base, sejam definidos de modo aleatório.
- Por serem definidos de modo aleatório, portanto desvinculados das demandas da aplicação, deve-se considerar um valor elevado para n, podendo inclusive ultrapassar o valor de N, que representa o número de amostras para treinamento.
- Por mais que pareça estranho trabalhar com valores de n elevados e até maiores que N, as máquinas de aprendizado extremo se sustentam em três argumentos muito poderosos:

- ✓ O problema de treinamento passa a ser linear nos parâmetros ajustáveis, o que representa uma enorme economia de recursos computacionais para se realizar o treinamento supervisionado;
- ✓ A capacidade de generalização pode ser maximizada ao se controlar a norma dos pesos na camada de saída, não dependendo de forma significativa do número *n* de neurônios na camada intermediária;
- ✓ Há recursos computacionais disponíveis para implementar redes neurais sobredimensionadas.

- Como as funções-base podem ser definidas aleatoriamente, então não há razão para que elas tenham formas sigmoidais ou base radial. Logo, o elenco de funções-base pode ser arbitrário, embora as demonstrações de capacidade de aproximação universal para ELMs restrinjam ainda as alternativas de funções-base.
- Para mais informações sobre ELM, consultar os trabalhos de Huang e co-autores, listados na Seção 18.

15.1 Treinamento das ELMs

- Treinar uma máquina de aprendizado extremo é equivalente a resolver o seguinte problema de otimização para cada uma das saídas da rede neural:

$$\mathbf{w}_k^* = \arg \min_{\mathbf{w}_k \in \Re^{n+1}} J(\mathbf{w}_k) + C_k \times \|\mathbf{w}_k\|_2^2$$

onde

- ✓ k é o índice da saída;
- ✓ n é o número de neurônios na camada intermediária;
- ✓ $\|\cdot\|_2$ é a norma euclidiana;
- ✓ C_k é um coeficiente de regularização, a ser determinado, por exemplo, por métodos de busca unidimensional;
- ✓ $J(\mathbf{w}_k) = \frac{1}{2}\sum_{l=1}^{N}\left[\sum_{j=1}^{n} w_{kj} f(\mathbf{v}_j, b_j, \mathbf{x}_l) + w_{k0} - s_{kl}\right]^2$;
- ✓ N é o número de amostras disponíveis para treinamento.

- Este problema de otimização é conhecido na literatura como *ridge regression* (HASTIE, TIBSHIRANI & FRIEDMAN, 2009; HOERL & KENNARD, 1970).

15.2 Como encontrar os pesos sinápticos

- Uma vez fornecido o coeficiente de regularização C_k, para a k-ésima saída da rede neural, o vetor de pesos sinápticos é obtido como segue:

  1. Monta-se a matriz H_{inicial} de dimensão $N \times n$, com as ativações de todos os neurônios para todos os padrões de entrada, produzindo:

$$H_{\text{inicial}} = \begin{bmatrix} f(\mathbf{v}_1, b_1, \mathbf{x}_1) & f(\mathbf{v}_2, b_2, \mathbf{x}_1) & \cdots & f(\mathbf{v}_n, b_n, \mathbf{x}_1) \\ f(\mathbf{v}_1, b_1, \mathbf{x}_2) & \ddots & & \vdots \\ \vdots & & & \\ f(\mathbf{v}_1, b_1, \mathbf{x}_N) & \cdots & & f(\mathbf{v}_n, b_n, \mathbf{x}_N) \end{bmatrix}$$

  2. Acrescenta-se uma coluna de 1's à matriz H_{inicial}, produzindo a matriz H:

$$H = \begin{bmatrix} 1 & f(\mathbf{v}_1, b_1, \mathbf{x}_1) & f(\mathbf{v}_2, b_2, \mathbf{x}_1) & \cdots & f(\mathbf{v}_n, b_n, \mathbf{x}_1) \\ 1 & f(\mathbf{v}_1, b_1, \mathbf{x}_2) & \ddots & & \vdots \\ \vdots & \vdots & & & \\ 1 & f(\mathbf{v}_1, b_1, \mathbf{x}_N) & \cdots & & f(\mathbf{v}_n, b_n, \mathbf{x}_N) \end{bmatrix}$$

- A norma dos vetores de pesos da camada intermediária é um parâmetro sensível.

3. Monta-se o vetor $\mathbf{s}_k$, contendo todos os padrões de saída, na forma:

$$\mathbf{s}_k = \begin{bmatrix} s_{k1} & s_{k2} & \cdots & s_{kN} \end{bmatrix}^T$$

4. O vetor w_k é obtido como segue (se a matriz H não tiver posto completo, deve-se ter $C_k > 0$):

4.1. Se $n \leq N$, $\mathbf{w}_k = \left(H^T H + C_k I\right)^{-1} H^T \mathbf{s}_k$;

4.2. Se $n > N$, $\mathbf{w}_k = H^T \left(HH^T + C_k I\right)^{-1} \mathbf{s}_k$.

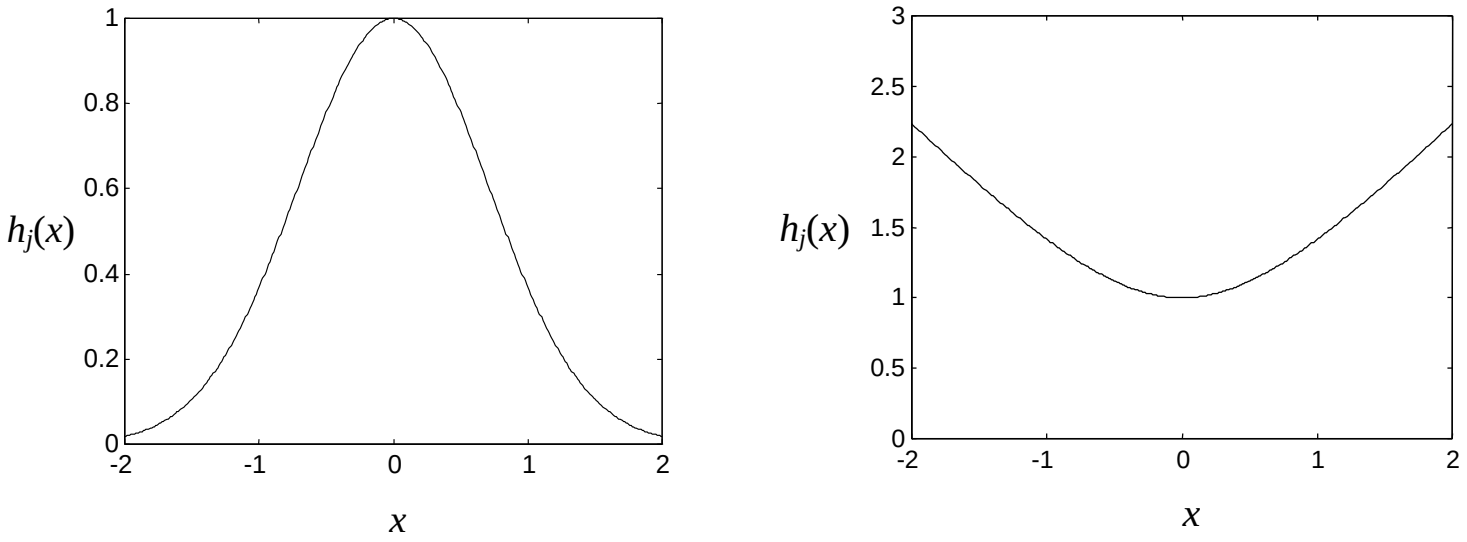
15.3 Como encontrar o coeficiente de regularização

- A maximização da capacidade de generalização requer a definição de um valor adequado para o coeficiente de regularização C_k, associado à saída *k*.
- Sugere-se aqui o uso de uma busca unidimensional (por exemplo, via seção áurea), empregando um conjunto de validação. O valor "ótimo" de C_k é aquele que minimiza o erro junto ao conjunto de validação.

16. Função de Base Radial

- Uma função de ativação de base radial é caracterizada por apresentar uma resposta que decresce (ou cresce) monotonicamente com a distância a um ponto central.
- O centro e a taxa de decrescimento (ou crescimento) em cada direção são alguns dos parâmetros a serem definidos. Estes parâmetros devem ser constantes caso o modelo de regressão seja tomado como linear nos parâmetros ajustáveis.
- Uma função de base radial monotonicamente decrescente típica é a função gaussiana, dada na forma:
  - $h_j(x) = \exp\left(-\frac{\left(x - c_j\right)^2}{r_j^2}\right)$, para o caso escalar (veja Figura 30(a));
- Uma função de base radial monotonicamente crescente típica é a função multiquádrica dada na forma:

- $h_j(x) = \dfrac{\sqrt{r_j^2 + \left(x - c_j\right)^2}}{r_j}$, para o caso escalar (veja Figura 30(b));



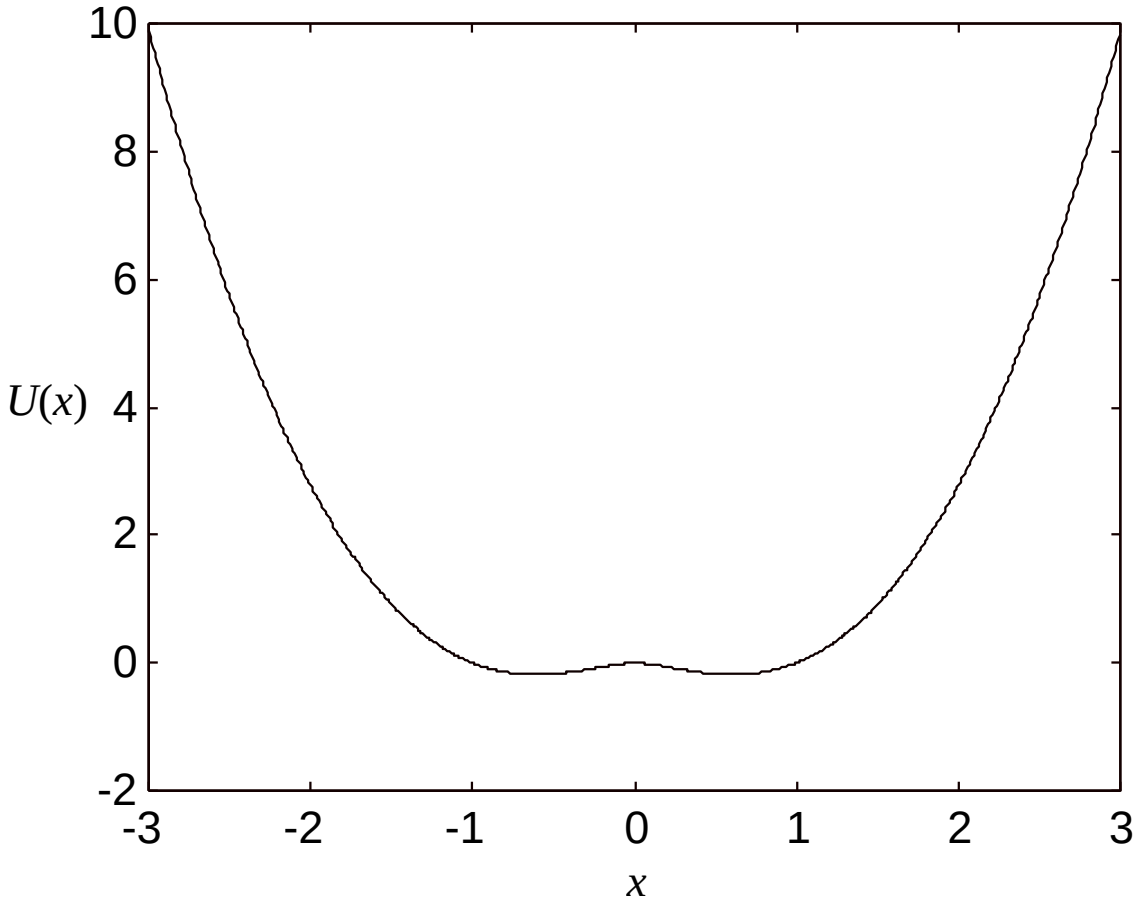
(a) (b)

Figura 30 – Exemplos de funções de base radial monovariáveis, com $c_j = 0$ e $r_j = 1$.

- Também apresenta propriedades interessantes a função *thin plate spline*. Considerando uma e duas variáveis, ela assume a forma:

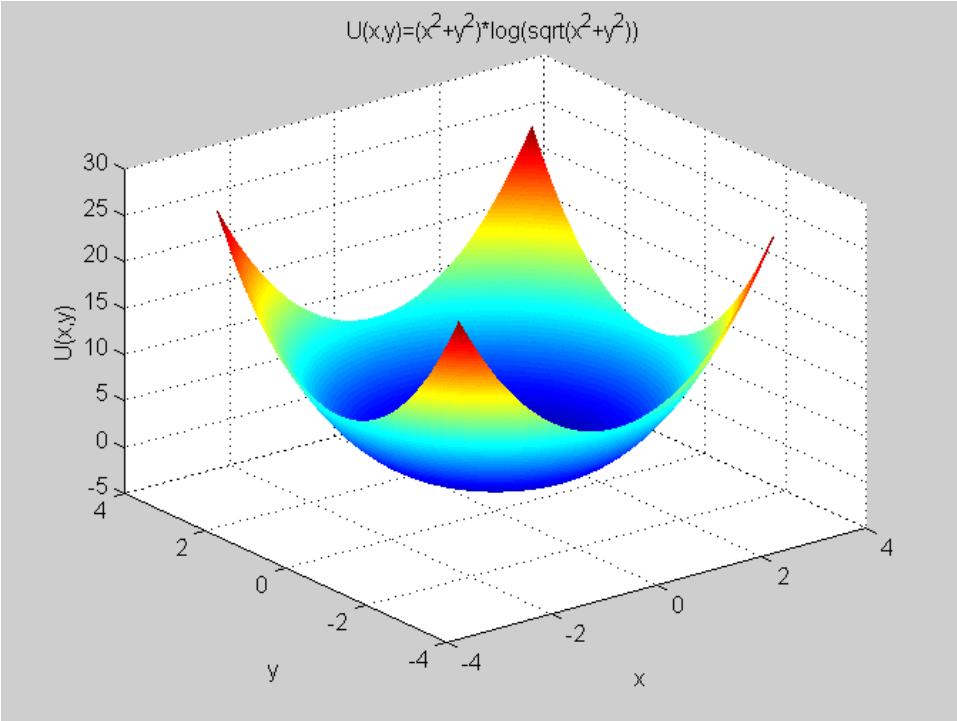
$$h_j(x) = U(x) = (x - c_j)^2 \log(|x - c_j|)$$

$$U(x) = r^2 \log(r) = x^2 \log(|x|)$$



Thin plate spline considerando $c_j = 0$.

$$h_j(x,y) = \left((x-x_j)^2 + (y-y_j)^2\right)\log\left(\sqrt{(x-x_j)^2 + (y-y_j)^2}\right)$$



Thin plate spline considerando $(x_j, y_j) = (0,0)$.

- No caso multidimensional e tomando a função gaussiana, $h_j(\mathbf{x})$ assume a forma:

$$h_j(\mathbf{x}) = \exp\left(-\left(\mathbf{x}-\mathbf{c}_j\right)^T \Sigma_j^{-1}\left(\mathbf{x}-\mathbf{c}_j\right)\right) \qquad (1)$$

onde $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$ é o vetor de entradas, $\mathbf{c}_j = \begin{bmatrix} c_{j1} & c_{j2} & \cdots & c_{jn} \end{bmatrix}^T$ é o vetor que define o centro da função de base radial e a matriz Σ_j é definida positiva e diagonal, dada por:

$$\Sigma_j = \begin{bmatrix} \sigma_{j1} & 0 & \cdots & 0 \\ 0 & \sigma_{j2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{jn} \end{bmatrix},$$

de modo que $h_j(\mathbf{x})$ pode ser expandida na forma:

$$h_j(\mathbf{x}) = \exp\left(-\frac{\left(x_1-c_{j1}\right)^2}{\sigma_{j1}} - \frac{\left(x_2-c_{j2}\right)^2}{\sigma_{j2}} - \ldots - \frac{\left(x_n-c_{jn}\right)^2}{\sigma_{jn}}\right). \qquad (2)$$

- Neste caso, os elementos do vetor $\sigma_j = \begin{bmatrix} \sigma_{j1} & \sigma_{j2} & \cdots & \sigma_{jn} \end{bmatrix}^T$ são responsáveis pela taxa de decrescimento da gaussiana junto a cada coordenada do espaço de entrada, e o argumento da função exponencial é uma norma ponderada da diferença entre o vetor de entrada e o centro da função de base radial.
- A matriz Σ_j pode ser não-diagonal, mas ainda simétrica e definida positiva. No entanto, por envolver muitos parâmetros, geralmente não é utilizada nessa forma.

17. Rede Neural RBF (*Radial Basis Function Neural Network*)

- As funções de base radial (são funções não-lineares) podem ser utilizadas como funções-base em qualquer tipo de modelo de regressão não-linear (linear ou não-linear nos parâmetros) e, particularmente, como função de ativação de qualquer tipo de rede multicamada.
- O fato de o modelo de regressão resultante ser linear ou não-linear nos parâmetros se deve à possibilidade ou não de se ajustar os centros e as dispersões das funções.

- Se apenas os pesos da camada de saída formarem o conjunto de parâmetros ajustáveis, então a rede neural é linear nos parâmetros. Caso contrário, ou seja, quando os centros $\mathbf{c}_j$ e as matrizes Σ_j, $j = 1,...,n$, também são ajustáveis, a rede neural é não-linear nos parâmetros, admitindo o próprio algoritmo de retro-propagação do erro para o processo de ajuste via treinamento supervisionado, como feito no caso do perceptron multicamadas, embora aqui os mínimos locais tenham uma influência bem maior e se sugere evitar este mecanismo de ajuste.
- A arquitetura da rede é apresentada na Figura 31, para o caso de uma única saída, resultando no seguinte mapeamento de entrada-saída:

$$y = \sum_{j=1}^{m} w_j h_j(\mathbf{x})$$

- Caso $\mathbf{c}_j$ e Σ_j, $j = 1,...,n$, sejam ajustáveis, a saída assume a forma:

$$y = \sum_{j=1}^{m} w_j h_j(\mathbf{c}_j, \Sigma_j, \mathbf{x}).$$

- Substituindo as formas compactas e expandidas de $h_j(\mathbf{x})$, dadas respectivamente pelas equações (1) e (2), resultam:

$$y = \sum_{j=1}^{m} w_j \exp\left(-\left(\mathbf{x}-\mathbf{c}_j\right)^T \Sigma_j^{-1}\left(\mathbf{x}-\mathbf{c}_j\right)\right)$$

e

$$y = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left(x_1-c_{j1}\right)^2}{\sigma_{j1}} - \frac{\left(x_2-c_{j2}\right)^2}{\sigma_{j2}} - \ldots - \frac{\left(x_n-c_{jn}\right)^2}{\sigma_{jn}}\right)$$

- Uma versão para múltiplas saídas é apresentada na Figura 32.
- Para mais detalhes sobre redes neurais RBF, favor consultar BUHMANN (2003).

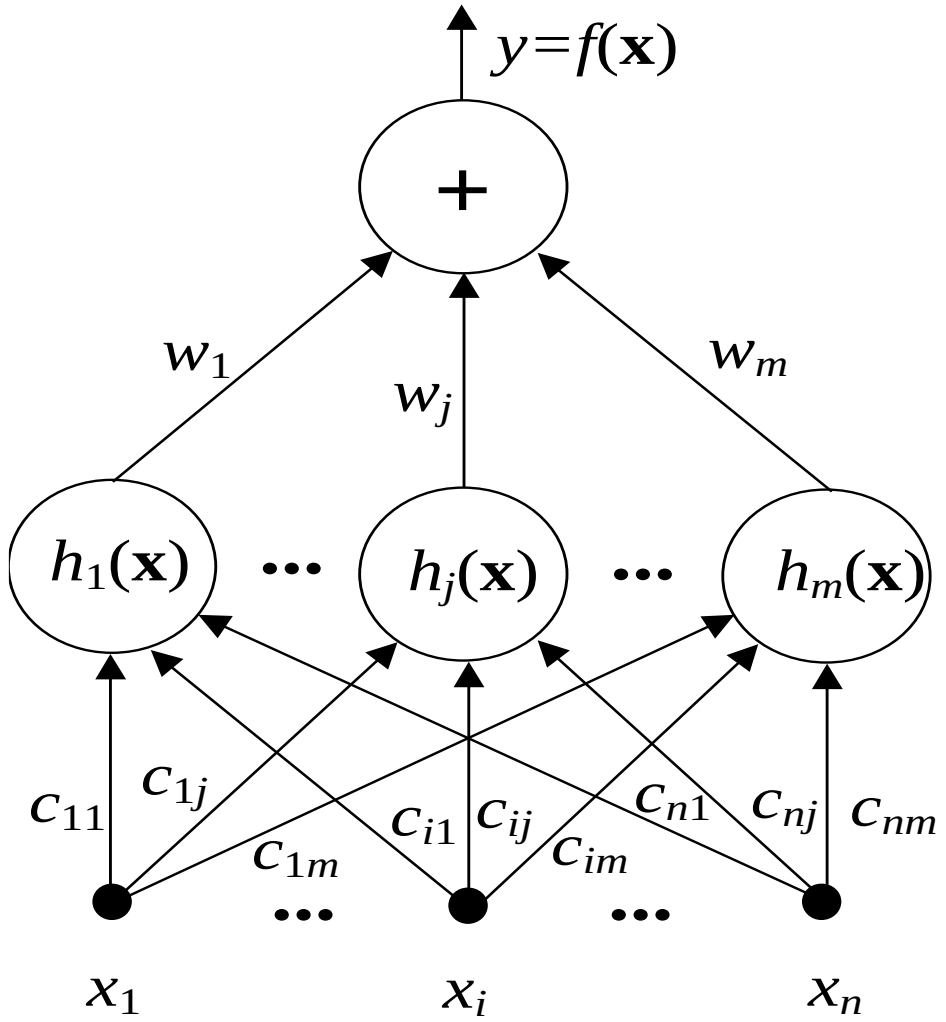


Figura 31 – Rede neural de base radial (BROOMHEAD & LOWE, 1988).

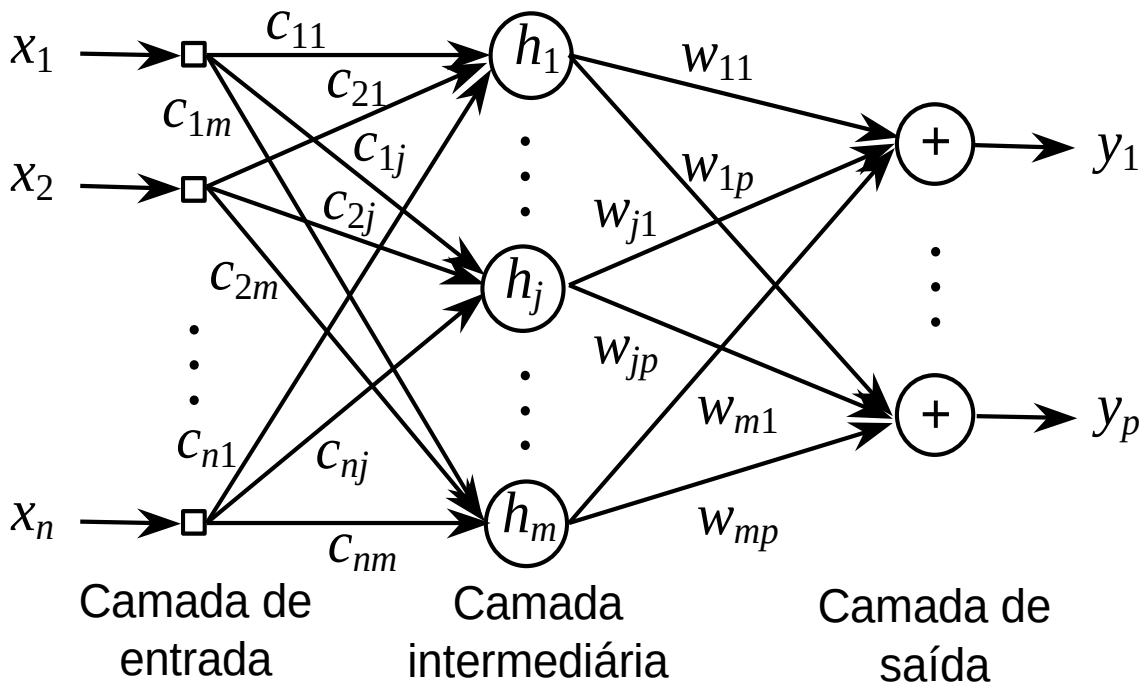
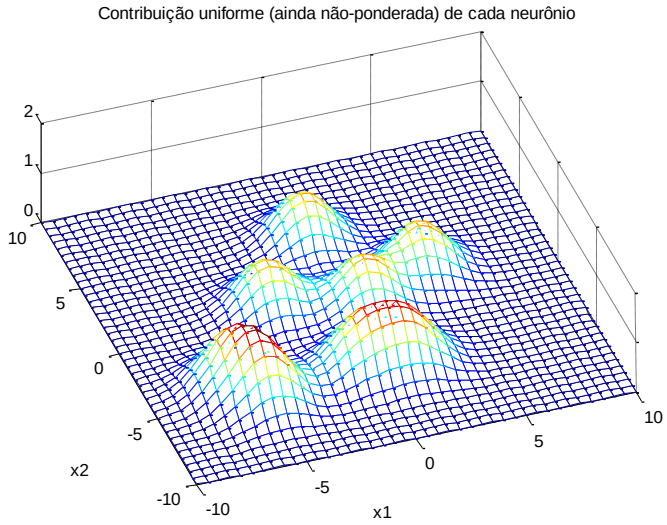


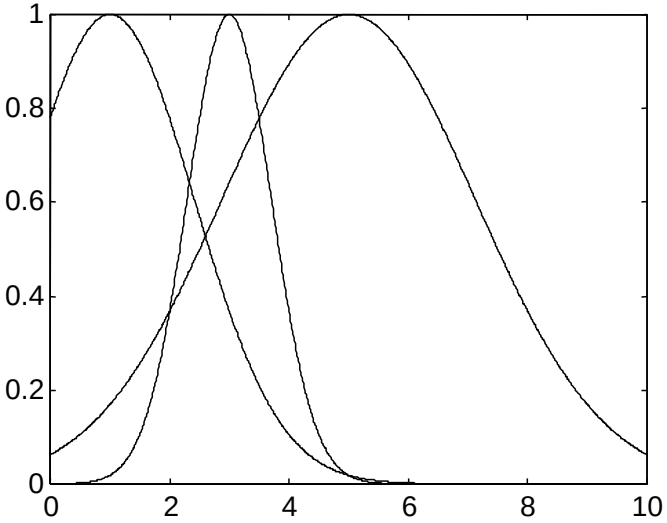
Figura 32 – Rede neural de base radial com múltiplas saídas

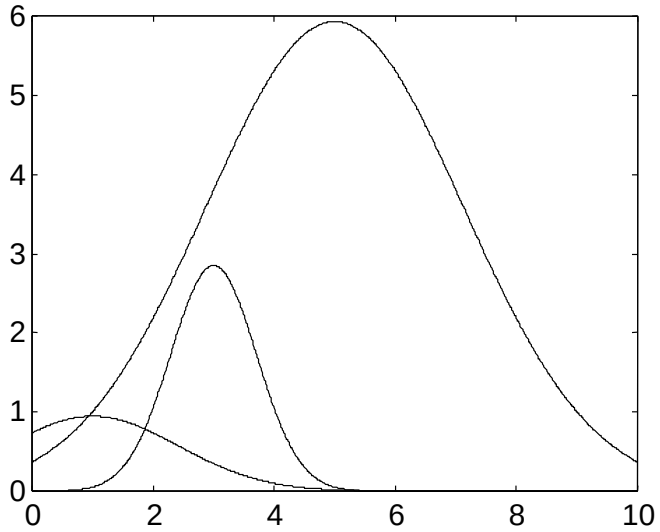
- Ao invés da ativação interna de cada neurônio da camada intermediária se dar pelo emprego do produto escalar (produto interno) entre o vetor de entradas e o vetor de pesos, como no caso do perceptron, ela é obtida a partir de uma norma ponderada da diferença entre ambos os vetores.

- Uma vez definidos centros e dispersões, o problema de ajuste dos pesos da camada de saída de uma rede neural RBF é o mesmo daquele apresentado na Seção 15.2, para o caso da ELM.
- Propostas para a definição de centros e dispersões serão apresentadas nas Seções 17.2 e 17.3.
- Em termos práticos, uma vez dispondo de funções de base radial, como combiná-las linearmente para se obter um mapeamento desejado?



17.1 Exemplo didático de aproximação usando uma rede neural RBF



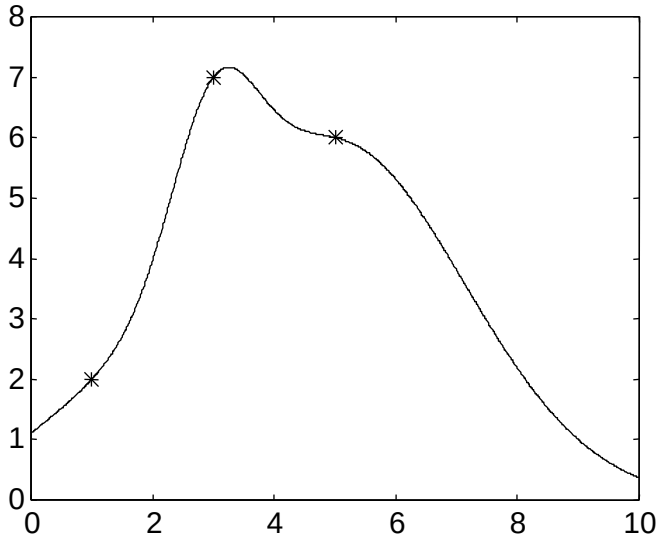


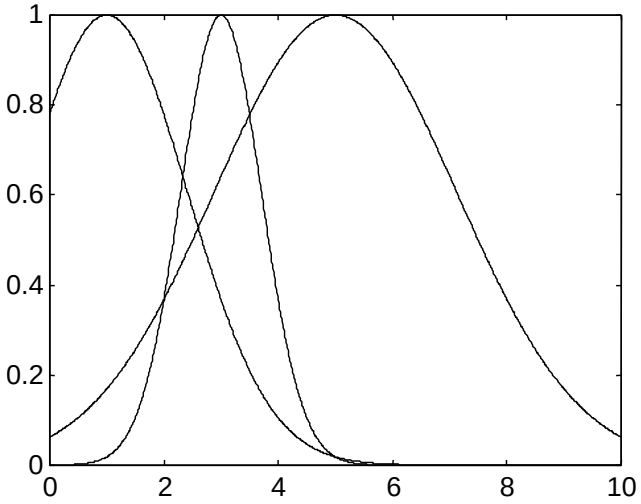
Caso 1: $m = N$

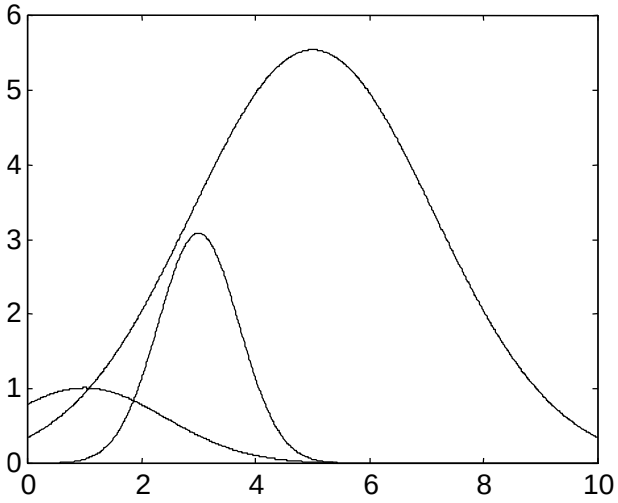
Pontos amostrados: (1,2); (3,7); (5,6)

$$\mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}; \quad \mathbf{w} = \begin{bmatrix} 0.945 \\ 2.850 \\ 5.930 \end{bmatrix}$$

Obs: As funções de base radial têm centros nos valores de x e dispersões arbitrárias.





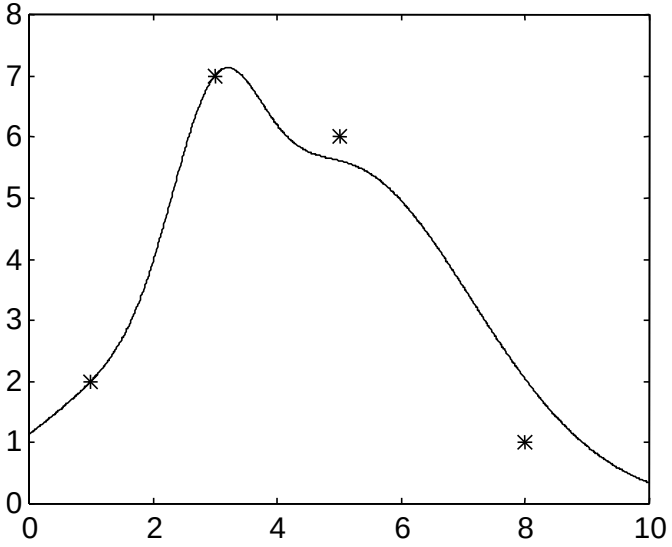


Caso 2: $m < N$

Pontos amostrados: (1,2); (3,7); (5,6); (8,1)

$$\mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}; \quad \mathbf{w} = \begin{bmatrix} 1.012 \\ 3.084 \\ 5.538 \end{bmatrix}$$

Obs: As funções de base radial são as mesmas do Caso 1.



17.2 Técnicas para Determinação dos Centros e Dispersões

- No caso de algoritmos que se ocupam apenas com o ajuste dos pesos da camada de saída de uma rede RBF (modelos lineares nos parâmetros), é necessário estabelecer algum critério para fixação dos centros.
- Existem critérios para o caso de número variável de centros (redes construtivas, por exemplo), mas serão mencionados aqui apenas aqueles geralmente empregados para o caso de um número fixo e previamente especificado de centros.
- Existem basicamente 3 alternativas:
    1. Espalhar os centros uniformemente ao longo da região em que se encontram os dados;
    2. Escolher aleatoriamente, ou segundo algum critério específico, um subconjunto de dados de entrada como centros;
    3. Auto-organizar os centros, de acordo com a distribuição dos dados de entrada.

- Quanto às dispersões das funções de base radial, embora elas possam ser distintas (e até ajustáveis) para cada centro, usualmente se adota uma única dispersão para todos os centros, na forma (HAYKIN, 2008):

$$\sigma = \frac{d_{\max}}{\sqrt{2m}}$$

onde m é o número de centros, e $d_{\max}$ é a distância máxima entre os centros.

- Com este critério de dispersão, evita-se que as funções de base radial sejam excessivamente pontiagudas, ou então com uma base demasiadamente extensa.

17.3 Seleção dos centros por auto-organização

- Para auto-organizar os centros, é suficiente aplicar algum algoritmo capaz de refletir a distribuição dos dados de entrada.
- O algoritmo a ser apresentado a seguir, é um algoritmo de clusterização denominado *k*-means (COVER & HART, 1967; MACQUEEN, 1967). Este algoritmo

se assemelha ao adotado para as redes de Kohonen, mas não leva em conta noções de vizinhança entre os centros. Ele também está sujeito a mínimos locais.

- Sejam $\{\mathbf{c}_j(t)\}_{j=1}^{m}$ os centros das funções de base radial na iteração *t*. O algoritmo *k*-means padrão-a-padrão pode ser descrito da seguinte forma:

1. *Inicialização*: Escolha valores aleatórios distintos para os centros $\mathbf{c}_j(t)$.
2. *Amostragem*: Tome aleatoriamente um vetor $\mathbf{x}_i$ do conjunto de padrões de entrada;
3. *Matching*: Determine o índice *k* do centro que mais se aproxima deste padrão, na forma: $k(\mathbf{x}_i) = \arg\min_j \|\mathbf{x}_i(t) - \mathbf{c}_j(t)\|, \quad j = 1,...,m.$
4. *Ajuste*: Ajuste os centros usando a seguinte regra:

$$\mathbf{c}_j(t+1) = \begin{cases} \mathbf{c}_j(t) + \gamma[\mathbf{x}_i(t) - \mathbf{c}_j(t)], & k = k(\mathbf{x}_i) \\ \mathbf{c}_j(t), & \text{alhures} \end{cases}$$

onde $\gamma \in (0,1)$ é a taxa de ajuste.

5. *Ciclo*: Repita os Passos 2 a 5 para todos os *N* padrões de entrada e até que os centros não apresentem deslocamento significativo após cada apresentação completa dos *N* padrões.

- Sejam $\{\mathbf{c}_j(t)\}_{j=1}^{m}$ os centros das funções de base radial na iteração *t*. O algoritmo *k*-means em batelada pode ser descrito da seguinte forma:

1. *Inicialização*: Escolha valores aleatórios distintos para os centros $\mathbf{c}_j(t)$.
2. *Matching*: Determine o índice *k* do centro que mais se aproxima de cada padrão, na forma: $k(\mathbf{x}_i) = \arg\min_j \|\mathbf{x}_i(t) - \mathbf{c}_j(t)\|, \quad j = 1,...,m$.
3. *Ajuste*: Defina a nova posição de cada um dos *m* centros como a média dos padrões cujo índice corresponde àquele centro.
4. *Ciclo*: Repita os Passos 2 e 3 enquanto houver mudança de índice de centro para pelo menos um dos padrões.

17.4 Aplicação das propostas de determinação de centros e dispersão

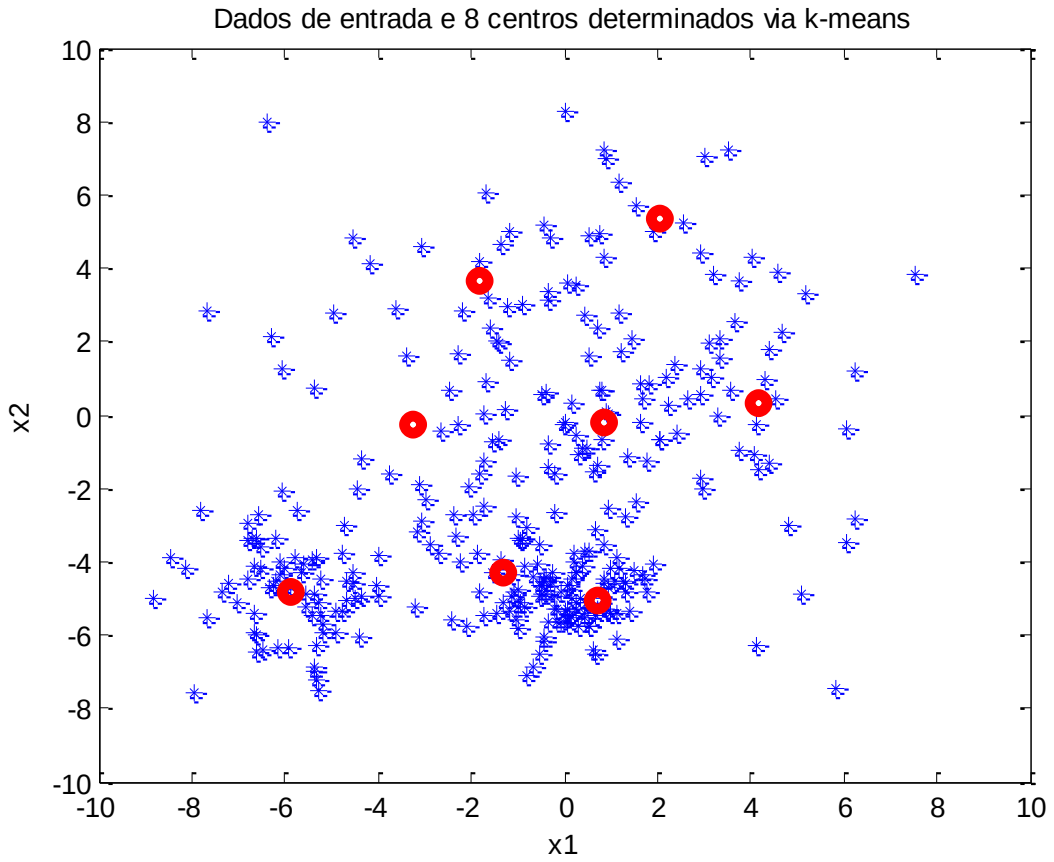


Figura 33 – Proposta de posicionamento dos centros das funções de base radial para uma rede neural RBF com 8 neurônios na camada intermediária.

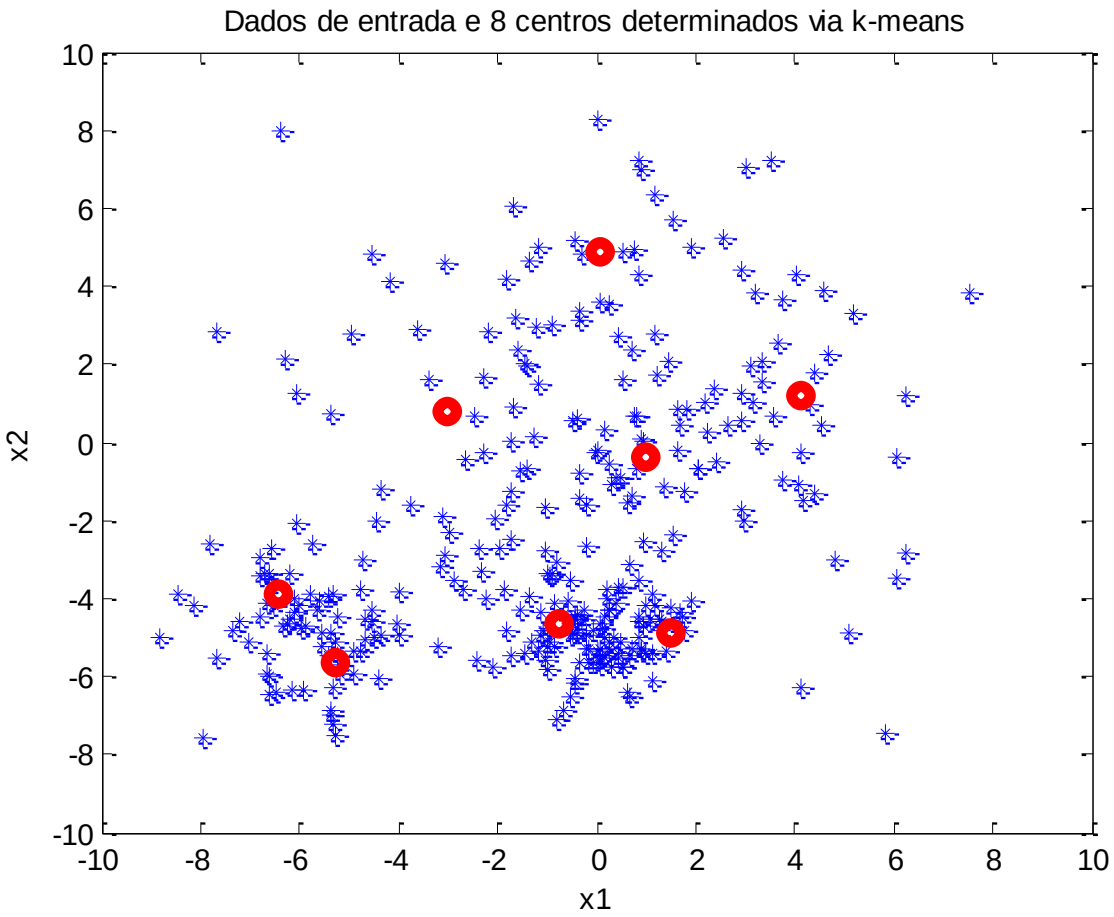


Figura 34 – Outra proposta de posicionamento dos centros para os mesmos dados, produzida por uma segunda execução do algoritmo k-means.

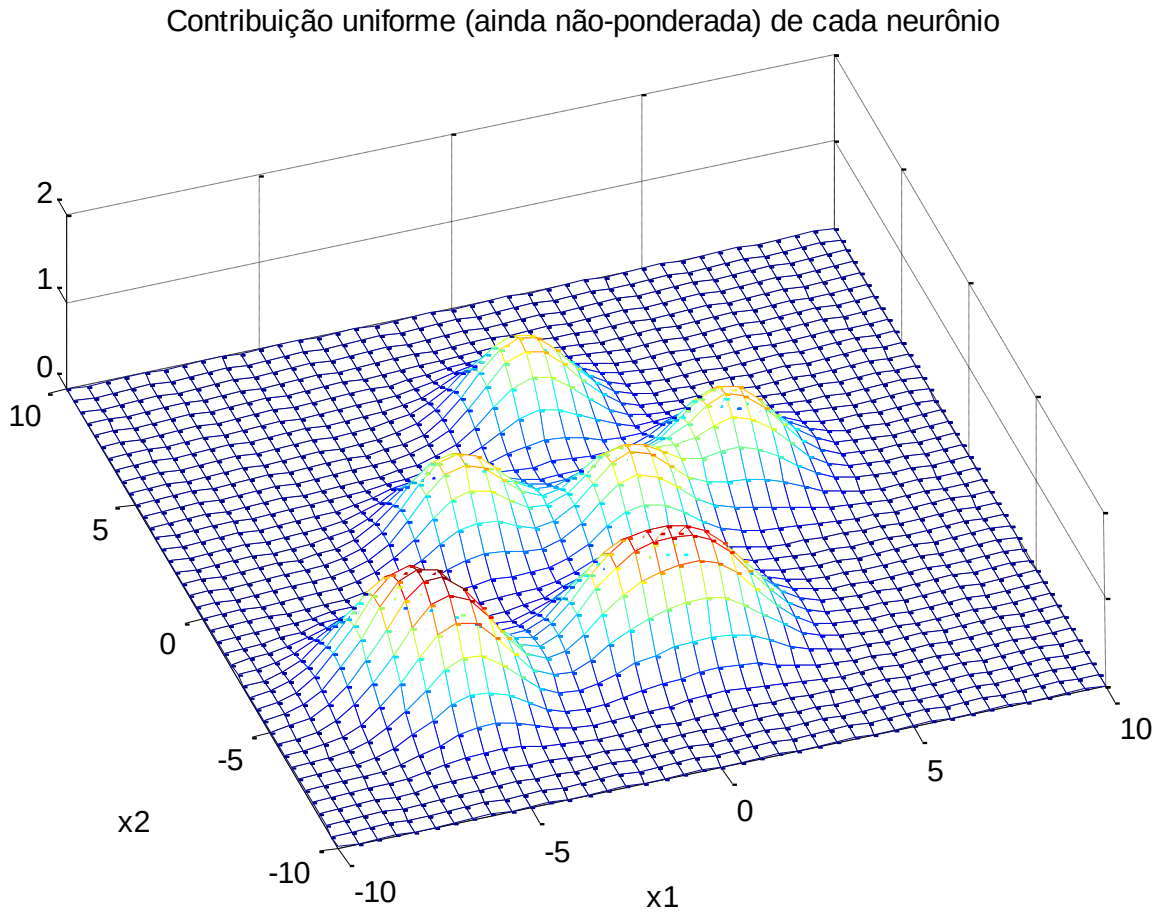


Figura 35 – Ativação dos neurônios da rede neural RBF com os centros da Figura 6, considerando todos os pesos de saída iguais a 1 e ausência de peso de limiar. A dispersão é a mesma para todas as funções de ativação, dada pela fórmula da pg. 92.

18. Referências

BARTLETT, P.L. For valid generalization the size of the weights is more important than the size of the network. *Advances in Neural Information Processing Systems*, volume 9, pp. 134-140, 1997.

BARTLETT, P.L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525-536,1998.

BATTITI, R. First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. *Neural Computation*, vol. 4, no. 2, pp. 141-166, 1992.

BISHOP, C. Exact Calculation of the Hessian Matrix for the Multilayer Perceptron. *Neural Computation*, vol. 4, no. 4, pp. 494-501, 1992.

BROOMHEAD, D.S. & LOWE, D. "Multivariate functional interpolation and adaptive networks", *Complex Systems*, 2: 321-355, 1988.

BUHMANN, M.D. "Radial Basis Functions: Theory and Implementations", Cambridge University, 2003.

COVER, T.M. & HART, P.E. "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, 13:21-27, 1967.

CYBENKO, G. Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303-314, 1989.

DUCHI, J., HAZAN, E., SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization, *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

GLOROT, X., BENGIO, Y. "Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256, 2010.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. The Elements of Statistical Learning – Data Mining, Inference, and Prediction, Springer, 2nd edition, 2009.

HAYKIN, S. "Neural Networks and Learning Machines", 3rd edition, Prentice-Hall, 2008.

HOERL, A.E., KENNARD, R. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, vol. 12, pp. 55-67, 1970.

HORNIK, K., STINCHCOMBE, M., WHITE, H. Multi-layer feedforward networks are universal approximators. *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989.

HORNIK, K., STINCHCOMBE, M., WHITE, H. Universal approximation of an unknown function and its derivatives using multilayer feedforward networks. *Neural Networks*, vol. 3, no. 5, pp. 551-560, 1990.

HORNIK, K., STINCHCOMBE, M., WHITE, H., AUER, P. Degree of Approximation Results for Feedforward Networks Approximating Unknown Mappings and Their Derivatives. *Neural Computation*, vol. 6, pp. 1262-1275, 1994.

HUANG, G.-B., CHEN, L., SIEW, C.-K. Universal Approximation Using Incremental Constructive Feedfoward Networks with Random Hidden Nodes. *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.

HUANG, G.-B., WANG, D.H., LAN, Y. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, vol. 2, pp. 107-122, 2011.

HUANG, G.-B., ZHOU, H., DING, X., ZHANG, R. Extreme Learning Machines for Regression and Multiclass Classification. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 42, no. 2, pp. 513-529, 2012.

HUANG, G.-B., ZHU, Q.-Y., SIEW, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. *Proceedings of the International Joint Conference on Neural Networks* (IJCNN'2004), vol. 2, pp. 985-990, 2004.

HUANG, G.-B., ZHU, Q.-Y., SIEW, C.-K. Extreme learning machine: theory and applications. Neurocomputing, vol. 70, pp. 489-501, 2006.

MACQUEEN, J. "Some methods for classification and analysis of multivariate observation", *in* L.M. LeCun and J Neyman (eds.) *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1:281-297, 1967.

MCCULLOCH, W.S. & PITTS, W. "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.

MINSKY, M.L. & PAPERT, S.A. "Perceptrons: Introduction to Computational Geometry", Expanded edition, The MIT Press, 1988 (1st edition: 1969).

PEDAMONTI, D. "Comparison of non-linear activation functions for deep neural networks on MNIST classification task", arXiv:1804.02763v1, 2018.

RUMELHART, D.E. & MCCLELLAND, J.L. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", volumes 1 & 2, The MIT Press, 1986. (ISBN: 026268053X)