



Instituto Tecnológico de Aeronáutica

---

# **Introdução às Redes Neurais e aos Grandes Modelos de Linguagem**

**Prof. Mauri Aparecido de Oliveira**

---

**CURSO – Extensivo**

**PO-249**



# **Backpropagation e Perceptrons de Múltiplas Camadas MLP**

ai



Uma rede neural é chamada de alimentada adiante (feedforward) se a informação se propaga apenas no sentido da entrada para a saída, não havendo conexões recorrentes ou atrasadas. A necessidade de não linearidade para resolver problemas como o XOR implica no uso de uma função de ativação (ou transferência) que possibilite obter uma não linearidade suave e monotonicamente crescente. A Figura–1 mostra exemplos de rede neural alimentada adiante.

Portanto, o novo modelo de rede neural a ser explorado é aquele que apresenta várias camadas e por isso é denominado de perceptron de múltiplas camadas (multilayer perceptron). A Figura–2 apresenta uma rede neural com  $m$  entradas,  $n$  neurônios e  $o$  saídas é comum indicar essa arquitetura do perceptron com sendo  $MLP(m, n, o)$ . A primeira camada é chamada de camada de entrada, a última camada é a camada de saída e entre elas pode haver uma ou mais camadas que recebem o nome de camadas intermediárias ou ocultas (hidden layers).

A dinâmica de funcionamento do perceptron de múltiplas camadas implica que em cada neurônio seja realizada uma transformação do sinal aplicando uma função de ativação à soma ponderada dos sinais que alimentam o neurônio a partir das conexões que realiza. O algoritmo de backpropagation fornece a regra de aprendizagem que é aplicada para modificar os pesos a partir de um conjunto de treinamento. Essa regra de aprendizagem denominada de backpropagation e que será construída em uma seção a seguir também recebe o nome de regra delta generalizada e foi proposta por Rumelhart, McClelland e Williams em 1986.

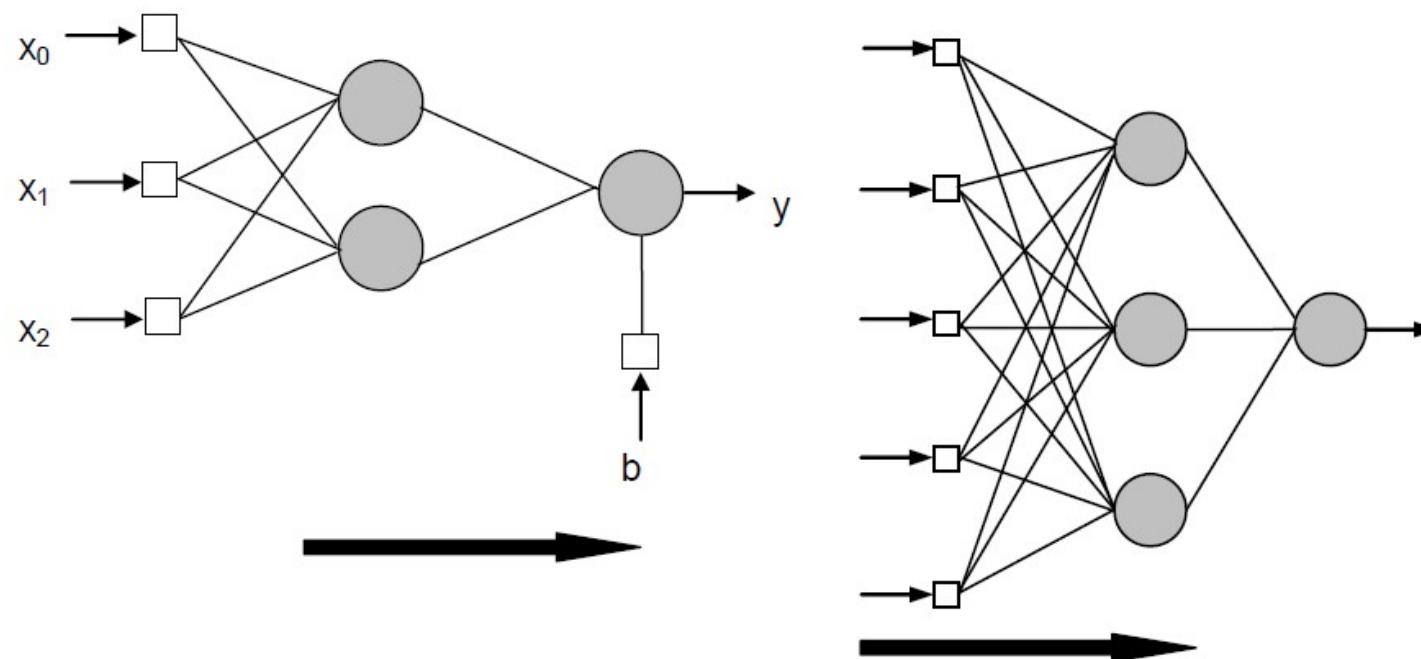


Figura-1 Exemplos de Rede Feedforward – Alimentada Adiante.

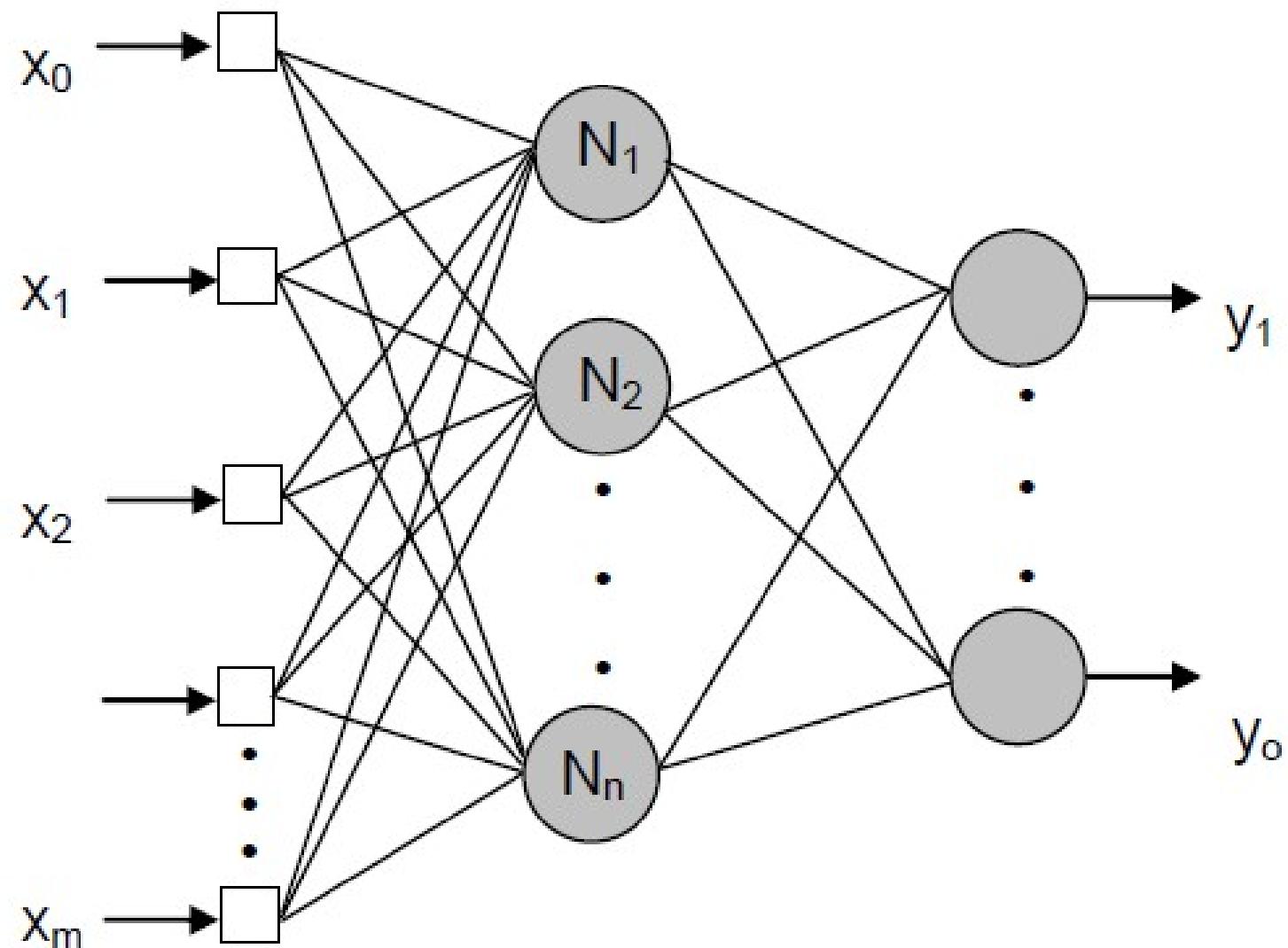


Após o treinamento, a rede neural de múltiplas camadas deve inferir uma generalização a partir dos dados de treinamento. A ideia subjacente é a mesma do perceptron simples, a saída produzida pela rede deve estar a mais próxima possível do valor desejado. A utilização da função sigmoide (ou logística) tem algumas vantagens quando comparada com as funções que são aplicadas no perceptron simples, entre elas pode-se destacar que é uma função diferenciável.

No caso do MLP a regra de aprendizagem é um pouco mais complicada que no caso do perceptron simples, e o ponto de partida é a definição de uma função de erro que representa a diferença entre a saída atual da rede e a saída que se desejaría produzir. Essa função de erro dependerá das variações dos pesos que deverão ser ajustados de tal forma a minimizar essa função.

O nome retropropagação deriva do fato de que os erros são retropropagados através da rede permitindo que os pesos entre todas as camadas sejam corretamente ajustados. Em cada iteração ou época, o erro deve decrescer, uma vez que o erro esteja abaixo de um valor pré-determinado o treinamento encerra.

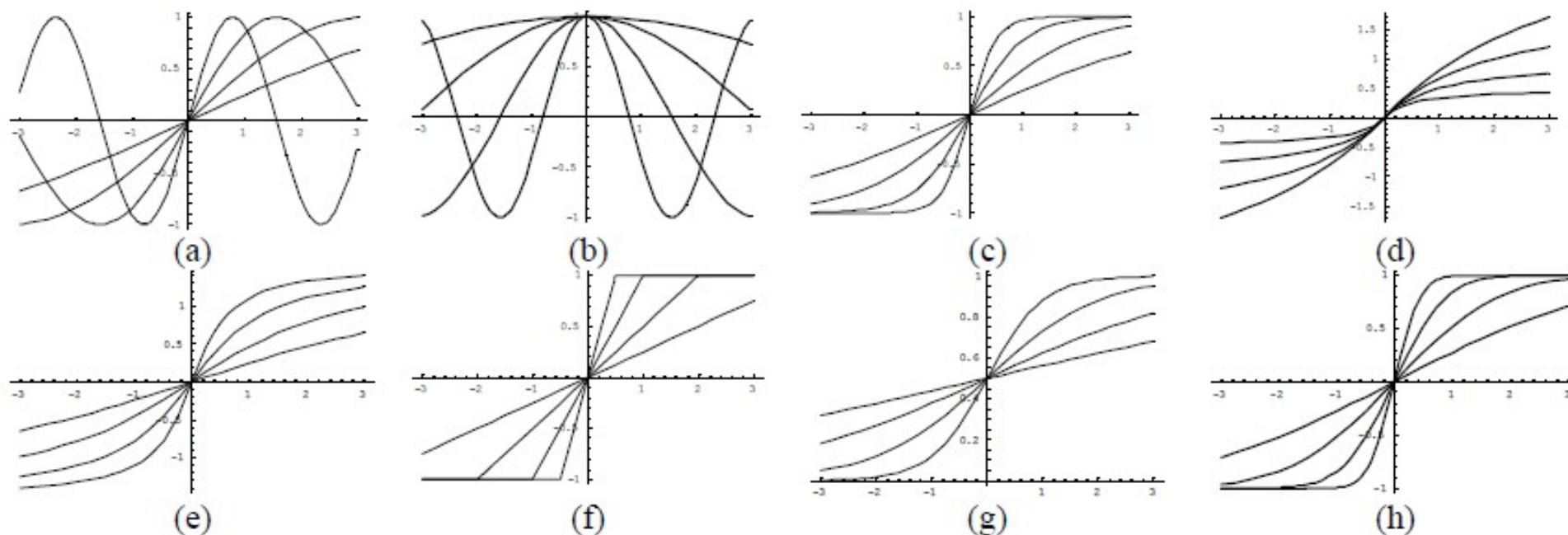
Para realizar o ajuste dos pesos, o algoritmo de backpropagation usa uma técnica comum em otimização, o cálculo do gradiente. À medida que os pesos são ajustados, a rede neural deve produzir saídas mais próximas dos valores desejados. A técnica do gradiente descendente é utilizada para determinar o impacto dos pesos sobre a função de erro. A taxa de aprendizagem pondera o gradiente podendo acelerar ou retardar a aprendizagem. Além da taxa de aprendizagem o algoritmo de backpropagation considera um outro fator de escalonamento da aprendizagem denominado de momento. O momento determina a porcentagem de mudança do peso da iteração anterior que deve ser aplicada a iteração atual. O momento propicia à rede neural uma força adicional em manter a direção correta na busca pelo mínimo da função de erro. Tanto a taxa de aprendizagem como o momento são valores arbitrados pelo analista.



**Figura–2** Rede MLP( $m, n, o$ ).

# Função de Ativação

A função de ativação é a regra para mapeamento das entradas somadas,  $v$ , do neurônio até sua saída  $e$ , por uma escolha adequada, isto significa a introdução de uma capacidade de processar não linearidade na rede. Na prática, estas funções são escolhidas de tal forma a serem monotônicas e saturar nos extremos  $[0,1]$  ou  $[-1,1]$ . Existem vários tipos de funções de ativação, sendo que as primeiras a serem utilizadas são as clássicas: linear, sigmoide, linear por partes, função de limiar, arco tangente, seno, cosseno, tangente hiperbólica, função erf,  $v/(1+|v|)$ , Gaussiana, etc. Ver Figura-3.



**Figura-3** Apresenta oito funções de ativação: (a) seno, (b) cosseno, (c) tangente hiperbólica, (d)  $v/(1+|v|)$ , (e) arco tangente, (f) linear por partes, (g) sigmoide e (h) erf. Todas as figuras apresentam ganhos de 0,25, 0,5, 1 e 2.



A mais simples é a função linear, cuja saída é igual à entrada. A função sigmoide é a forma mais comum de função de ativação utilizada na construção de uma RNA treinada com backpropagation.

$$\varphi(v) = \frac{1}{1 + e^{-gv}} , \quad (1)$$

em que  $g$  é o ganho, ou parâmetro de inclinação da função sigmoide. Variando-se o parâmetro  $g$ , obtêm-se funções sigmoides com diferentes inclinações. Para escolha de outros tipos de funções de ativação, ver Chen *et al.* (2001) e Apicella *et al.* (2021). Considerando uma função sigmoide em que  $g$  é igual a 1 sua derivada é dada por

$$\varphi(v) = (1 + e^{-v})^{-1},$$
$$\varphi'(v) = -(1 + e^{-v})^{-2} e^{-v} (-1),$$

$$\varphi'(v) = \frac{1}{1+e^{-v}} \frac{e^{-v}}{1+e^{-v}}, \quad (2)$$

como  $[1 - \varphi(v)]$  produz

$$\begin{aligned} 1 - \varphi(v) &= 1 - \frac{1}{1+e^{-v}}, \\ 1 - \varphi(v) &= \frac{1+e^{-v}}{1+e^{-v}} - \frac{1}{1+e^{-v}}, \\ 1 - \varphi(v) &= \frac{e^{-v}}{1+e^{-v}}, \end{aligned}$$

substituindo em (2) resulta em

$$\varphi'(v) = \varphi(v)[1 - \varphi(v)]. \quad (3)$$

# Derivação Matemática do Algoritmo de Backpropagation

O algoritmo de backpropagation é utilizado para aprender quais são os pesos de uma rede neural de múltiplas camadas. Para isso é utilizado o método do gradiente descendente para minimizar a soma dos erros quadrados entre os valores de saída da rede e os valores desejados. Conceitualmente, ocorre uma propagação de ativações ao longo das sinapses para produzir uma saída e os erros são retropropagados para realizar as mudanças dos pesos.

Antes de iniciar a construção matemática, será utilizada a seguinte notação:

- O subscrito  $k$  denota que o neurônio está na camada de saída, ou à direita do neurônio  $j$ .
- O subscrito  $j$  denota que o neurônio está na camada oculta, ou à direita do neurônio  $i$ .
- O subscrito  $i$  denota que o neurônio está na camada de entrada.
- $w_{kj}(n)$  denota o peso da sinapse que liga a camada oculta à camada de saída, na iteração  $n$ .
- $w_{ji}(n)$  denota o peso da sinapse que liga a camada de entrada à camada oculta, na iteração  $n$ .
- $e_j(n)$  denota o sinal de erro na saída no neurônio  $j$ , na iteração  $n$ .
- $d_j(n)$  denota o valor desejado na saída no neurônio  $j$ , na iteração  $n$ .
- $y_j(n)$  denota o valor de saída do neurônio  $j$ , na iteração  $n$ .
- $v_j(n)$  denota o campo local induzido do neurônio  $j$ , na iteração  $n$ .
- $\varphi_j(\cdot)$  denota a função de ativação associada ao neurônio  $j$ .
- $b_j$  denota o bias aplicado ao neurônio  $j$ , também será escrito como  $w_{j0}$ .
- $\eta$  denota a taxa de aprendizagem.

Incialmente, consideramos que o erro é a diferença entre a função de ativação de uma saída atual de um determinado nó e o valor desejado para aquele nó (ou neurônio). O erro total é a soma de todos esses erros para cada nó de saída. Além disso, como não queremos que erros positivos e negativos cancelem um ao outro, realizamos a soma desses erros ao quadrado. Como estamos construindo uma função de erro visando buscar o seu mínimo, e isso normalmente faz uso do processo de cálculo da primeira derivada, convenientemente multiplicamos a soma de erros ao quadrado por 1/2. Assim temos que

$$e_j(n) = d_j(n) - y_j(n), \quad (19)$$

sendo que o neurônio  $j$  está na camada de saída, uma vez que o erro está definido apenas para a camada de saída. A função de erro total é escrita então como

$$E(n) = \frac{1}{2} \sum_j e_j^2(n). \quad (20)$$

O objetivo é ajustar os pesos da rede neural para reduzir o erro total, então podemos considerar que a variação dos pesos é proporcional ao negativo da derivada do erro total com relação ao peso, escrevendo essa ideia de maneira sucinta tem-se

$$\Delta w \propto -\frac{\partial E}{\partial w}. \quad (21)$$



O peso da sinapse que liga a saída do neurônio  $i$  à entrada do neurônio  $j$  será alterado na iteração  $n$  seguindo a regra criada a partir do conceito de descida do gradiente, dessa forma tem-se que

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}, \quad (22)$$

lembrando que o sinal negativo indica que as alterações nos pesos são realizadas na direção em que decresce o erro.

A Figura–5 apresenta os elementos envolvidos e o fluxo de sinais utilizados para estabelecer as primeiras relações na construção do algoritmo de backpropagation. Considerando que o neurônio  $j$  possui  $m$  entradas, o campo local induzido é dado por

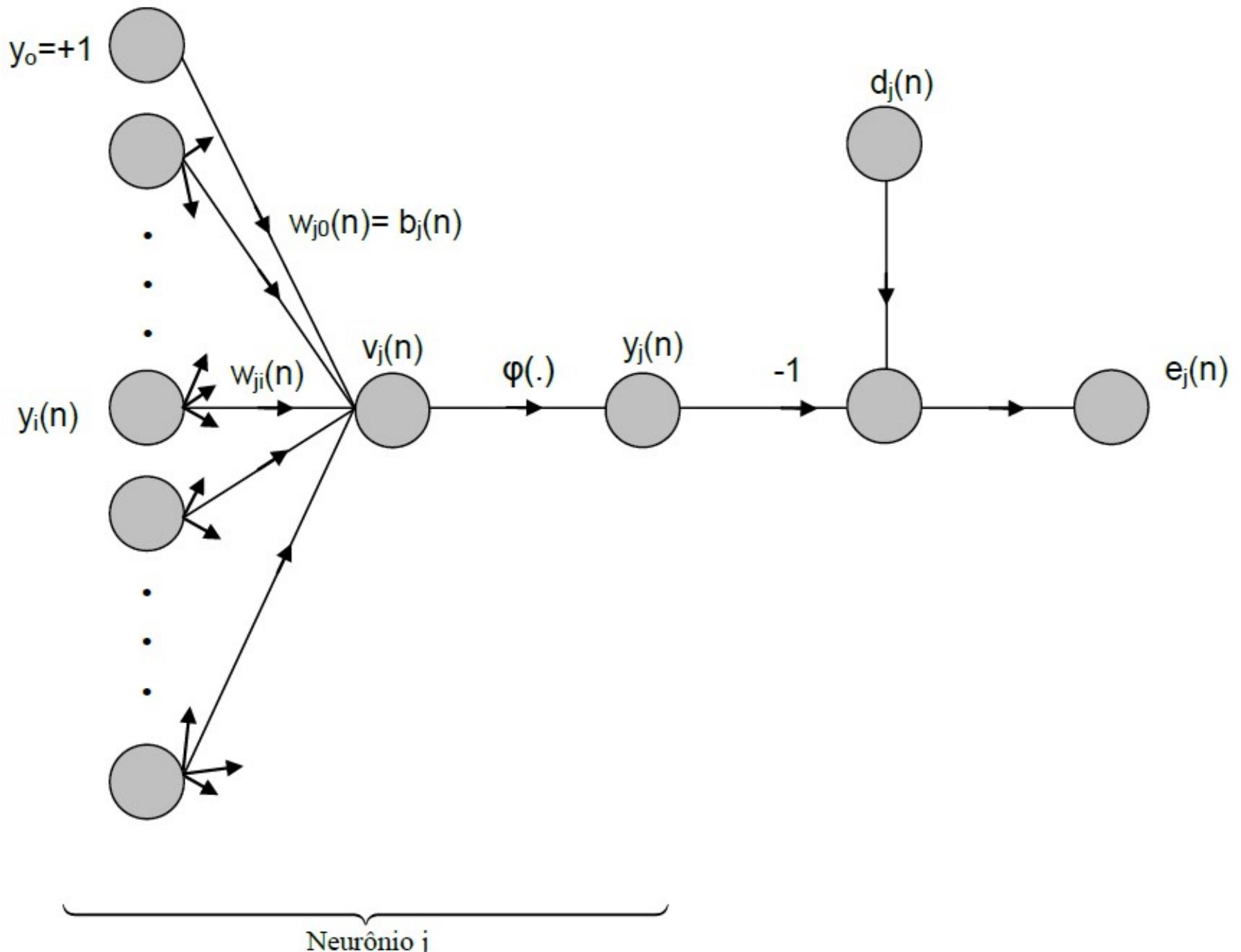
$$v_j(n) = \sum_{i=0}^m w_{ji}(n) y_i(n). \quad (23)$$

A saída do neurônio  $j$  na iteração  $n$  é obtida aplicando a função de ativação ao campo local induzido obtendo

$$y_j(n) = \varphi_j(v_j(n)). \quad (24)$$

A função de erro não é diretamente uma função do peso, por isso é aplicada a regra da cadeia do cálculo para expandir a derivada parcial da equação (22). Assim, o gradiente pode ser escrito como um produto de derivadas parciais que envolvem parcelas conhecidas da construção do neurônio. Dessa forma, tem-se que

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \quad (25)$$

Figura-5 Neurônio  $j$  e fluxo de sinais.

Das relações construídas anteriormente, podemos obter cada uma das derivadas parciais da equação (25), sendo que da equação (20) obtém-se

$$\frac{\partial E(n)}{\partial e_j(n)} = \frac{\partial}{\partial e_j(n)} \left[ \frac{1}{2} \sum_j e_j^2(n) \right],$$

ou

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n). \quad (26)$$

No caso da derivada do erro com relação à saída, pode-se utilizar a equação que produz

$$\begin{aligned} \frac{\partial e_j(n)}{\partial y_j(n)} &= \frac{\partial}{\partial y_j(n)} [d_j(n) - y_j(n)], \\ \frac{\partial e_j(n)}{\partial y_j(n)} &= -1. \end{aligned} \quad (27)$$

Para obter a derivada parcial de  $y_j(n)$  com relação ao campo local induzido  $v_j(n)$  pode-se utilizar a equação (24), de onde obtemos

$$\begin{aligned}\frac{\partial y_j(n)}{\partial v_j(n)} &= \frac{\partial}{\partial v_j(n)} [\varphi_j(v_j(n))], \\ \frac{\partial y_j(n)}{\partial v_j(n)} &= \varphi'_j(v_j(n)).\end{aligned}\tag{28}$$

A derivada de  $v_j(n)$  com relação ao peso na equação (23) produz

$$\begin{aligned}\frac{\partial v_j(n)}{\partial w_{ji}(n)} &= \frac{\partial}{\partial w_{ji}(n)} \left[ \sum_{i=0}^m w_{ji}(n) y_i(n) \right], \\ \frac{\partial v_j(n)}{\partial w_{ji}(n)} &= y_i(n).\end{aligned}\tag{29}$$

Substituindo os resultados das equações (26), (27), (28) e (29) na equação (25) resulta em

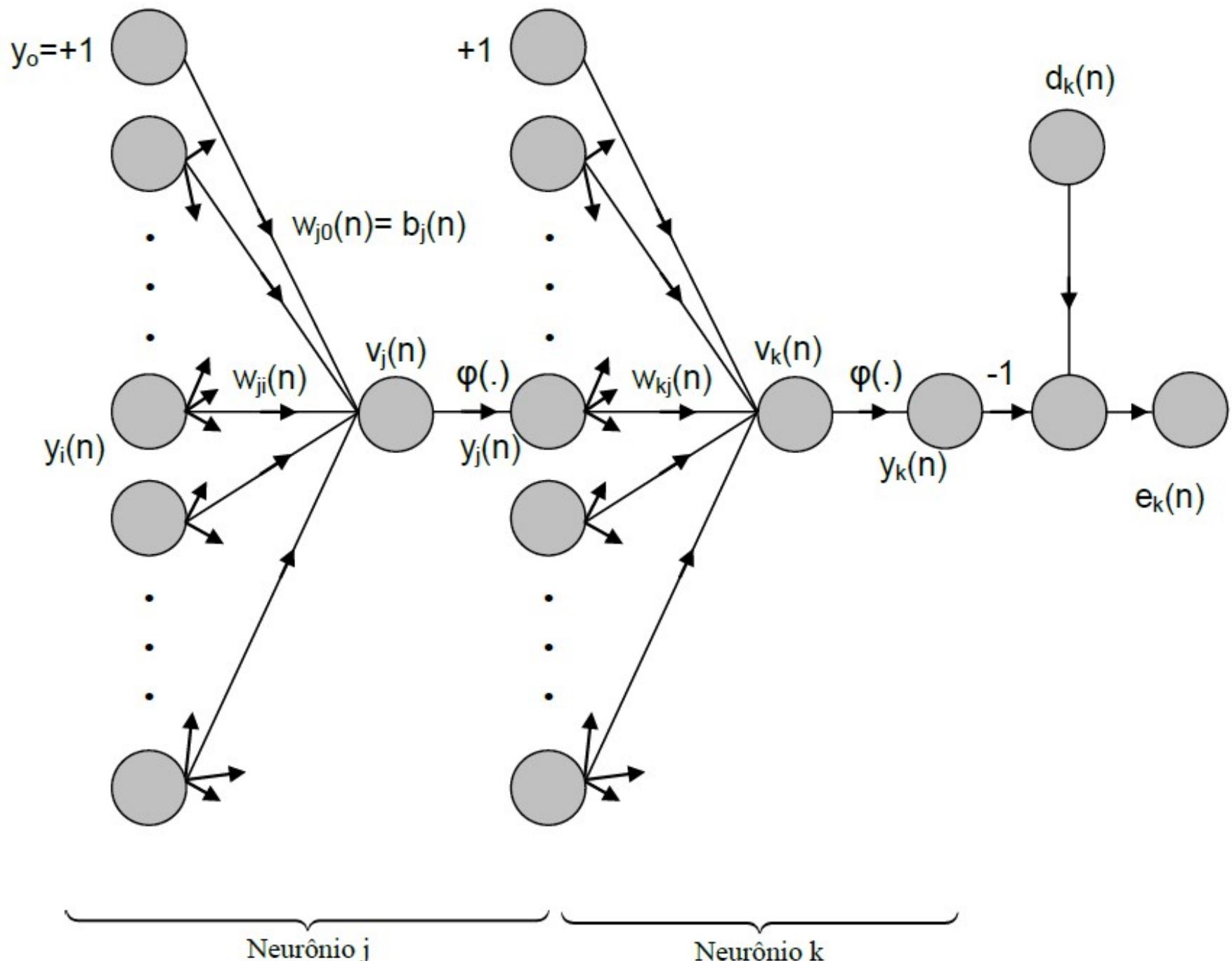
$$\begin{aligned}\frac{\partial E(n)}{\partial w_{ji}(n)} &= e_j(n)(-1)\varphi'_j(v_j(n))y_i(n), \\ \frac{\partial E(n)}{\partial w_{ji}(n)} &= -e_j(n)\varphi'_j(v_j(n))y_i(n).\end{aligned}\tag{30}$$

A seguir é definida uma quantidade denominada de gradiente local do erro, denotada por  $\delta_j(n)$ , e calculada como

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)},$$
$$\delta_j(n) = -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)},$$

onde, dos resultados anteriores produz

$$\delta_j(n) = -e_j(n)(-1)\varphi'_j(v_j(n)),$$
$$\delta_j(n) = e_j(n)\varphi'_j(v_j(n)). \quad (31)$$



Figura–6 Neurônio  $j$ , Neurônio  $k$  e fluxo de sinais.

De acordo com Haykin (2001) a vantagem em utilizar o gradiente local é que ele aponta para as modificações necessárias nos pesos sinápticos. O gradiente local é uma medida de quanto do erro da saída é devido a um peso particular  $w_{ji}(n)$ . Aplicando os resultados (30) e (31) na expressão para obtenção da atualização do peso dado pela equação (4) temos que

$$\Delta w_{ji}(n) = \eta \delta_i(n) y_j(n). \quad (32)$$



Nesse ponto é necessário considerar duas situações distintas:

1. Se o neurônio  $j$  é um nó de saída, então  $\delta_j(n)$  é o produto da derivada da função de ativação,  $\phi'_j(v_j(n))$ , e a diferença no  $j$ -ésimo componente entre o valor desejado e a saída da rede;
2. Se o neurônio  $j$  é um nó oculto da rede, não existe um valor desejado especificado para este neurônio. A solução é retropropagar os valores dos gradientes locais camada por camada através da rede, de tal forma a fazer que os pesos sejam atualizados. O resultado será que  $\delta_j(n)$  é dado pelo produto da derivada da função de ativação,  $\phi'_j(v_j(n))$ , e a soma ponderada dos gradientes locais calculados para os nós da próxima camada oculta ou camada de saída que estão conectadas ao neurônio  $j$ , sendo ponderados pelos pesos das conexões existentes com a próxima camada. Para demonstrar esse resultado considere a Figura–6 que mostra dois neurônios ( $j$  e  $k$ ) e o fluxo de sinais através deles.

A partir da definição de gradiente local , consideramos o caso em que o neurônio  $j$  está em uma camada oculta, então

$$\begin{aligned}\delta_j(n) &= -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)}, \\ \delta_j(n) &= -\frac{\partial E(n)}{\partial y_j(n)} \varphi'_j(v_j(n)).\end{aligned}\quad (33)$$

A função de erro,  $E(n)$ , no caso da rede neural mostrada na Figura–6, será obtida do somatório de todos os erros obtidos na saída do neurônio  $k$ , ou seja

$$E(n) = \frac{1}{2} \sum_k e_k^2(n), \quad (34)$$

onde o neurônio  $k$  é um nó de saída.

Diferenciando a função de erro (34) com relação a  $y_j(n)$  produz

$$\begin{aligned}\frac{\partial E(n)}{\partial y_j(n)} &= \frac{\partial}{\partial y_j(n)} \left[ \frac{1}{2} \sum_k e_k^2(n) \right], \\ \frac{\partial E(n)}{\partial y_j(n)} &= \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)}.\end{aligned}\quad (35)$$



Como  $e_k(n)$  não é uma medida que se possa obter diretamente a partir do valor de  $y_j(n)$  podemos utilizar a regra da cadeia para obter  $\partial e_k(n)/\partial y_j(n)$ , assim podemos reescrever a equação (35) da seguinte forma

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}. \quad (36)$$

Dado que o neurônio  $k$  é um neurônio de saída, temos

$$\begin{aligned} e_k(n) &= d_k(n) - y_k(n), \\ e_k(n) &= d_k(n) - \varphi_k(v_k(n)), \end{aligned}$$

o que permite calcular a derivada parcial do erro  $e_k(n)$  com relação ao campo local induzido  $v_k(n)$ . Assim temos

$$\begin{aligned} \frac{\partial e_k(n)}{\partial v_k(n)} &= \frac{\partial}{\partial v_k(n)} [d_k(n) - \varphi_k(v_k(n))], \\ \frac{\partial e_k(n)}{\partial v_k(n)} &= -\varphi'_k(v_k(n)). \end{aligned} \quad (37)$$

Da Figura–6 facilmente verificamos que no caso do neurônio  $k$  o campo local induzido é dado por

$$v_k(n) = \sum_{j=0}^m w_{kj}(n) y_j(n), \quad (38)$$

sendo que  $w_{k0}(n)$  corresponde ao bias aplicado ao neurônio  $k$ . Diferenciando (38) com relação a  $y_j(n)$  resulta em

$$\begin{aligned} \frac{\partial v_k(n)}{\partial y_j(n)} &= \frac{\partial}{\partial y_j(n)} \left[ \sum_{j=0}^m w_{kj}(n) y_j(n) \right], \\ \frac{\partial v_k(n)}{\partial y_j(n)} &= w_{kj}(n). \end{aligned} \quad (39)$$

Substituindo os resultados de (39) e (37) em (36) produz

$$\begin{aligned} \frac{\partial E(n)}{\partial y_j(n)} &= \sum_k e_k(n) [-\varphi'_k(v_k(n))] w_{kj}(n), \\ \frac{\partial E(n)}{\partial y_j(n)} &= -\sum_k e_k(n) \varphi'_k(v_k(n)) w_{kj}(n). \end{aligned} \quad (40)$$



Porém, da definição de gradiente local podemos escrever que

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k \delta_k(n) w_{kj}(n). \quad (41)$$

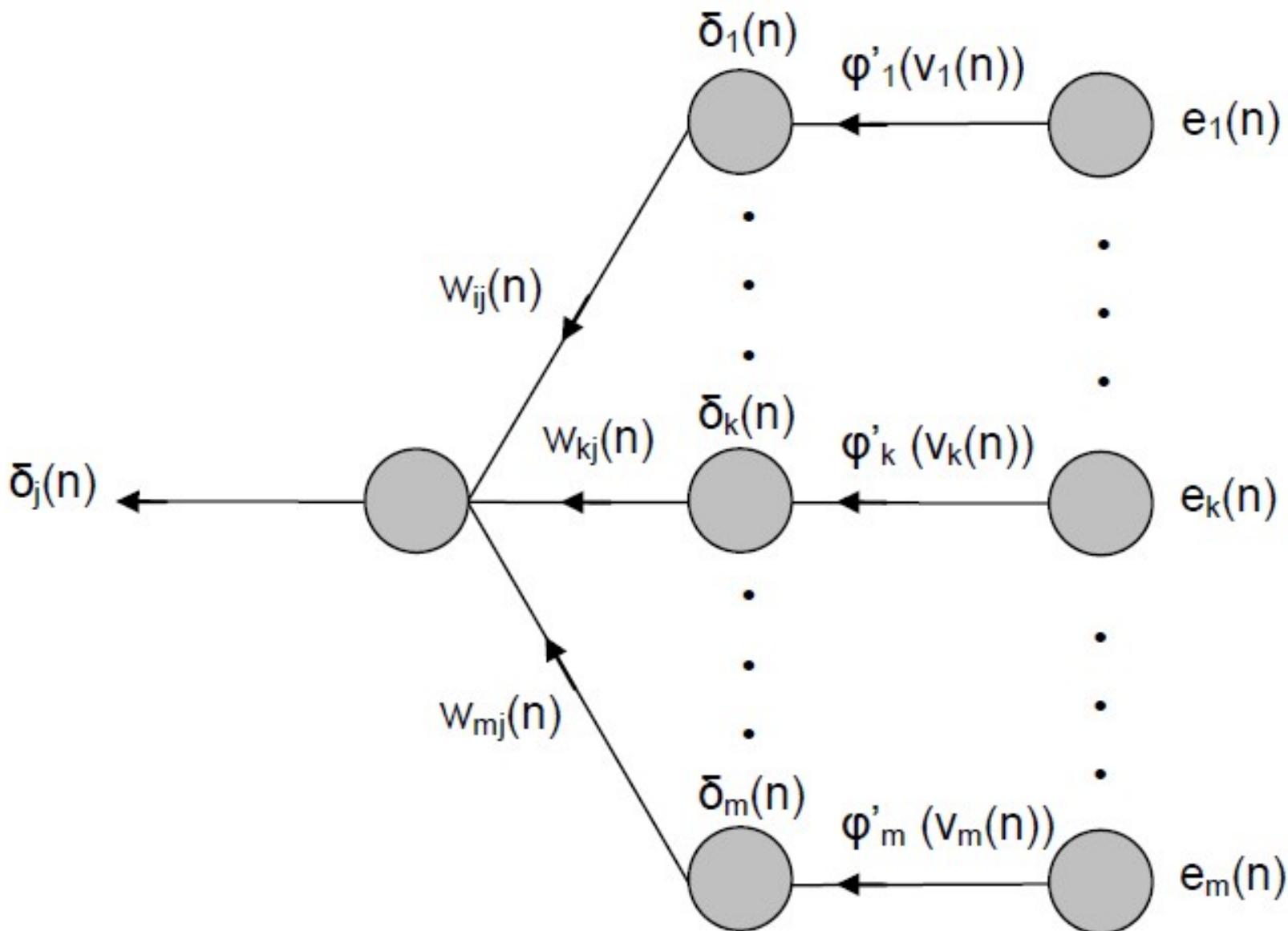
A equação (32) é denominada de **regra delta generalizada** e as equações utilizadas para calcular os gradientes locais em cada uma das situações descritas acima, ou seja: (i) o neurônio  $j$  é um nó de saída e (ii) o neurônio  $j$  é um nó na camada oculta. Considerando uma função de ativação do tipo sigmoide, para a primeira situação temos que o gradiente local é dado por

$$\delta_j(n) = \varphi_j(v_j(n)) (1 - \varphi_j(v_j(n))) [d_j(n) - y_j(n)], \quad j \text{ é um neurônio de saída.} \quad (42)$$

Para obter o gradiente local no caso em que o neurônio  $j$  está na camada oculta, substituímos o resultado (41) em (33) o que produz

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n), \quad j \text{ é um neurônio oculto.} \quad (43)$$

A equação (43) indica que o gradiente local do neurônio  $j$  na camada oculta está relacionado aos gradientes locais e pesos sinápticos de todos os nós da camada seguinte (nesse caso, a camada de saída). Portanto, na camada oculta,  $\delta_j(n)$ , só pode ser calculado após os gradientes locais da camada seguinte terem sido calculados. Consequentemente, o cálculo envolvido no algoritmo de backpropagation está baseado na computação dos resultados de camada por camada, iniciando da camada de saída. A Figura–7 apresenta o fluxo do sinal  $\delta_j(n)$  da equação (43) considerando que a camada de saída possui  $m$  neurônios.



**Figura–7** Retropropagação do sinal de erro para calcular o gradiente local na camada oculta.

A derivada da função de ativação sigmoide na equação (3) mostra que podemos escrever  $\varphi'(v)$  em termos de  $\varphi(v)$  e da equação (24) temos que a saída  $y_j(n)$  é igual a função de ativação aplicada a  $v_j(n)$ , assim sabendo que

$$\begin{cases} \varphi'(v) = \varphi(v)[1 - \varphi(v)] \\ y_j(n) = \varphi_j(v_j(n)) \end{cases}$$

resulta em

$$\varphi'(v_j(n)) = y_j(n)[1 - y_j(n)]. \quad (44)$$



Quando o neurônio  $j$  estiver localizado na camada de saída, podemos utilizar uma nomenclatura específica para descrever essa situação, para isso fazemos  $y_j(n) = o_j(n)$ . Assim, as equações do gradiente local no caso do neurônio  $j$  estar na saída da rede (42) e no caso em que está na camada oculta (43) são agora expressas como

### O neurônio $j$ é um nó de saída

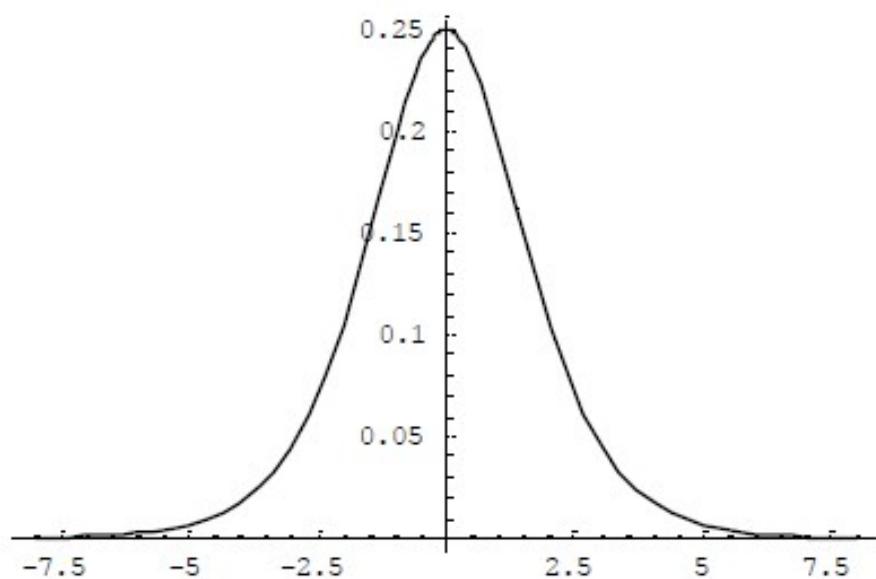
$$\delta_j(n) = o_j(n)(1 - o_j(n)) [d_j(n) - o_j(n)] \quad (45)$$

### O neurônio $j$ é um nó oculto

$$\delta_j(n) = y_j(n)[1 - y_j(n)] \sum_k \delta_k(n) w_{kj}(n) \quad (46)$$

e utilizadas para computar a atualização dos pesos, sendo que  $\Delta w_{ji}$  é igual a  $\eta \delta_j(n) y_i(n)$ .

Durante o processo de minimização em que utiliza-se o algoritmo de backpropagation é necessário que se tenha à disposição a derivada da função de ativação, que no caso da função sigmoide é dada por (3). A curva que representa a derivada da função sigmoide é apresentada na Figura–8. Pode-se verificar que a derivada da função de ativação alcança seu valor máximo em 0,25, considerando um ganho  $g$  igual a 1. O máximo da derivada da função de ativação é alcançado quando  $y_j(n) = 0,5$ , como pode facilmente ser verificado na equação (44), o mínimo da derivada da função de ativação é igual a zero e é obtido quando  $y_j(n) = 0$  ou  $y_j(n) = 1$ .



**Figura–8** Curva da derivada da função de ativação, ganho igual a 1.



O valor da modificação do peso sináptico da rede neural é proporcional à derivada da função de ativação, isto resulta que para uma função de ativação sigmoide, os pesos sinápticos são modificados mais intensamente para aqueles neurônios da rede neural onde os valores dos sinais estão no meio do seu intervalo (Haykin, 2001, Rumelhart *et al.*, 1986a). Os pesos são aspectos cruciais para a rede neural (Balkin, 2000). Inicialmente são associados valores aleatórios em torno de zero. Os pesos são incrementalmente ajustados durante a fase de treinamento, inicializações típicas situam-se no intervalo [-1,1].

Em resumo há dois passos que precisam ser realizados na condução dos cálculos para efetuar a atualização dos pesos sinápticos da rede neural. O primeiro é o passo para frente (forward step) e o segundo é o passo para trás, a retropropagação (backward step).

No passo para frente, forward, os sinais de entrada são conduzidos adiante através dos nós da rede para a camada oculta. Cada entrada é multiplicada pelo respectivo peso de conexão dos nós e os valores dos nós na camada oculta são calculados. Aplica-se a função de ativação para calcular as saídas dos neurônios ocultos, que são propagados para a camada seguinte que pode ser uma outra camada oculta ou a camada de saída da rede. Determinadas as saídas da rede, o passo adiante está concluído. Veja que nessa etapa, os pesos sinápticos não foram alterados em nenhum ponto da rede neural e têm valores iguais às atribuições iniciais. Sendo que a inicialização dos pesos pode ser arbitrária ou randômica.

No passo de retropropagação, backward, são calculados os erros nos neurônios de saída. Conhecidos os erros, é utilizada a retropropagação para ajuste dos pesos. Ou seja, o erro é propagado da camada de saída para a camada oculta à esquerda através da rede. É nesse ponto que são utilizadas as informações sobre a taxa de aprendizagem ( $\eta$ ) e do termo de momento ( $\alpha$ ) para atualizar os pesos.

Em 1986 no trabalho *Learning Internal Representations by Error Propagation*, Rumelhart propõe uma alteração na regra delta mostrada na equação (14) acrescentando um termo de momento. A ideia consiste em incorporar à atualização atual do peso alguma influência das iterações passadas.



Dessa forma, a equação (14) é alterada para

$$\Delta w_{ji}(n) = \eta \delta_i(n) y_j(n) + \alpha \Delta w_{ji}(n-1), \quad (47)$$

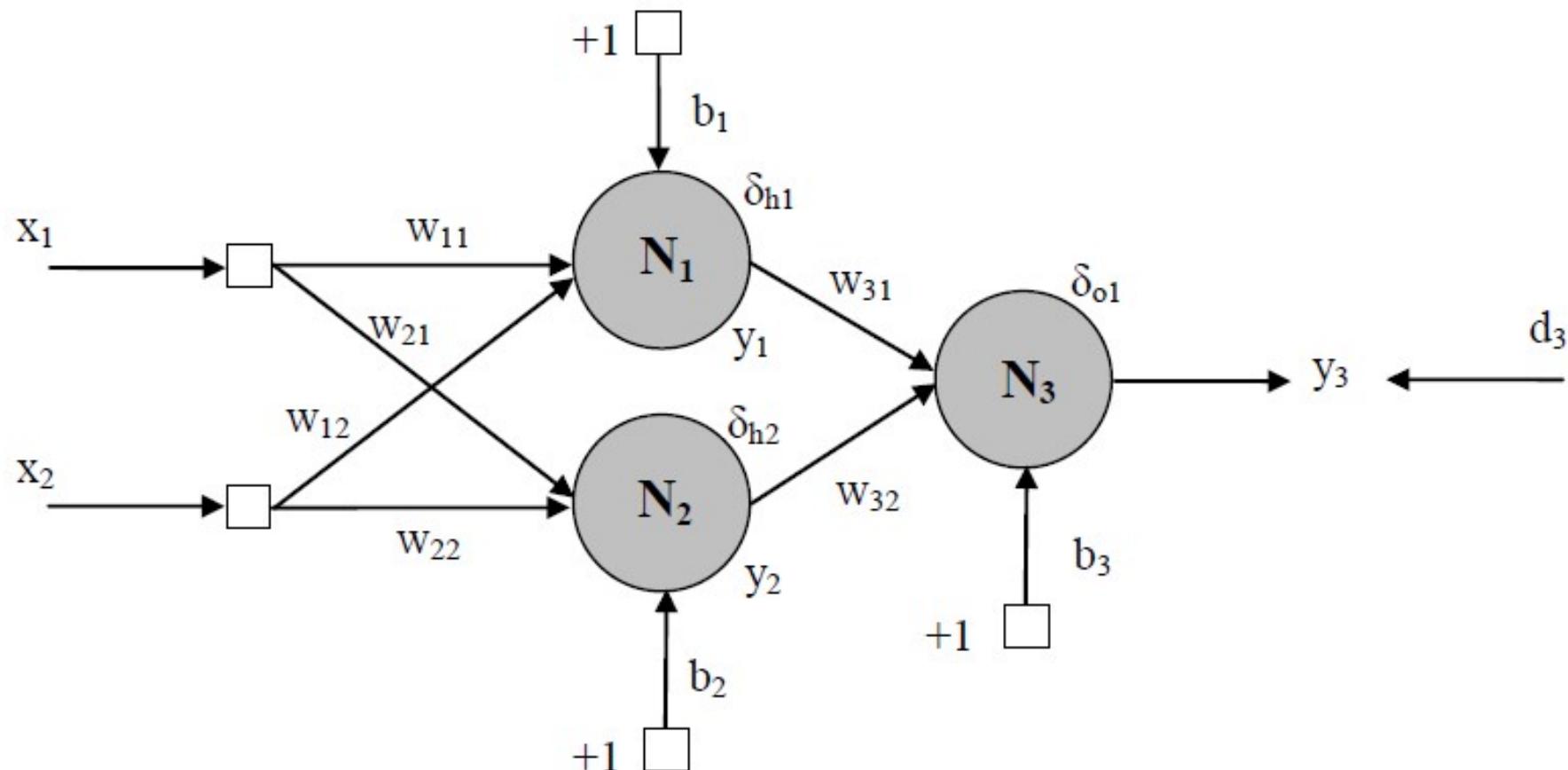
ou

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} + \alpha \Delta w_{ji}(n-1), \quad (48)$$

onde  $0 < \alpha < 1$  é o termo de momento e descreve a quantidade da influência das iterações anteriores sobre a iteração atual. O momento introduz um efeito de amortecimento sobre o procedimento de busca, evitando oscilações em regiões irregulares da superfície de erro. Devido ao termo de momento, se os pesos iniciam um movimento em uma direção particular, eles tenderão a continuar o movimento nessa direção, evitando a tendência de oscilar no caso em que a taxa de aprendizagem está com um valor muito grande.

## EXEMPLO

Utilize a rede neural mostrada na Figura–13 para resolver o problema do XOR. Para isso aplique o algoritmo de backpropagation.



**Figura–13** Rede neural MLP(2, 2, 1).



Considere inicialmente que

$$w_{11} = 0,$$

$$w_{21} = 0,$$

$$w_{12} = 0,$$

$$w_{22} = 0,$$

$$w_{31} = 0,$$

$$w_{32} = 0,$$

$$b_1 = 0,$$

$$b_2 = 0,$$

$$b_3 = 0,$$

também assuma que a função de ativação nos três neurônios é a função sigmoide dada por

$$\varphi(v) = \frac{1}{1 + \exp(-v)},$$

com primeira derivada

$$\varphi'(v) = \varphi(v)[1 - \varphi(v)].$$

Adotar a taxa de aprendizagem como sendo  $\eta = 0,1$  e o termo de momento  $\alpha = 0,5$ . Lembrando que

$x_1$	$x_2$	$x_1 \vee x_2$
1	1	0
1	0	1
0	1	1
0	0	0

Definida a arquitetura da rede, atribuídos os valores iniciais de pesos e bias, taxa de aprendizagem e termo de momento, precisamos especificar um valor de erro que será aceitável no processo de aprendizagem. A medida de desempenho da rede será a raiz do erro quadrático médio (RMSE – Root Mean Squared Error) dada por

$$RMSE = \hat{\varepsilon} = \sqrt{\frac{\sum_{\mu} \sum_{o} (d_3 - y_3)^2}{N_o N_e}}, \quad (50)$$

$N_e$  é o número total de padrões em uma época e  $N_o$  é o número de neurônios na camada de saída. A soma em (50) é realizada sobre os padrões  $\mu$  e os neurônios de saída  $o$ ,  $d_3$  é o valor alvo e  $y_3$  é a saída observada da rede. Nesse caso, vamos analisar o comportamento do RMSE à medida que o número épocas aumenta. Dessa forma, procede-se a aplicação do algoritmo de backpropagation.

## Primeira Época

**Primeira entrada,**  $x_1 = 1$  e  $x_2 = 1$ , saída desejada  $d_3 = 0$ .

Cálculo da saída do Neurônio 1

$$v_1 = 1 \cdot b_1 + x_1 \cdot w_{11} + x_2 \cdot w_{12},$$

$$v_1 = 1 \cdot 0 + x_1 \cdot 0 + x_2 \cdot 0,$$

$$v_1 = 0,$$

$$y_1 = \varphi(v_1) = \frac{1}{1 + \exp(-v_1)},$$

$$y_1 = \varphi(v_1) = \frac{1}{1 + \exp(0)},$$

$$y_1 = \frac{1}{2} = 0,5.$$

Cálculo da saída do Neurônio 2

$$v_2 = 1 \cdot b_2 + x_1 \cdot w_{21} + x_2 \cdot w_{22},$$

$$v_2 = 1 \cdot 0 + x_1 \cdot 0 + x_2 \cdot 0,$$

$$v_2 = 0,$$

$$y_2 = \varphi(v_2) = \frac{1}{1 + \exp(-v_2)},$$

$$y_2 = \varphi(v_2) = \frac{1}{1 + \exp(0)},$$

$$y_2 = \frac{1}{2} = 0,5.$$

Cálculo da saída do Neurônio 3

$$v_3 = 1 \cdot b_3 + y_1 \cdot w_{31} + y_2 \cdot w_{32},$$

$$v_3 = 1 \cdot 0 + y_1 \cdot 0 + y_2 \cdot 0,$$

$$v_3 = 0,$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(-v_3)},$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(0)},$$

$$y_3 = \frac{1}{2} = 0,5.$$

Cálculo do erro na saída da rede para a primeira entrada

$$e_1 = d_3 - y_3,$$

$$e_1 = 0 - 0,5,$$

$$e_1 = -0,5.$$

Cálculo do gradiente local no Neurônio 3

$$\delta_{o1} = \varphi'(v_3) \cdot (d_3 - y_3),$$

$$\delta_{o1} = \varphi'(0) \cdot (-0,5),$$

$$\delta_{o1} = \varphi(0) \cdot [1 - \varphi(0)] \cdot (-0,5),$$

$$\delta_{o1} = \frac{1}{2} \cdot \left[ 1 - \frac{1}{2} \right] \cdot (-0,5),$$

$$\delta_{o1} = -0,125.$$

Cálculo do gradiente local no Neurônio 1

$$\delta_{h1} = \varphi'(v_1) \cdot (\delta_{o1} \cdot w_{31}),$$

$$\delta_{h1} = \varphi(v_1) \cdot [1 - \varphi(v_1)] \cdot (\delta_{o1} \cdot w_{31}),$$

$$\delta_{h1} = \frac{1}{2} \cdot \left[ 1 - \frac{1}{2} \right] \cdot (-0,125 \cdot 0),$$

$$\delta_{h1} = 0.$$



## Cálculo do gradiente local no Neurônio 2

$$\begin{aligned}\delta_{h2} &= \varphi'(v_2) \cdot (\delta_{o1} \cdot w_{32}), \\ \delta_{h2} &= \varphi(v_2) \cdot [1 - \varphi(v_2)] \cdot (\delta_{o1} \cdot w_{32}), \\ \delta_{h2} &= \frac{1}{2} \cdot \left[1 - \frac{1}{2}\right] \cdot (-0,125.0), \\ \delta_{h2} &= 0.\end{aligned}$$

Ajuste dos pesos utilizando a regra de aprendizagem

$$\begin{aligned}w_{31}(n+1) &= w_{31}(n) + \eta \cdot \delta_{o1}(n) \cdot y_1 + \alpha \cdot w_{31}(n-1), \\ w_{31}(n+1) &= 0 + 0,1 \cdot (-0,125) \cdot 0,5 + 0,5 \cdot 0, \\ w_{31}(n+1) &= -0,00625.\end{aligned}$$

$$\begin{aligned}w_{32}(n+1) &= w_{32}(n) + \eta \cdot \delta_{o1}(n) \cdot y_2 + \alpha \cdot w_{32}(n-1), \\ w_{32}(n+1) &= 0 + 0,1 \cdot (-0,125) \cdot 0,5 + 0,5 \cdot 0, \\ w_{32}(n+1) &= -0,00625.\end{aligned}$$

$$\begin{aligned}w_{11}(n+1) &= w_{11}(n) + \eta \cdot \delta_{h1}(n) \cdot x_1 + \alpha \cdot w_{11}(n-1), \\ w_{11}(n+1) &= 0 + 0,1 \cdot 0,1 + 0,5 \cdot 0, \\ w_{11}(n+1) &= 0.\end{aligned}$$

$$w_{21}(n+1) = w_{21}(n) + \eta \cdot \delta_{h2}(n) \cdot x_1 + \alpha \cdot w_{21}(n-1),$$

$$w_{21}(n+1) = 0 + 0,1 \cdot 0,1 + 0,5 \cdot 0,$$

$$w_{21}(n+1) = 0.$$

$$w_{12}(n+1) = w_{12}(n) + \eta \cdot \delta_{h1}(n) \cdot x_2 + \alpha \cdot w_{12}(n-1),$$

$$w_{12}(n+1) = 0 + 0,1 \cdot 0,1 + 0,5 \cdot 0,$$

$$w_{12}(n+1) = 0.$$

$$w_{22}(n+1) = w_{22}(n) + \eta \cdot \delta_{h2}(n) \cdot x_2 + \alpha \cdot w_{22}(n-1),$$

$$w_{22}(n+1) = 0 + 0,1 \cdot 0,1 + 0,5 \cdot 0,$$

$$w_{22}(n+1) = 0.$$

Ajuste dos bias utilizando a regra de aprendizagem

$$b_3(n+1) = b_3(n) + \eta \cdot \delta_{o1}(n) \cdot 1 + \alpha \cdot b_3(n-1),$$

$$b_3(n+1) = 0 + 0,1 \cdot (-0,125) \cdot 1 + 0,5 \cdot 0,$$

$$b_3(n+1) = -0,0125.$$



$$b_1(n+1) = b_1(n) + \eta \cdot \delta_{h1}(n) \cdot 1 + \alpha \cdot b_1(n-1),$$

$$b_1(n+1) = 0 + 0,1 \cdot 0,1 + 0,5 \cdot 0,$$

$$b_1(n+1) = 0.$$

$$b_2(n+1) = b_2(n) + \eta \cdot \delta_{h2}(n) \cdot 1 + \alpha \cdot b_2(n-1),$$

$$b_2(n+1) = 0 + 0,1 \cdot 0,1 + 0,5 \cdot 0,$$

$$b_2(n+1) = 0.$$

Após a apresentação do primeiro padrão de entrada, os pesos e bias foram atualizados de tal forma que agora correspondem a

$$w_{11} = 0,$$

$$w_{21} = 0,$$

$$w_{12} = 0,$$

$$w_{22} = 0,$$

$$w_{31} = -0,00625,$$

$$w_{32} = -0,00625,$$

$$b_1 = 0,$$

$$b_2 = 0,$$

$$b_3 = -0,0125.$$

**Segunda entrada**,  $x_1 = 1$  e  $x_2 = 0$ , saída desejada  $d_3 = 1$ .

Cálculo da saída do Neurônio 1

$$v_1 = 1 \cdot b_1 + x_1 \cdot w_{11} + x_2 \cdot w_{12},$$

$$v_1 = 1.0 + 1.0 + 0.0,$$

$$v_1 = 0,$$

$$y_1 = \varphi(v_1) = \frac{1}{1 + \exp(-v_1)},$$

$$y_1 = \varphi(v_1) = \frac{1}{1 + \exp(0)},$$

$$y_1 = \frac{1}{2} = 0,5.$$

Cálculo da saída do Neurônio 2

$$v_2 = 1 \cdot b_2 + x_1 \cdot w_{21} + x_2 \cdot w_{22},$$

$$v_2 = 1.0 + 1.0 + 0.0,$$

$$v_2 = 0,$$

$$y_2 = \varphi(v_2) = \frac{1}{1 + \exp(-v_2)},$$

$$y_2 = \varphi(v_2) = \frac{1}{1 + \exp(0)},$$

$$y_2 = \frac{1}{2} = 0,5.$$

Cálculo da saída do Neurônio 3

$$v_3 = 1 \cdot b_3 + y_1 \cdot w_{31} + y_2 \cdot w_{32},$$

$$v_3 = 1 \cdot (-0,0125) + 0,5 \cdot (-0,00625) + 0,5 \cdot (-0,00625),$$

$$v_3 = -0,01875,$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(-v_3)},$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(0,01875)},$$

$$y_3 = 0,495313.$$

Cálculo do erro na saída da rede para a segunda entrada

$$e_2 = d_3 - y_3,$$

$$e_2 = 1 - 0,495313,$$

$$e_2 = 0,504687.$$



### Cálculo do gradiente local no Neurônio 3

$$\begin{aligned}\delta_{o1} &= \varphi'(v_3). (d_3 - y_3), \\ \delta_{o1} &= \varphi'(-0,01875). (0,504687), \\ \delta_{o1} &= \varphi(-0,01875). [1 - \varphi(-0,01875)]. (0,504687), \\ \delta_{o1} &= 0,495313. [1 - 0,495313]. (0,504687), \\ \delta_{o1} &= 0,126161.\end{aligned}$$

### Cálculo do gradiente local no Neurônio 1

$$\begin{aligned}\delta_{h1} &= \varphi'(v_1). (\delta_{o1}. w_{31}), \\ \delta_{h1} &= \varphi(v_1). [1 - \varphi(v_1)]. (\delta_{o1}. w_{31}), \\ \delta_{h1} &= \frac{1}{2}. \left[1 - \frac{1}{2}\right]. [0,126161. (-0,00625)], \\ \delta_{h1} &= -0,000197.\end{aligned}$$

### Cálculo do gradiente local no Neurônio 2

$$\begin{aligned}\delta_{h2} &= \varphi'(v_2). (\delta_{o1}. w_{32}), \\ \delta_{h2} &= \varphi(v_2). [1 - \varphi(v_2)]. (\delta_{o1}. w_{32}), \\ \delta_{h2} &= \frac{1}{2}. \left[1 - \frac{1}{2}\right]. [0,126161. (-0,00625)], \\ \delta_{h2} &= -0,000197.\end{aligned}$$

Ajuste dos pesos utilizando a regra de aprendizagem

$$\begin{aligned} w_{31}(n+1) &= w_{31}(n) + \eta \cdot \delta_{o1}(n) \cdot y_1 + \alpha \cdot w_{31}(n-1), \\ w_{31}(n+1) &= -0,00625 + 0,1 \cdot 0,126161 \cdot 0,5 + 0,5 \cdot 0, \\ w_{31}(n+1) &= 0,000058. \end{aligned}$$

Verifique que  $w_{31}(n)$  é o valor de que dispomos desse peso,  $w_{31}(n-1)$  é o valor da etapa anterior que era zero e  $w_{31}(n+1)$  é o novo valor que estamos calculando. Continuando para os próximos pesos temos que

$$\begin{aligned} w_{32}(n+1) &= w_{32}(n) + \eta \cdot \delta_{o1}(n) \cdot y_2 + \alpha \cdot w_{32}(n-1), \\ w_{32}(n+1) &= -0,00625 + 0,1 \cdot 0,126161 \cdot 0,5 + 0,5 \cdot 0, \\ w_{32}(n+1) &= 0,000058. \end{aligned}$$

$$\begin{aligned} w_{11}(n+1) &= w_{11}(n) + \eta \cdot \delta_{h1}(n) \cdot x_1 + \alpha \cdot w_{11}(n-1), \\ w_{11}(n+1) &= 0 + 0,1 \cdot (-0,000197) \cdot 1 + 0,5 \cdot 0, \\ w_{11}(n+1) &= 0,000058. \end{aligned}$$

$$w_{11}(n+1) = -0,0000197.$$

$$w_{21}(n+1) = w_{21}(n) + \eta \cdot \delta_{h2}(n) \cdot x_1 + \alpha \cdot w_{21}(n-1),$$

$$w_{21}(n+1) = 0 + 0,1.(-0,000197).1 + 0,5.0,$$

$$w_{21}(n+1) = -0,0000197.$$

$$w_{12}(n+1) = w_{12}(n) + \eta \cdot \delta_{h1}(n) \cdot x_2 + \alpha \cdot w_{12}(n-1),$$

$$w_{12}(n+1) = 0 + 0,1.(-0,000197).0 + 0,5.0,$$

$$w_{12}(n+1) = 0.$$

$$w_{22}(n+1) = w_{22}(n) + \eta \cdot \delta_{h2}(n) \cdot x_2 + \alpha \cdot w_{22}(n-1),$$

$$w_{22}(n+1) = 0 + 0,1.(-0,000197).0 + 0,5.0,$$

$$w_{22}(n+1) = 0.$$

Ajuste dos bias utilizando a regra de aprendizagem

$$b_3(n+1) = b_3(n) + \eta \cdot \delta_{o1}(n) \cdot 1 + \alpha \cdot b_3(n-1),$$

$$b_3(n+1) = -0,0125 + 0,1 \cdot 0,126161 \cdot 1 + 0,5 \cdot 0,$$

$$b_3(n+1) = 0,000116.$$

$$b_1(n+1) = b_1(n) + \eta \cdot \delta_{h1}(n) \cdot 1 + \alpha \cdot b_1(n-1),$$

$$b_1(n+1) = 0 + 0,1 \cdot (-0,000197) \cdot 1 + 0,5 \cdot 0,$$

$$b_1(n+1) = -0,0000197.$$

$$b_2(n+1) = b_2(n) + \eta \cdot \delta_{h2}(n) \cdot 1 + \alpha \cdot b_2(n-1),$$

$$b_2(n+1) = 0 + 0,1 \cdot (-0,000197) \cdot 1 + 0,5 \cdot 0,$$

$$b_2(n+1) = -0,0000197.$$



Após a apresentação do segundo padrão de entrada, os pesos e bias foram atualizados de tal forma que agora correspondem a

$$w_{11} = -0,0000197,$$

$$w_{21} = -0,0000197,$$

$$w_{12} = 0,$$

$$w_{22} = 0,$$

$$w_{31} = 0,000058,$$

$$w_{32} = 0,000058,$$

$$b_1 = -0,0000197,$$

$$b_2 = -0,0000197,$$

$$b_3 = 0,000116.$$

**Terceira entrada,**  $x_1 = 0$  e  $x_2 = 1$ , saída desejada  $d_3 = 1$ .

Cálculo da saída do Neurônio 1

$$v_1 = 1.b_1 + x_1.w_{11} + x_2.w_{12},$$

$$v_1 = 1.(-0,0000197) + 0.(-0,0000197) + 1.0,$$

$$v_1 = -0,0000197,$$

$$y_1 = \varphi(v_1) = \frac{1}{1 + \exp(-v_1)},$$

$$y_1 = \varphi(v_1) = \frac{1}{1 + \exp(0,0000197)},$$

$$y_1 = 0,499995.$$

Cálculo da saída do Neurônio 2

$$v_2 = 1.b_2 + x_1.w_{21} + x_2.w_{22},$$

$$v_2 = 1.(-0,0000197) + 0.(-0,0000197) + 1.0,$$

$$v_2 = -0,0000197,$$

$$y_2 = \varphi(v_2) = \frac{1}{1 + \exp(-v_2)},$$

$$y_2 = \varphi(v_2) = \frac{1}{1 + \exp(0,0000197)},$$

$$y_2 = 0,499995.$$



### Cálculo da saída do Neurônio 3

$$v_3 = 1 \cdot b_3 + y_1 \cdot w_{31} + y_2 \cdot w_{32},$$

$$v_3 = 1 \cdot (0,000116) + 0,499995 \cdot (0,000058) + 0,499995 \cdot (0,000058),$$

$$v_3 = 0,000174,$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(-v_3)},$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(-0,000174)},$$

$$y_3 = 0,5000435.$$

### Cálculo do erro na saída da rede para a terceira entrada

$$e_3 = d_3 - y_3,$$

$$e_3 = 1 - 0,5000435,$$

$$e_3 = 0,499957.$$

### Cálculo do gradiente local no Neurônio 3

$$\begin{aligned}\delta_{o1} &= \varphi'(v_3) \cdot (d_3 - y_3), \\ \delta_{o1} &= \varphi'(0,000174) \cdot (0,499957), \\ \delta_{o1} &= \varphi(0,000174) \cdot [1 - \varphi(0,000174)] \cdot (0,499957), \\ \delta_{o1} &= 0,5000435 \cdot [1 - 0,5000435] \cdot (0,499957), \\ \delta_{o1} &= 0,1249891.\end{aligned}$$

### Cálculo do gradiente local no Neurônio 1

$$\begin{aligned}\delta_{h1} &= \varphi'(v_1) \cdot (\delta_{o1} \cdot w_{31}), \\ \delta_{h1} &= \varphi(v_1) \cdot [1 - \varphi(v_1)] \cdot (\delta_{o1} \cdot w_{31}), \\ \delta_{h1} &= 0,499995 \cdot [1 - 0,499995] \cdot [0,1249891 \cdot 0,000058], \\ \delta_{h1} &= 0,0000018.\end{aligned}$$

### Cálculo do gradiente local no Neurônio 2

$$\begin{aligned}\delta_{h2} &= \varphi'(v_2) \cdot (\delta_{o1} \cdot w_{32}), \\ \delta_{h2} &= \varphi(v_2) \cdot [1 - \varphi(v_2)] \cdot (\delta_{o1} \cdot w_{32}), \\ \delta_{h2} &= 0,499995 \cdot [1 - 0,499995] \cdot [0,1249891 \cdot 0,000058], \\ \delta_{h2} &= 0,0000018.\end{aligned}$$



Ajuste dos pesos utilizando a regra de aprendizagem

$$w_{31}(n+1) = w_{31}(n) + \eta \cdot \delta_{o1}(n) \cdot y_1 + \alpha \cdot w_{31}(n-1),$$

$$w_{31}(n+1) = 0,000058 + 0,1 \cdot (0,1249891) \cdot 0,499995 + 0,5 \cdot (-0,00625),$$

$$w_{31}(n+1) = 0,003182.$$

Verifique que  $w_{31}(n)$  é o valor de que dispomos desse peso,  $w_{31}(n-1)$  é o valor da etapa anterior que era -0,00625 e  $w_{31}(n+1)$  é o novo valor que estamos calculando. Continuando para os próximos pesos, temos que

$$w_{32}(n+1) = w_{32}(n) + \eta \cdot \delta_{o1}(n) \cdot y_2 + \alpha \cdot w_{32}(n-1),$$

$$w_{32}(n+1) = 0,000058 + 0,1 \cdot (0,1249891) \cdot 0,499995 + 0,5 \cdot (-0,00625),$$

$$w_{32}(n+1) = 0,003182.$$

$$\begin{aligned}w_{11}(n+1) &= w_{11}(n) + \eta \cdot \delta_{h1}(n) \cdot x_1 + \alpha \cdot w_{11}(n-1), \\w_{11}(n+1) &= -0,0000197 + 0,1 \cdot 0,0000018 \cdot 0 + 0,5 \cdot 0, \\w_{11}(n+1) &= -0,0000197.\end{aligned}$$

$$\begin{aligned}w_{21}(n+1) &= w_{21}(n) + \eta \cdot \delta_{h2}(n) \cdot x_1 + \alpha \cdot w_{21}(n-1), \\w_{21}(n+1) &= -0,0000197 + 0,1 \cdot 0,0000018 \cdot 0 + 0,5 \cdot 0, \\w_{21}(n+1) &= -0,0000197.\end{aligned}$$

$$w_{12}(n+1) = w_{12}(n) + \eta \cdot \delta_{h1}(n) \cdot x_2 + \alpha \cdot w_{12}(n-1),$$

$$\begin{aligned}w_{12}(n+1) &= 0 + 0,1 \cdot 0,0000018 \cdot 1 + 0,5 \cdot 0, \\w_{12}(n+1) &= 0,00000018.\end{aligned}$$

$$\begin{aligned}w_{22}(n+1) &= w_{22}(n) + \eta \cdot \delta_{h2}(n) \cdot x_2 + \alpha \cdot w_{22}(n-1), \\w_{22}(n+1) &= 0 + 0,1 \cdot 0,0000018 \cdot 1 + 0,5 \cdot 0, \\w_{22}(n+1) &= 0,00000018.\end{aligned}$$

Ajuste dos bias utilizando a regra de aprendizagem

$$b_3(n+1) = b_3(n) + \eta \cdot \delta_{o1}(n) \cdot 1 + \alpha \cdot b_3(n-1),$$

$$b_3(n+1) = 0,000116 + 0,1 \cdot 0,1249891 \cdot 1 + 0,5 \cdot (-0,0125),$$

$$b_3(n+1) = 0,006365.$$

$$b_1(n+1) = b_1(n) + \eta \cdot \delta_{h1}(n) \cdot 1 + \alpha \cdot b_1(n-1),$$

$$b_1(n+1) = -0,0000197 + 0,1 \cdot 0,0000018 \cdot 1 + 0,5 \cdot 0,$$

$$b_1(n+1) = -0,00001953.$$

$$b_2(n+1) = b_2(n) + \eta \cdot \delta_{h2}(n) \cdot 1 + \alpha \cdot b_2(n-1),$$

$$b_2(n+1) = -0,0000197 + 0,1 \cdot 0,0000018 \cdot 1 + 0,5 \cdot 0,$$

$$b_2(n+1) = -0,00001953.$$

Após a apresentação do terceiro padrão de entrada, os pesos e bias foram atualizados de tal forma que agora correspondem a

$$w_{11} = -0,0000197,$$

$$w_{21} = -0,0000197,$$

$$w_{12} = 0,00000018,$$

$$w_{22} = 0,00000018,$$

$$w_{31} = 0,003182,$$

$$w_{32} = 0,003182,$$

$$b_1 = -0,00001953,$$

$$b_2 = -0,00001953,$$

$$b_3 = 0,006365.$$



**Quarta entrada,**  $x_1 = 0$  e  $x_2 = 0$ , saída desejada  $d_3 = 0$ .

Cálculo da saída do Neurônio 1

$$\begin{aligned}v_1 &= 1 \cdot b_1 + x_1 \cdot w_{11} + x_2 \cdot w_{12}, \\v_1 &= 1 \cdot (-0,00001953) + 0 \cdot (-0,0000197) + 0 \cdot (0,00000018), \\v_1 &= -0,00001953, \\y_1 &= \varphi(v_1) = \frac{1}{1 + \exp(-v_1)}, \\y_1 &= \varphi(v_1) = \frac{1}{1 + \exp(0,00001952)}, \\y_1 &= 0,499995.\end{aligned}$$

Cálculo da saída do Neurônio 2

$$\begin{aligned}v_2 &= 1 \cdot b_2 + x_1 \cdot w_{21} + x_2 \cdot w_{22}, \\v_2 &= 1 \cdot (-0,00001953) + 0 \cdot (-0,0000197) + 0 \cdot (0,00000018), \\v_2 &= -0,00001953, \\y_2 &= \varphi(v_2) = \frac{1}{1 + \exp(-v_2)}, \\y_2 &= \varphi(v_2) = \frac{1}{1 + \exp(0,00001952)}, \\y_2 &= 0,499995.\end{aligned}$$

### Cálculo da saída do Neurônio 3

$$v_3 = 1 \cdot b_3 + y_1 \cdot w_{31} + y_2 \cdot w_{32},$$

$$v_3 = 1 \cdot (0,006367) + 0,499995 \cdot (0,003183) + 0,499995 \cdot (0,003183),$$

$$v_3 = 0,00955,$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(-v_3)},$$

$$y_3 = \varphi(v_3) = \frac{1}{1 + \exp(-0,00955)},$$

$$y_3 = 0,502387.$$

### Cálculo do erro na saída da rede para a quarta entrada

$$e_4 = d_3 - y_3,$$

$$e_4 = 0 - 0,502387,$$

$$e_4 = -0,502387.$$

### Cálculo do gradiente local no Neurônio 3

$$\delta_{o1} = \varphi'(v_3) \cdot (d_3 - y_3),$$

$$\delta_{o1} = \varphi'(0,00955) \cdot (-0,502387),$$

$$\delta_{o1} = \varphi(0,00955) \cdot [1 - \varphi(0,00955)] \cdot (-0,502387),$$

$$\delta_{o1} = 0,502387 \cdot [1 - 0,502387] \cdot (-0,502387),$$

$$\delta_{o1} = -0,125594.$$



## Cálculo do gradiente local no Neurônio 1

$$\begin{aligned}\delta_{h1} &= \varphi'(v_1) \cdot (\delta_{o1} \cdot w_{31}), \\ \delta_{h1} &= \varphi(v_1) \cdot [1 - \varphi(v_1)] \cdot (\delta_{o1} \cdot w_{31}), \\ \delta_{h1} &= 0,499995 \cdot [1 - 0,499995] \cdot [-0,125597 \cdot 0,003182], \\ \delta_{h1} &= -0,00009992.\end{aligned}$$

## Cálculo do gradiente local no Neurônio 2

$$\begin{aligned}\delta_{h2} &= \varphi'(v_2) \cdot (\delta_{o1} \cdot w_{32}), \\ \delta_{h2} &= \varphi(v_2) \cdot [1 - \varphi(v_2)] \cdot (\delta_{o1} \cdot w_{32}), \\ \delta_{h2} &= 0,499995 \cdot [1 - 0,499995] \cdot [-0,125594 \cdot 0,003182], \\ \delta_{h2} &= -0,00009992.\end{aligned}$$

Ajuste dos pesos utilizando a regra de aprendizagem

$$w_{31}(n+1) = w_{31}(n) + \eta \cdot \delta_{o1}(n) \cdot y_1 + \alpha \cdot w_{31}(n-1),$$

$$w_{31}(n+1) = 0,003182 + 0,1 \cdot (-0,125594) \cdot 0,499995 + 0,5 \cdot (0,000058),$$

$$w_{31}(n+1) = -0,003068.$$

$$w_{32}(n+1) = w_{32}(n) + \eta \cdot \delta_{o1}(n) \cdot y_2 + \alpha \cdot w_{32}(n-1),$$

$$w_{32}(n+1) = 0,003182 + 0,1 \cdot (-0,125594) \cdot 0,499995 + 0,5 \cdot (0,000058),$$

$$w_{32}(n+1) = -0,003068.$$



$$\begin{aligned}w_{11}(n+1) &= w_{11}(n) + \eta \cdot \delta_{h1}(n) \cdot x_1 + \alpha \cdot w_{11}(n-1), \\w_{11}(n+1) &= -0,0000197 + 0,1 \cdot (-0,00009992) \cdot 0 + 0,5 \cdot (-0,0000197), \\w_{11}(n+1) &= -0,00002957.\end{aligned}$$

$$\begin{aligned}w_{21}(n+1) &= w_{21}(n) + \eta \cdot \delta_{h2}(n) \cdot x_1 + \alpha \cdot w_{21}(n-1), \\w_{21}(n+1) &= -0,0000197 + 0,1 \cdot (-0,00009992) \cdot 0 + 0,5 \cdot (-0,0000197), \\w_{21}(n+1) &= -0,00002957.\end{aligned}$$

$$\begin{aligned}w_{12}(n+1) &= w_{12}(n) + \eta \cdot \delta_{h1}(n) \cdot x_2 + \alpha \cdot w_{12}(n-1), \\w_{12}(n+1) &= 0,00000018 + 0,1 \cdot (-0,00009992) \cdot 0 + 0,5 \cdot 0, \\w_{12}(n+1) &= 0,00000018.\end{aligned}$$

$$\begin{aligned}w_{22}(n+1) &= w_{22}(n) + \eta \cdot \delta_{h2}(n) \cdot x_2 + \alpha \cdot w_{22}(n-1), \\w_{22}(n+1) &= 0,00000018 + 0,1 \cdot (-0,00009992) \cdot 0 + 0,5 \cdot 0, \\w_{22}(n+1) &= 0,00000018.\end{aligned}$$

Ajuste dos bias utilizando a regra de aprendizagem

$$b_3(n+1) = b_3(n) + \eta \cdot \delta_{o1}(n) \cdot 1 + \alpha \cdot b_3(n-1),$$

$$b_3(n+1) = 0,006365 + 0,1 \cdot (-0,125594) \cdot 1 + 0,5 \cdot (0,000116),$$

$$b_3(n+1) = -0,006136.$$

$$b_1(n+1) = b_1(n) + \eta \cdot \delta_{h1}(n) \cdot 1 + \alpha \cdot b_1(n-1),$$

$$b_1(n+1) = -0,00001953 + 0,1 \cdot (-0,00009992) \cdot 1 + 0,5 \cdot (-0,0000197),$$

$$b_1(n+1) = -0,00003938.$$

$$b_2(n+1) = b_2(n) + \eta \cdot \delta_{h2}(n) \cdot 1 + \alpha \cdot b_2(n-1),$$

$$b_2(n+1) = -0,00001953 + 0,1 \cdot (-0,00009992) \cdot 1 + 0,5 \cdot (-0,0000197),$$

$$b_2(n+1) = -0,00003938.$$



Após a apresentação do quarto padrão de entrada os pesos e bias foram atualizados de tal forma que agora correspondem a

$$w_{11} = -0,00002957,$$

$$w_{21} = -0,00002957,$$

$$w_{12} = 0,00000018,$$

$$w_{22} = 0,00000018,$$

$$w_{31} = -0,003068,$$

$$w_{32} = -0,003068,$$

$$b_1 = -0,00003938,$$

$$b_2 = -0,00003938,$$

$$b_3 = -0,006136.$$

Após o encerramento da apresentação da Primeira Época em que foram apresentados os quatro padrões de entrada para a rede MLP, calculamos o RMSE.

$$RMSE = \hat{\varepsilon} = \sqrt{\frac{\sum_{\mu} \sum_{o} (d_3 - y_3)^2}{N_o N_e}},$$

onde,

$N_e$  é o número total de padrões em uma época  $\rightarrow N_e = 4$

$N_o$  é o número de neurônios na camada de saída  $\rightarrow N_o = 1$

a soma em  $\mu$  é realizada sobre os padrões de entrada  $\rightarrow \mu = 4$

e os neurônios de saída  $o$   $\rightarrow o = 1$

Ou seja, nesse caso o RMSE é dado por

$$\hat{\varepsilon}(1) = \sqrt{\frac{e_1^2 + e_2^2 + e_3^2 + e_4^2}{4}},$$

$$\hat{\varepsilon}(1) = \sqrt{\frac{(-0,5)^2 + (0,504687)^2 + (0,499957)^2 + (-0,502387)^2}{4}},$$

$$\hat{\varepsilon}(1) = 0,501762.$$