Universidade Federal do Espírito Santo – CCA UFES

Universidade Federal do Espírito Santo
Centro de Ciências Agrárias – CCA UFES
Departamento de Computação

Perceptron de Múltiplas Camadas e Backpropagation

Redes Neurais Artificiais
Site: http://jeiks.net
E-mail: jacsonrcsilva@gmail.com

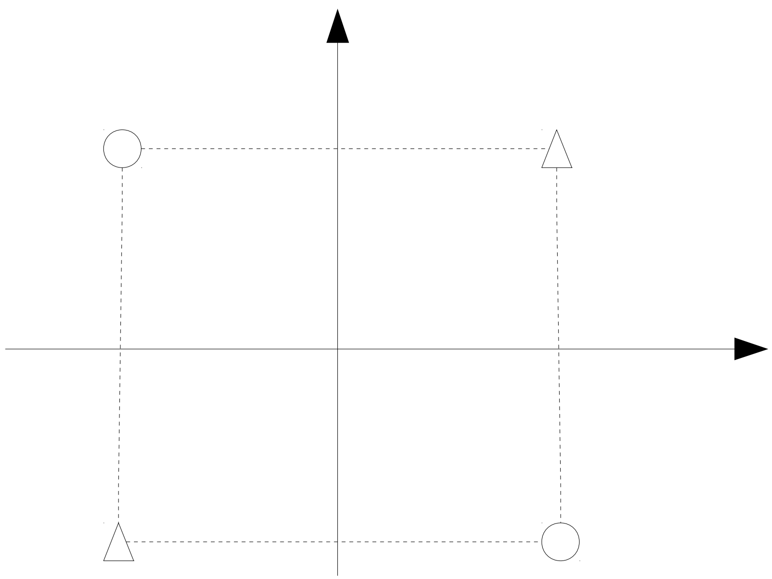
Perceptron de Múltiplas Camadas

- O Perceptron simples não consegue trabalhar com sistemas não lineares.
- Porém, o Perceptron de Múltiplas Camadas consegue efetuar o mapeamento não linear do espaço de entrada *X* para o espaço de saída *y*.
- Para resolver o problema do XOR (ou exclusivo) foi utilizado um Perceptron de uma camada intermediária (oculta) e uma função ed ativação de sinal (sgn).

MLP – XOR

- Perceptron de Múltiplas Camadas

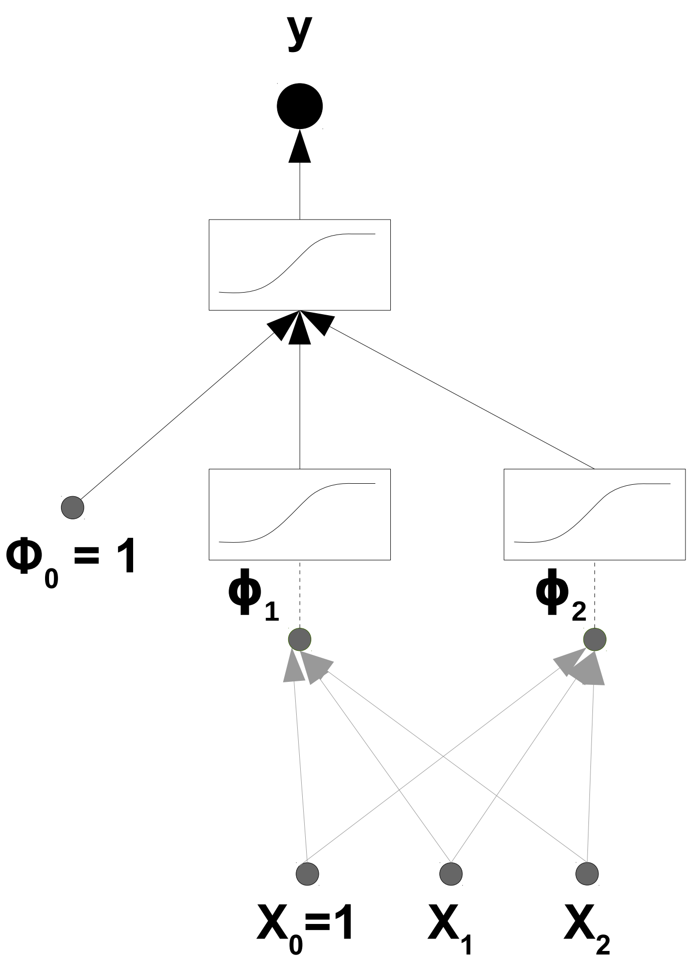
  (*MLP – Multi-Layer Perceptron*)



x_1	x_2	XOR
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

- Vamos trabalhar na implementação de uma Função Discriminativa Não-Linear usando uma camada intermediária que realiza o mapeamento não linear.

MLP – XOR



$$\Phi(x)=\begin{pmatrix}\Phi_0(\underline{x})\\ \Phi_1(\underline{x})\\ \Phi_2(\underline{x})\end{pmatrix}=\begin{pmatrix}\Phi_0(x_0;x_1;x_2)\\ \Phi_1(x_0;x_1;x_2)\\ \Phi_2(x_0;x_1;x_2)\end{pmatrix}$$

$$\Phi_0(\underline{x})=1$$

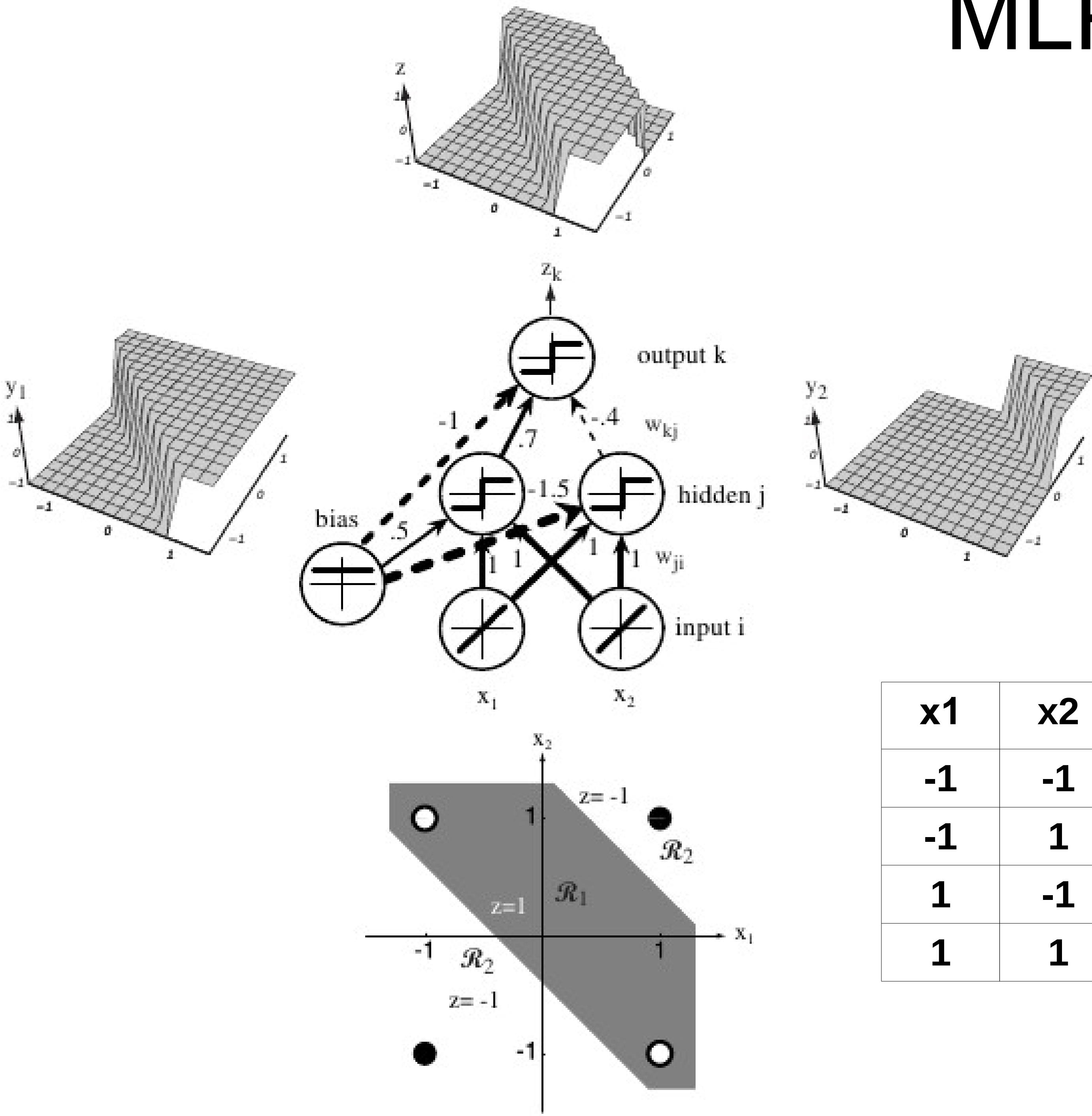
$$\Phi_1(\underline{x})=sgn\left(\frac{1}{2}.x_0+1.x_1+1.x_2\right)$$

$$\Phi_2(\underline{x})=sgn\left(-\frac{3}{2}.x_0+1.x_1+1.x_2\right)$$

$$y=sgn\left(-1.\Phi_0+0{,}7.\Phi_1(x)-0{,}4.\Phi_2(x)\right)$$

Abra o Octave, crie o vetores *X*, *y*, *y_est*, *W*, *W_y*, e teste essa RNA e suas sinapses.

MLP – XOR



x1	x2	Φ_0	Φ_1	Φ_2
-1	-1	1	-3/2	-7/2
-1	1	1	1/2	-3/2
1	-1	1	1/2	-3/2
1	1	1	5/2	1/2

Perceptron de Múltiplas Camadas

- Em geral, o perceptron com uma camada oculta calcula:

$$y_i(\underline{x}) = Z_i\left(\underline{W}_i \cdot \underline{\Phi}(\underline{x})\right) = Z_i\left(\sum_{k=0}^{H} W_{ik} \cdot \Phi_k(\underline{x})\right)$$

$$= Z_i\left(\sum_{k=0}^{H} W_{ik} \cdot F\left(W_k \cdot \underline{x}\right)\right)$$

$$= Z_i\left(\sum_{k=0}^{H} W_{ik} \cdot F\left(\sum_{j=0}^{d} W_{kj} \cdot x_j\right)\right)$$

sendo: Z função de saída e F função da camada oculta

Perceptron de Múltiplas Camadas

- No modelo totalmente conectado, a função de ativação é igual para todos os neurônios da mesma camada.
- Características das funções Z e F:
  - Deve ser não linear em pelo menos uma camada oculta;
  - Deve ser diferenciável: Z'(a) deve existir.
- A camada oculta implementa a função de base $\phi_k(\underline{x})$;
- A função base não é fixa, sendo então adaptável:
  - Poder de cálculo do MLP.

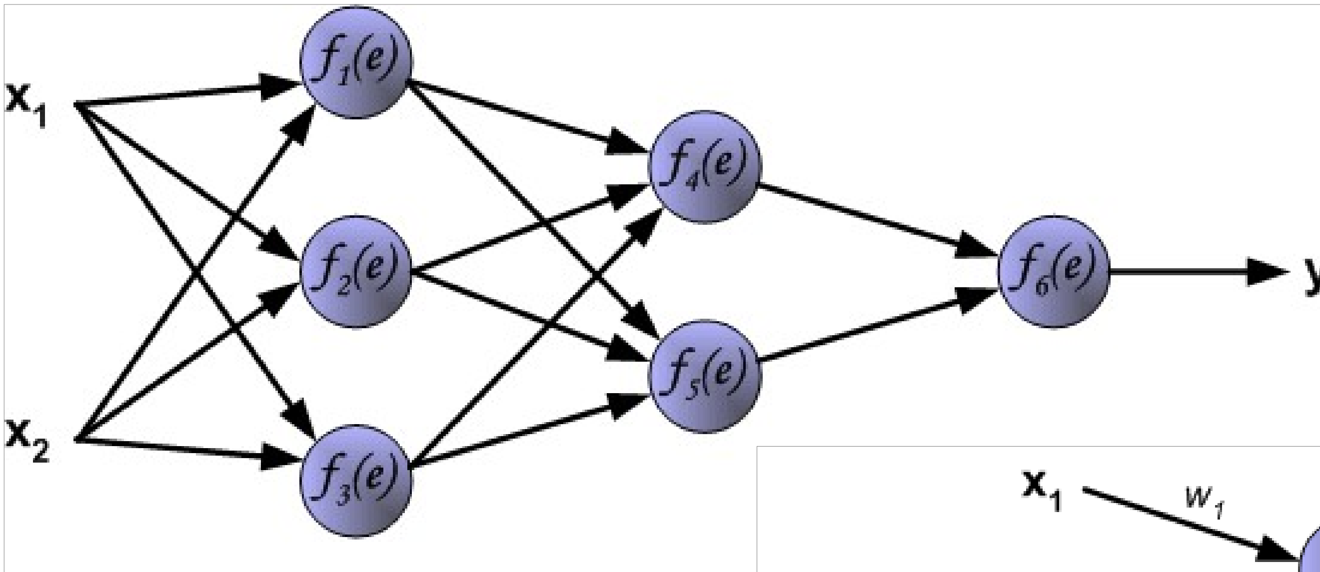
Backpropagation

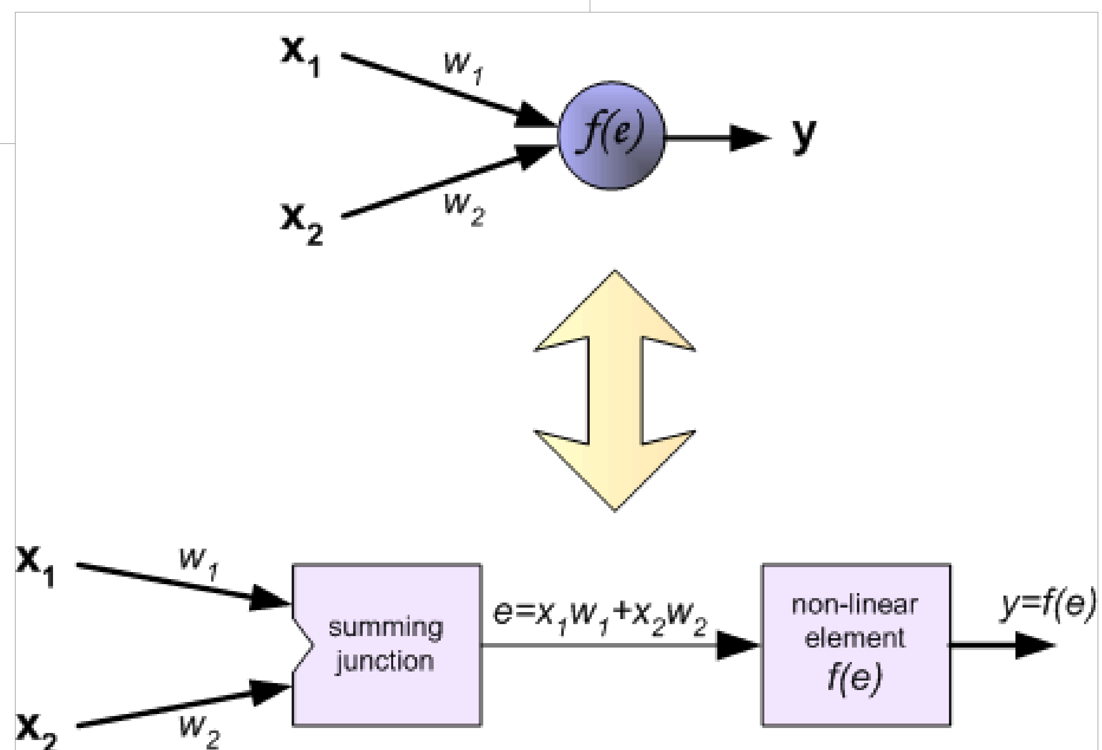
- Vamos ver o funcionamento iterativo do backpropagation utilizando a referência de

  Mariusz Bernacki, Przemysław Włodarczyk, (2005)

  http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

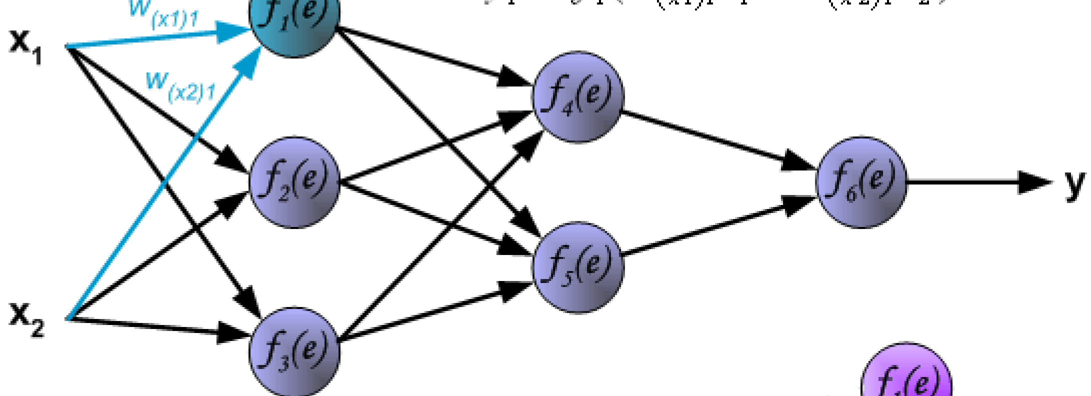
Rede Neural Utilizada



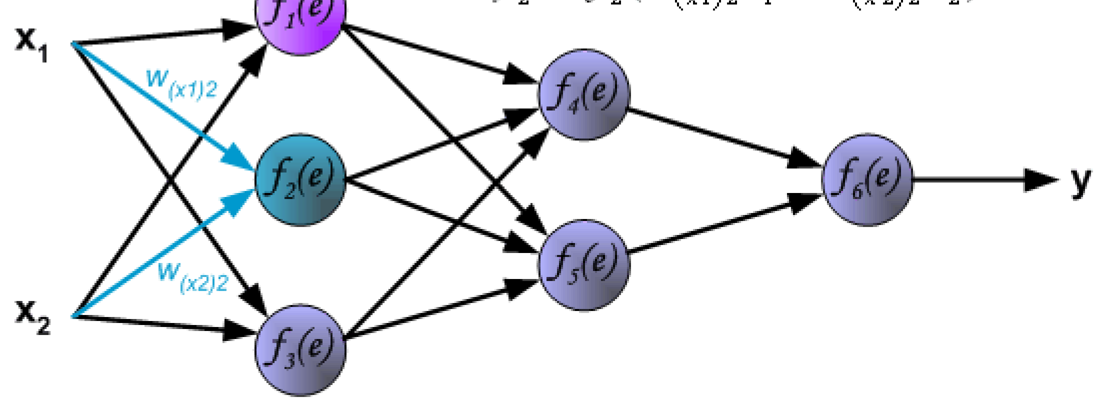


Calculando a saída desejada...

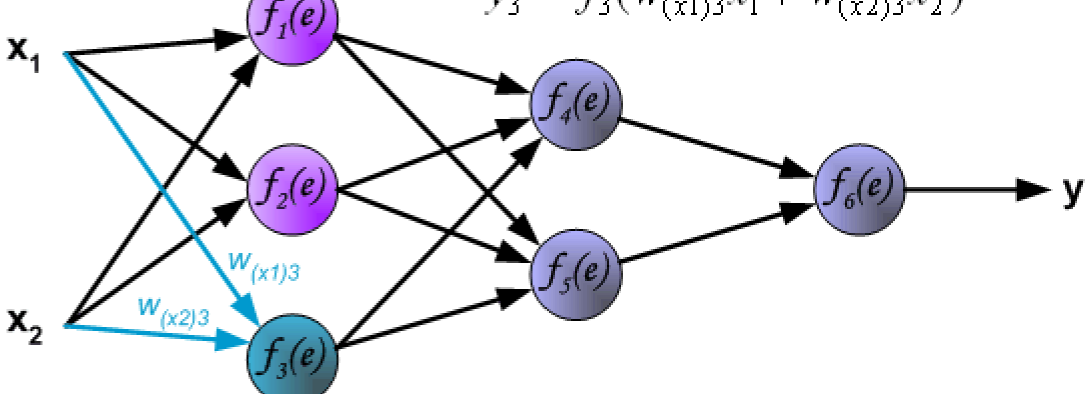
$$y_1 = f_1(w_{(x1)1}x_1 + w_{(x2)1}x_2)$$



$$y_2 = f_2(w_{(x1)2}x_1 + w_{(x2)2}x_2)$$

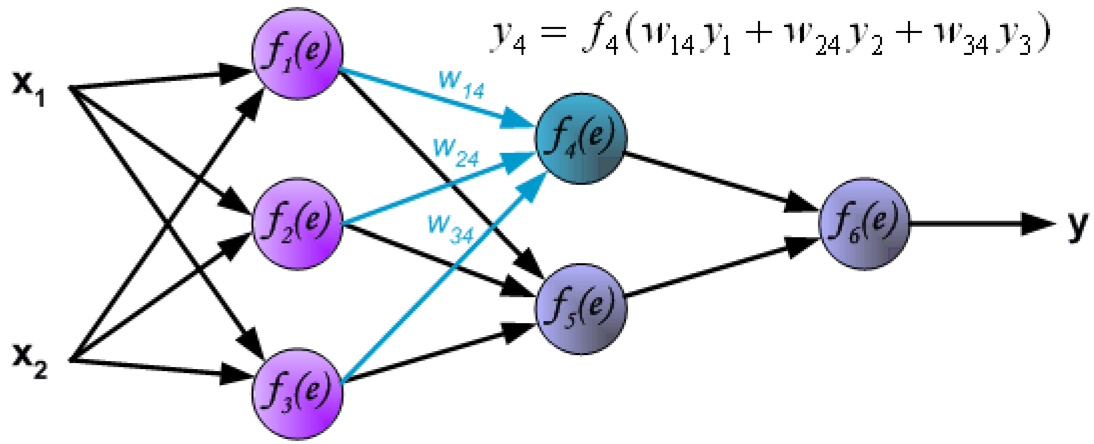


$$y_3 = f_3(w_{(x1)3}x_1 + w_{(x2)3}x_2)$$

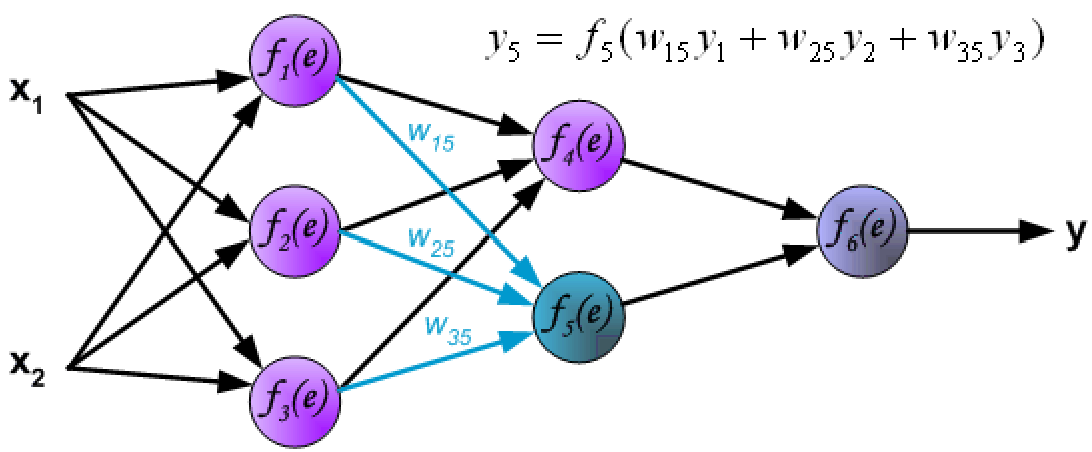


Calculando a saída desejada...

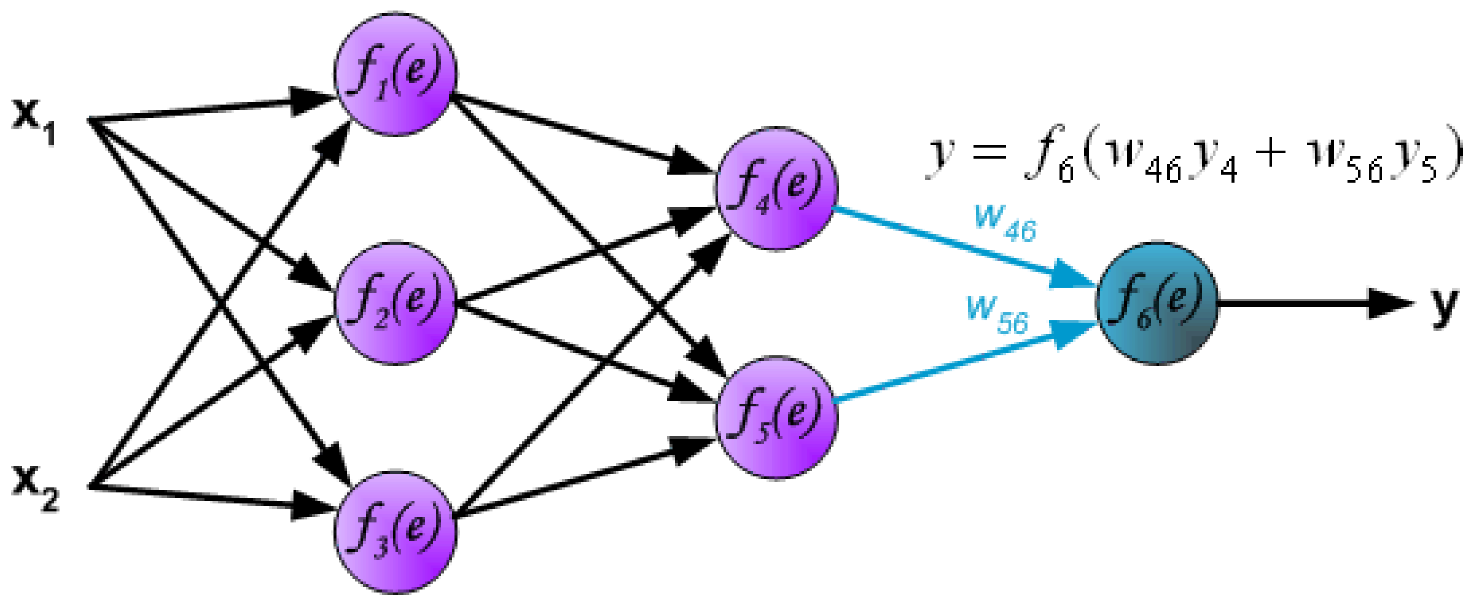
$$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$$



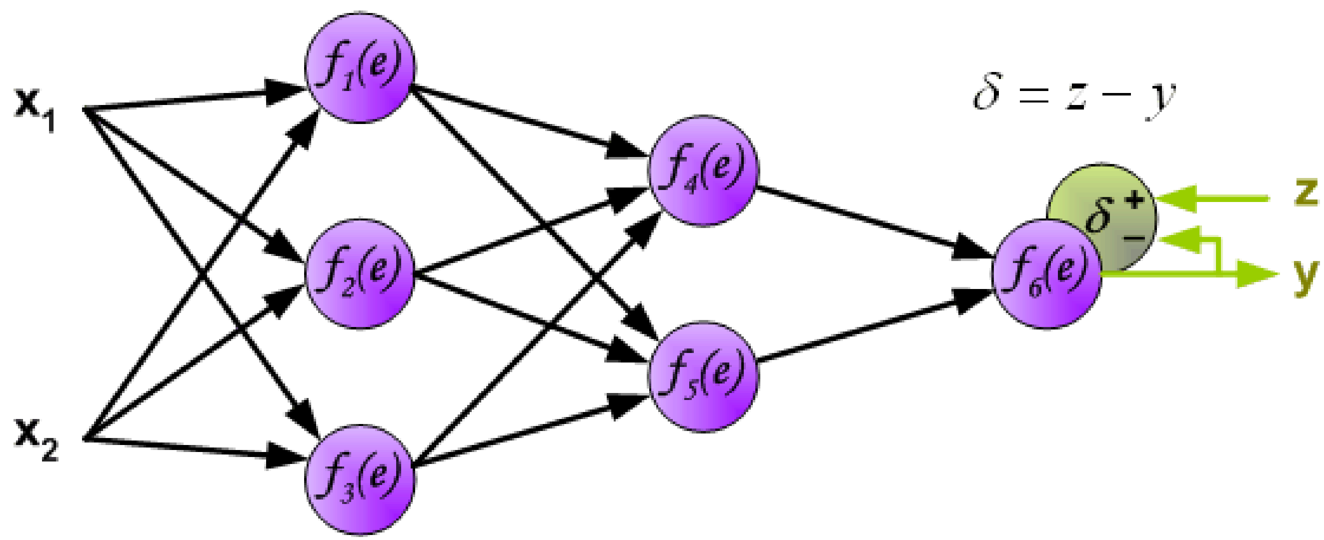
$$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$$



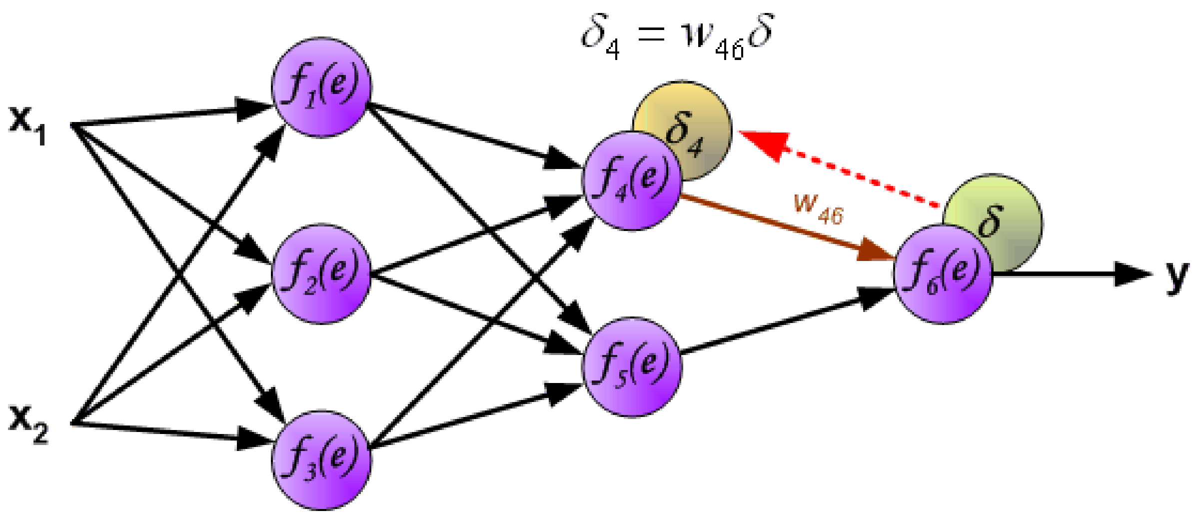
Calculando a saída desejada...

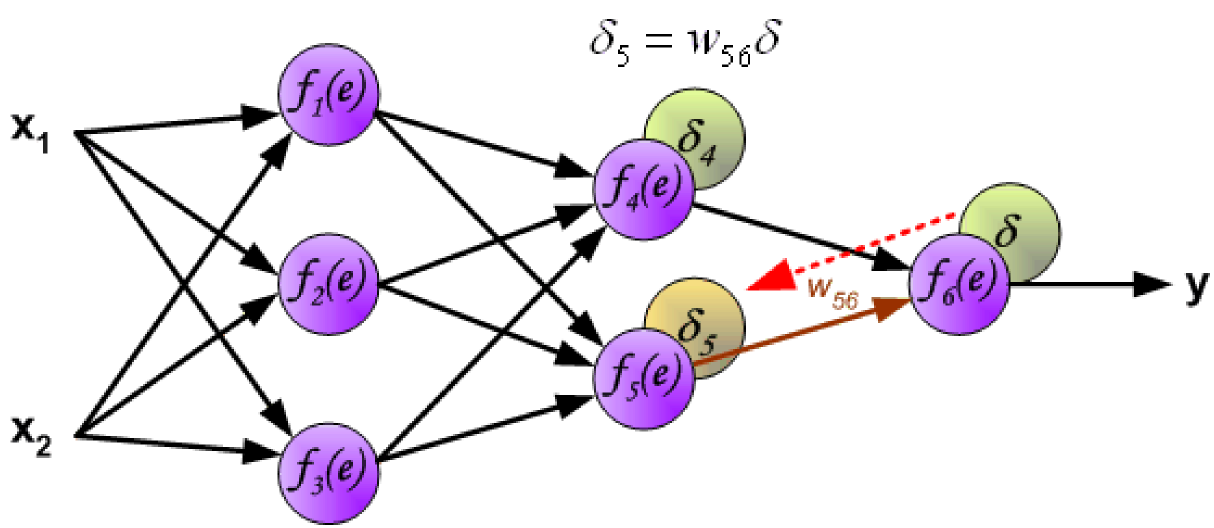


Calculando o erro da saída



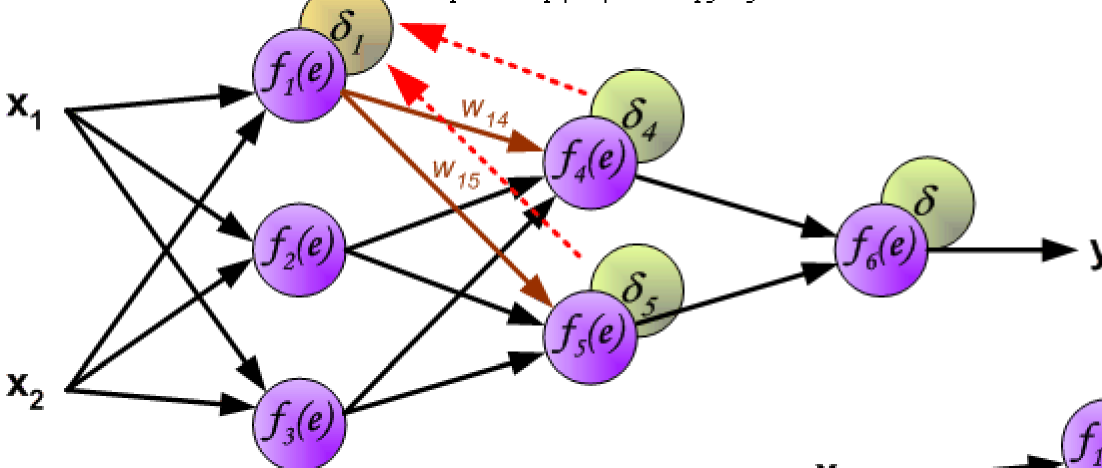
Retrocedendo com o Erro...



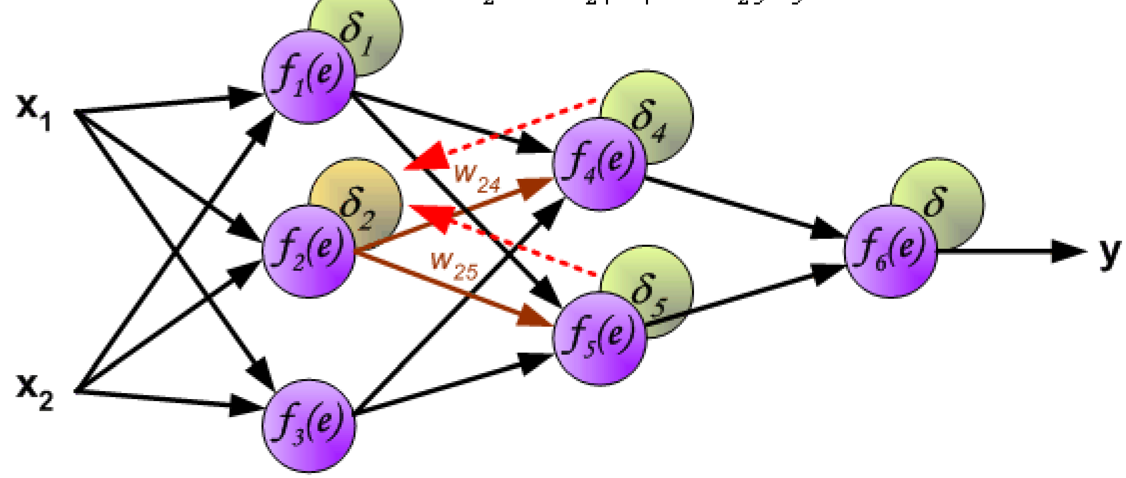


Retrocedendo com o Erro...

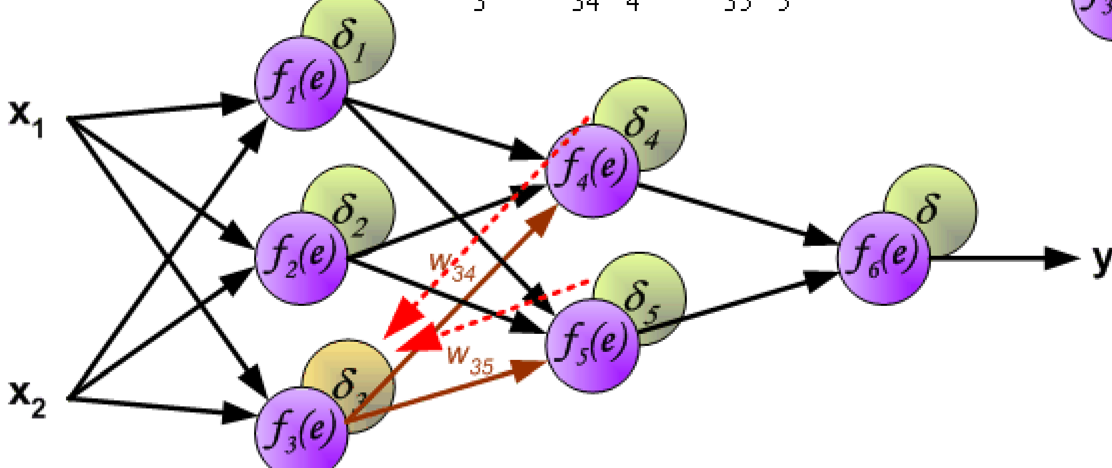
$$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$$



$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$



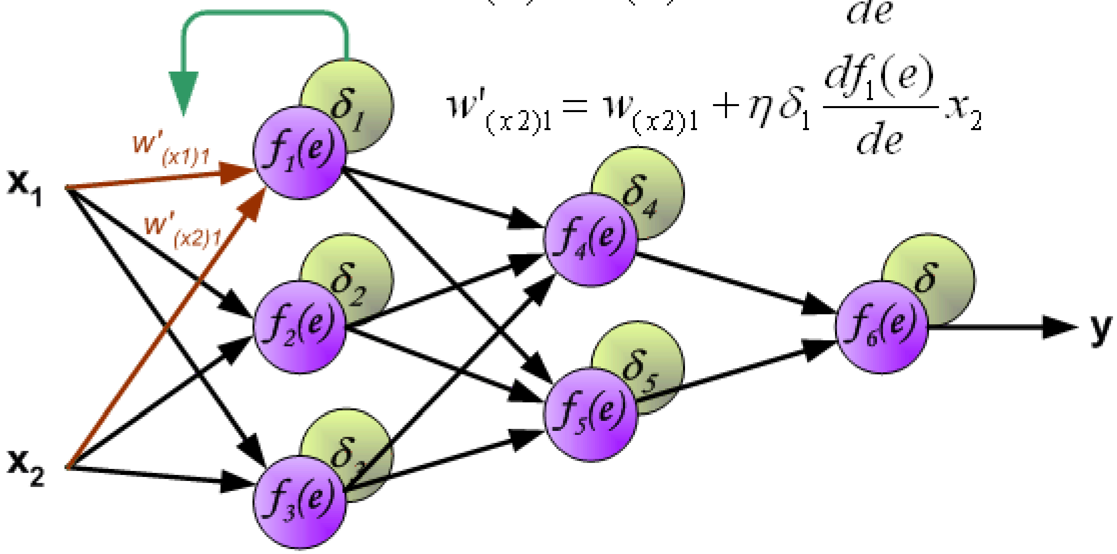
$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$



Atualizando os pesos...

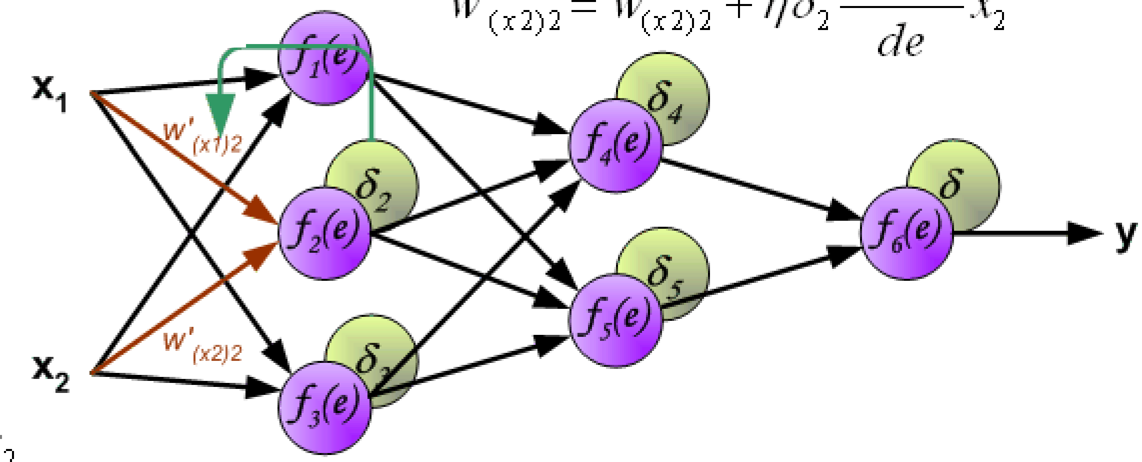
$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$



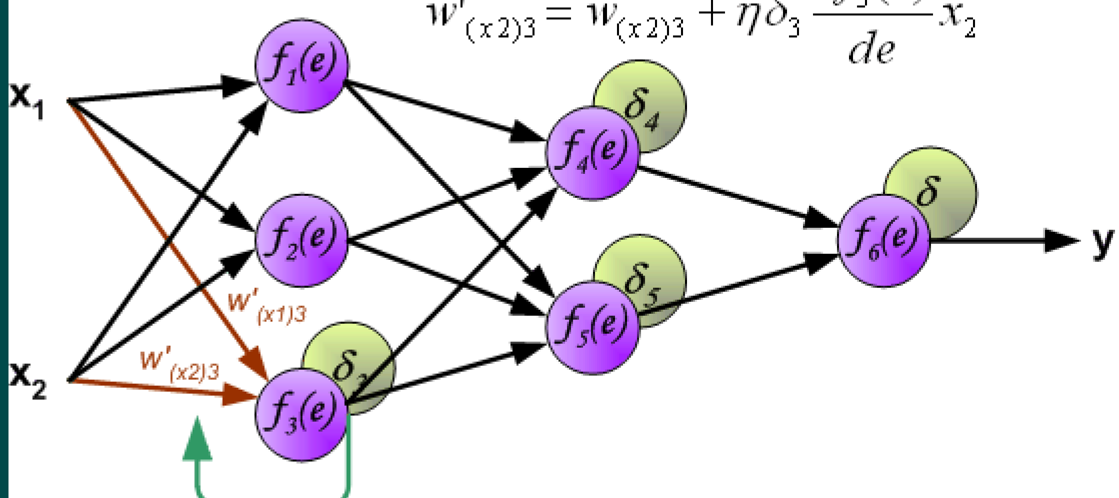
$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2$$



$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$

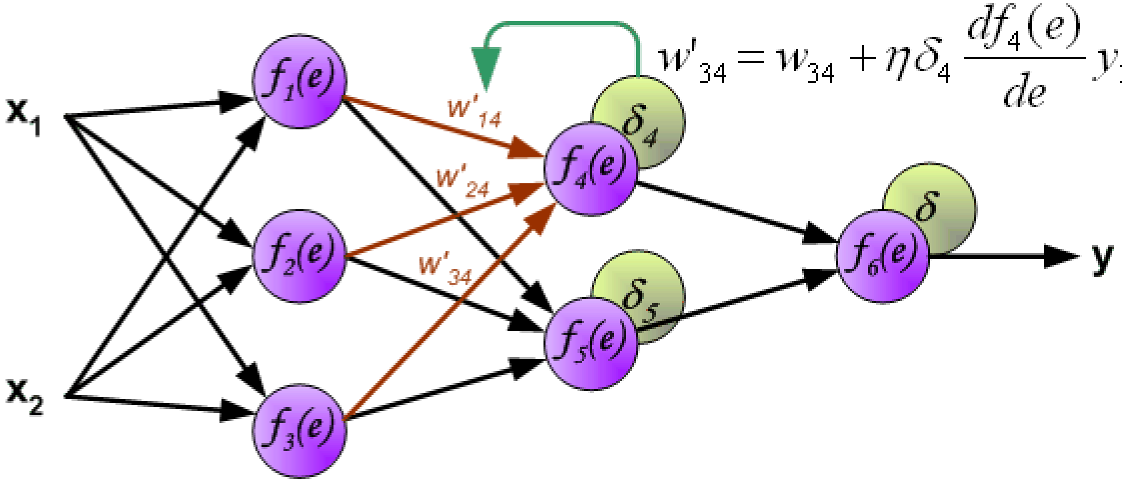


Atualizando os pesos...

$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$

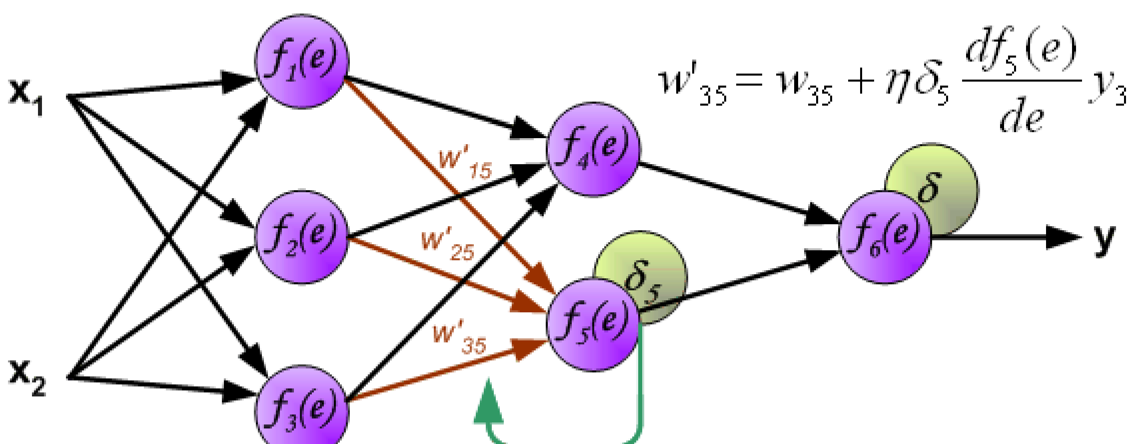
$$w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3$$



$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

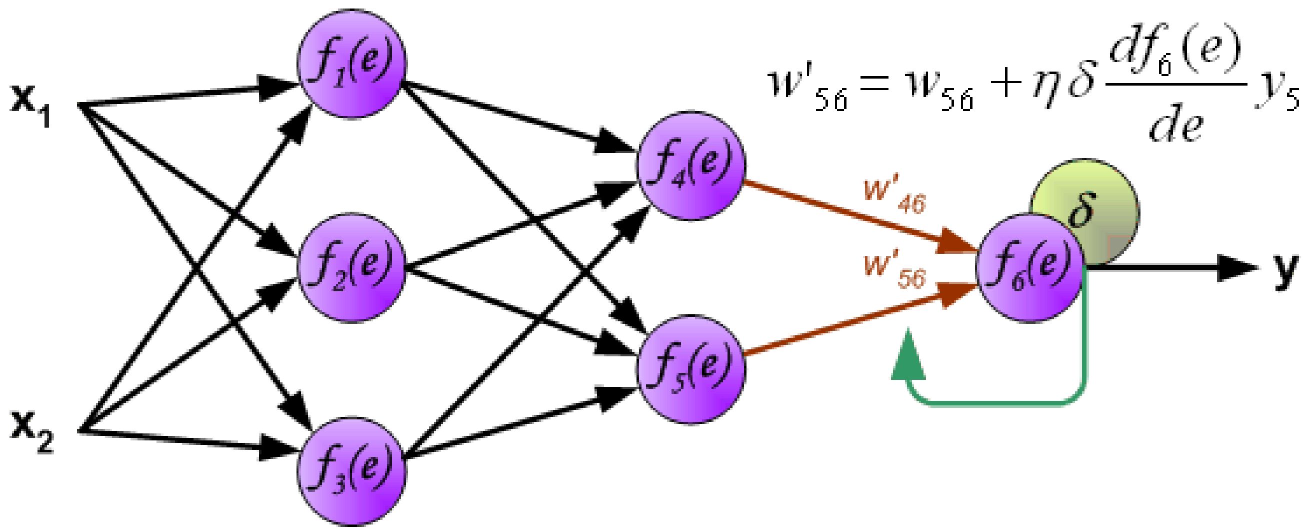
$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$



Atualizando os pesos...

$$w'_{46} = w_{46} + \eta\,\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\,\delta \frac{df_6(e)}{de} y_5$$



Regra de ajuste do peso

- Considere:
  - i, ..., c: índice dos neurônios da camada de saída;
  - h, …, H: índice dos neurônios da camada oculta;
  - j, …, d: índice dos neurônios da camada de entrada.
- Sensibilidade:
  - A *S* (sensibilidade) do neurônio *i* descreve como o erro global E(w) muda o valor de net_i.
- A regra de ajuste do peso utiliza a sensibilidade de cada neurônio.
- Seu cálculo será adicionado no algoritmo...

Feed Forward: Alimentação Adiante

// Entrada para Oculta
Para $h=0,\ldots,H$
$\quad net_h \leftarrow 0$
$\quad$ Para $j=0,\ldots,d$
$\quad\quad net_h \leftarrow net_h + W_{hj} X_j$
$\quad$ Fim Para
$\quad \Phi_h \leftarrow Z(net_h)$
Fim Para

// Oculta para Saída
Para $i=0,\ldots,c$
$\quad net_i \leftarrow 0$
$\quad$ Para $h=0,\ldots,H$
$\quad\quad net_i \leftarrow net_i + W_{ih} \Phi_h$
$\quad$ Fim Para
$\quad y_i \leftarrow Z(net_i)$
Fim Para

Backpropagation

Inicialização aleatória dos pesos W' e W, onde $\|W_i\|=1$ e $\|W_h\|=1$

Para todos os padrões $\underline{X}^{(k)}, k=1,\ldots,n,$ em ordem aleatória:

Feedfoward de $\underline{X}^{(k)}: \underline{\hat{y}}(x^{(k)})$

Para $i=1,\ldots,c:$

$\delta_i^{(k)} = y_i^{(k)} - \hat{y}_i(\underline{x}^{(k)})$ // Erro na saída i do padrão k

$S_i^{(k)} \leftarrow 2.\delta_i^{(k)}.Z'(net_i^{(k)})$ // Sensibilidade

Para $h=0,\ldots,H:$

$w_{ih}^{novo} \leftarrow w_{ih}^{velho} + \eta.S_i^{(k)}.Z(net_h^{(k)})$ $\Phi_h(X^{(k)})$

// Adaptar os pesos de entrada para a camada oculta W'

Para $h=1,\ldots,H:$

$aux \leftarrow 0$

Para $i=0,\ldots,c:$

$aux \leftarrow aux + w_{ih}^{velho}.S_i^{(k)}$

Para $j=0,\ldots,d:$

$w_{hj}^{novo} \leftarrow w_{hj}^{velho} + \eta.S_h^{(k)}.X_j^{(k)}$

Metodologia de Treinamento

- Deve-se realizar a divisão do conjunto de dados dos n padrões $X^{(k)}$, k=1, …, n em três conjuntos de dados:
  - Treinar com 75% dos exemplos;
  - Validar com 15% dos exemplos;
  - Testar com 15% dos exemplos.

  Obs.: Essas porcentagens são as padrões do Matlab, mas podem ser alteradas.
- Antes, os dados devem ser normalizados ou estandardizados:
  - Normalização:
    - $\underline{X} = (\underline{X} - \min(\underline{X})) / (\max(\underline{X}) - \min(\underline{X}))$
  - Estandardização:
    - $\underline{X} = (\underline{X} - \text{média}(\underline{X})) / \sigma(\underline{X})$.