Introdução à Modelagem Matemática – SME0241

Deep Learning: Introdução

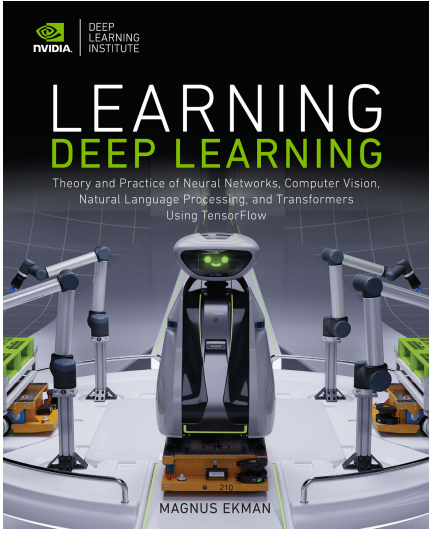
Afonso Paiva Neto

Instituto de Ciências Matemáticas e de Computação
USP - São Carlos

29 de novembro de 2024

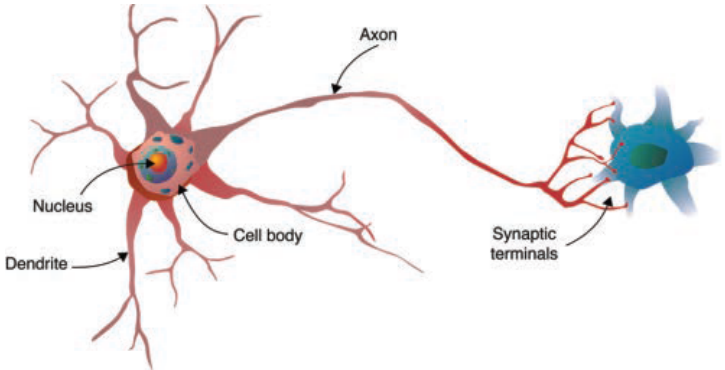
Referência Bibliográfica

- Magnus Ekman - Learning Deep Learning.



Rosenblatt's Perceptron (1957)

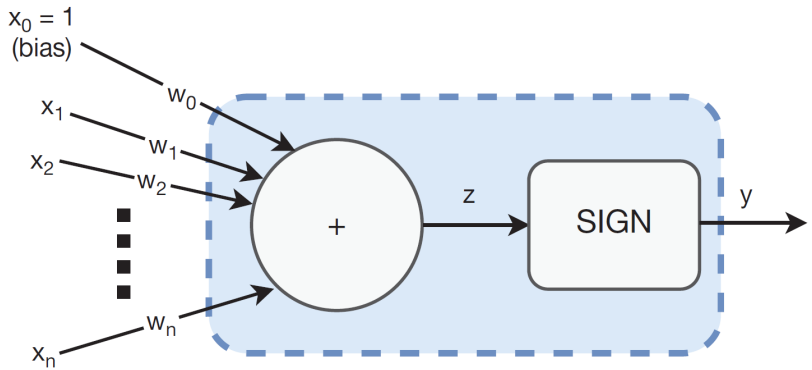
- O Perceptron é um neurônio artificial, ou seja, é um modelo matemático de um neurônio biológico.



Um neurônio biológico (Glassner, A., Deep Learning: From Basics to Practice, The Imaginary Institute, 2018)

Rosenblatt's Perceptron (1957)

- O Perceptron é um neurônio artificial, ou seja, é um modelo matemático de um neurônio biológico.



O Perceptron

Rosenblatt's Perceptron (1957)

- Entrada: $x_0, x_1, \cdots, x_n$, tal que $x_0 = 1$ (bias);
- Pesos: $w_0, w_1, \cdots, w_n$;

Rosenblatt's Perceptron (1957)

- Entrada: $x_0, x_1, \cdots, x_n$, tal que $x_0 = 1$ (bias);
- Pesos: $w_0, w_1, \cdots, w_n$;
- Saída:

$$y = f(z)$$

onde

$$z = \sum_{i=0}^{n} w_i x_i$$

Rosenblatt's Perceptron (1957)

- Entrada: $x_0, x_1, \cdots, x_n$, tal que $x_0 = 1$ (bias);
- Pesos: $w_0, w_1, \cdots, w_n$;
- Saída:

$$y = f(z)$$

onde

$$z = \sum_{i=0}^{n} w_i x_i$$

- Função de ativação (função sinal):

$$f(z) = \begin{cases} -1, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

Perceptron - como o algoritmo de aprende?

- Como determinar os pesos w_i?

Perceptron - como o algoritmo de aprende?

- Como determinar os pesos w_i?
- O Perceptron é um algoritmo de *aprendizado supervisionado* (dados de entrada e saída desejada).

Perceptron - como o algoritmo de aprende?

- Como determinar os pesos w_i?
- O Perceptron é um algoritmo de *aprendizado supervisionado* (dados de entrada e saída desejada).

Algoritmo de aprendizagem

1. Inicializar os pesos de forma aleatória;
2. Selecionar um par entrada/saída de forma aleatória;
3. Calcular a saída y do modelo com os dados de entrada;
4. Se a saída y for diferente do valor esperado, ajustar os pesos da seguinte forma:
   - 4.1 Se $y < 0$, adicionar ηx_i em cada w_i. (η: taxa de aprendizado (hiperparâmetro))
   - 4.2 Se $y > 0$, subtrair ηx_i de cada w_i.
5. Repetir os passos 2, 3 e 4 até o algoritmo predizer os exemplos corretamente.

Aprendizado pelo Gradiente

- Gostaríamos que:

$$y - \hat{y} = 0$$

onde y é o valor desejado e $\hat{y}$ é a saída do modelo.

Aprendizado pelo Gradiente

- Gostaríamos que:

$$y - \hat{y} = 0$$

  onde y é o valor desejado e $\hat{y}$ é a saída do modelo.
- Porém, precisamos combinar múltiplos exemplos de treinamento;

Aprendizado pelo Gradiente

- Gostaríamos que:

$$y - \hat{y} = 0$$

  onde y é o valor desejado e $\hat{y}$ é a saída do modelo.
- Porém, precisamos combinar múltiplos exemplos de treinamento;
- Podemos minimizar o erro quadrático médio (MSE):

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$$

Aprendizado pelo Gradiente

- Gostaríamos que:

$$y - \hat{y} = 0$$

  onde y é o valor desejado e $\hat{y}$ é a saída do modelo.
- Porém, precisamos combinar múltiplos exemplos de treinamento;
- Podemos minimizar o erro quadrático médio (MSE):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

- Método do *Gradiente Descendente*

$$x_{n+1} = x_n - \eta\nabla y$$

Aprendizado pelo Gradiente

- Gostaríamos que:

$$y - \hat{y} = 0$$

  onde y é o valor desejado e $\hat{y}$ é a saída do modelo.
- Porém, precisamos combinar múltiplos exemplos de treinamento;
- Podemos minimizar o erro quadrático médio (MSE):

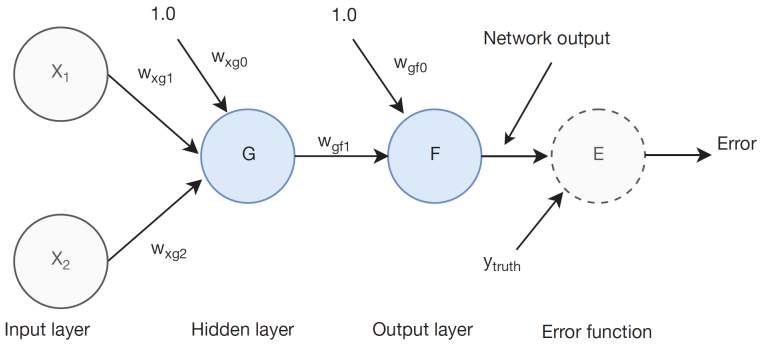
$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

- Método do *Gradiente Descendente*

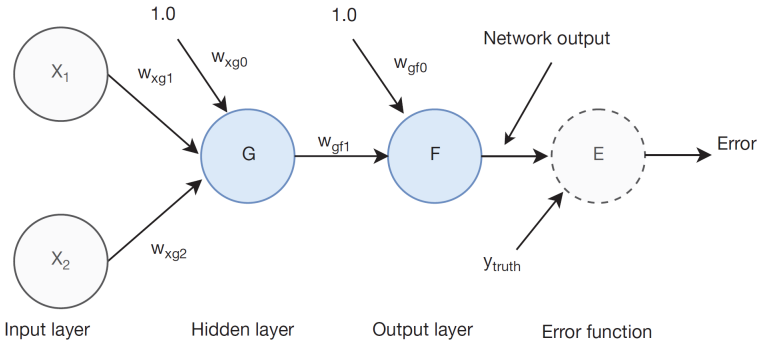
$$x_{n+1} = x_n - \eta\nabla y$$

- **As variáveis são os pesos (w)**. A entrada (x) é constante.

Perceptron Multicamadas (MLP)

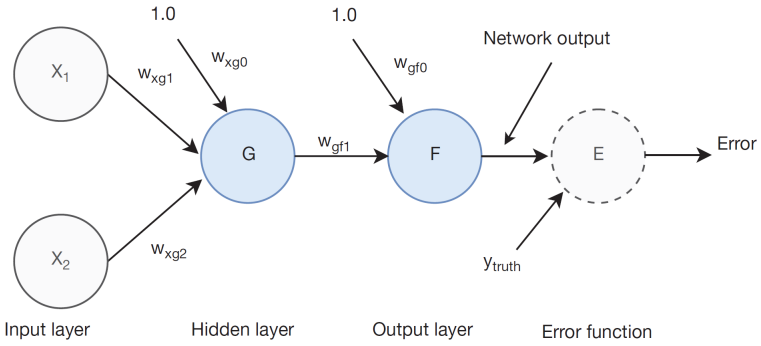


Perceptron Multicamadas (MLP)



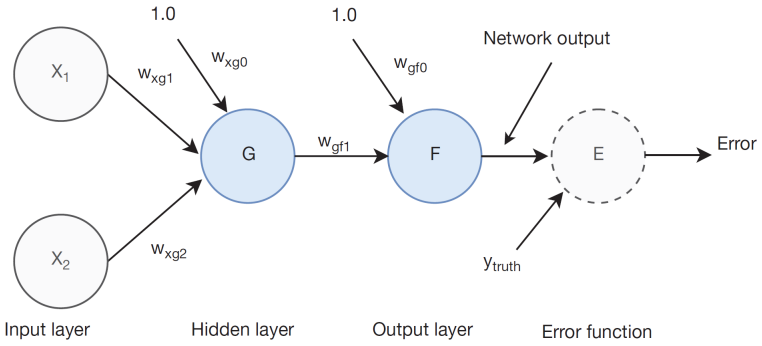
- Extensão de um único perceptron para redes com múltiplas camadas;

Perceptron Multicamadas (MLP)



- Extensão de um único perceptron para redes com múltiplas camadas;
- O modelo também é treinado via gradiente descendente;

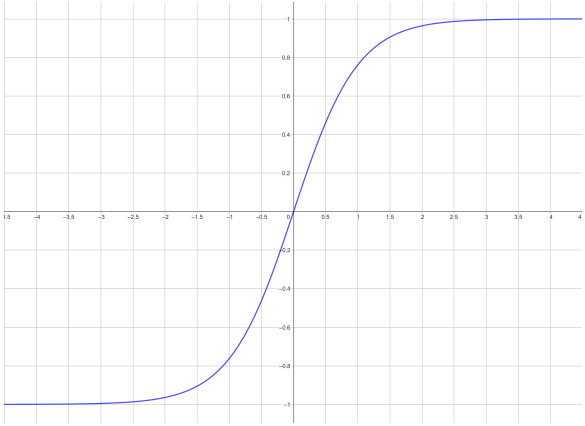
Perceptron Multicamadas (MLP)



- Extensão de um único perceptron para redes com múltiplas camadas;
- O modelo também é treinado via gradiente descendente;
- Como ficam as derivadas parciais?

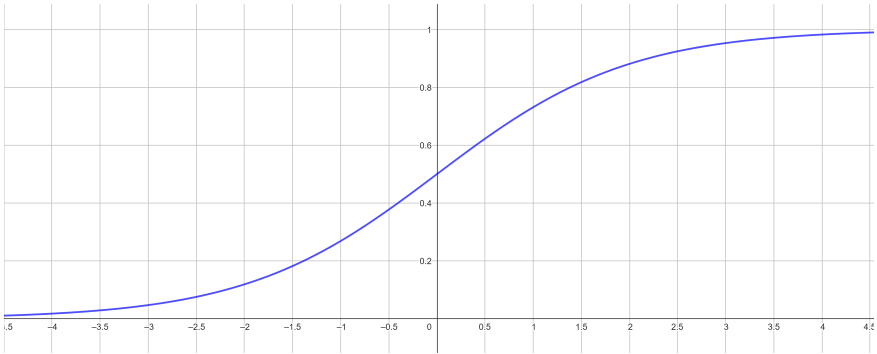
Funções de ativação: Tangente hiperbólica

- $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$ e $tanh'(x) = 1 - tanh^2(x)$

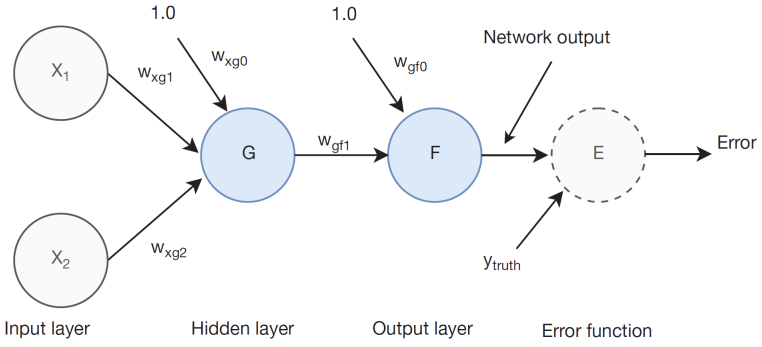


Funções de ativação: Função sigmoide

- $S(x) = \frac{e^x}{e^x+1}$ e $S'(x) = S(x)(1 - S(x))$

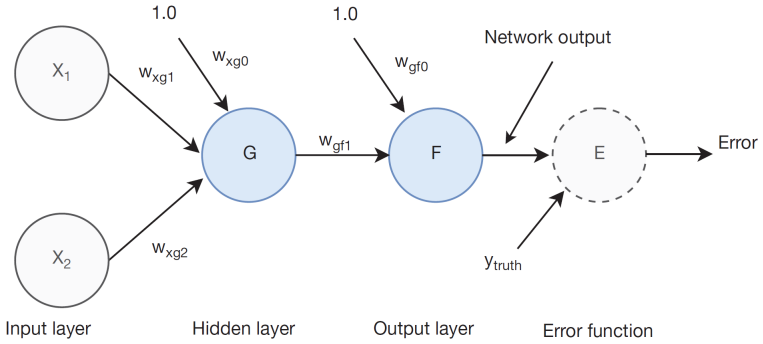


Perceptron Multicamadas (MLP)



Rede simples com duas camadas.

Perceptron Multicamadas (MLP)

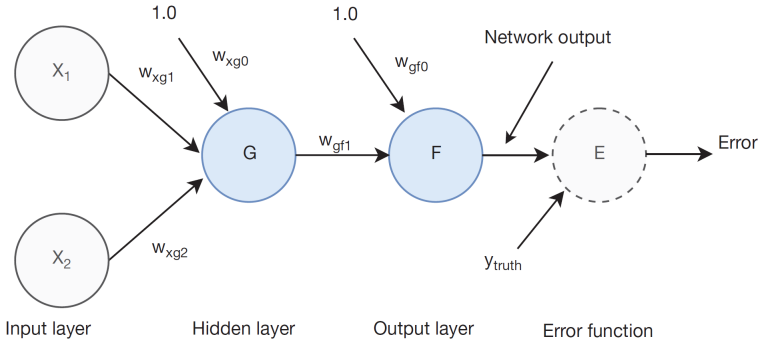


Rede simples com duas camadas.

- A rede neural implementa a seguinte função:

$$\hat{y} = S(w_{gf0} + w_{gf1} tanh(w_{xg0} + w_{xg1} x_1 + w_{xg2} x_2)$$

Perceptron Multicamadas (MLP)



Rede simples com duas camadas.

- Queremos minimizar o erro (função custo):

$$Error = (y - S(w_{gf0} + w_{gf1} tanh(w_{xg0} + w_{xg1} x_1 + w_{xg2} x_2))^2$$

Perceptron Multicamadas (MLP)

- Entrada da função de ativação do neurônio G:

$$z_g(w_{xg0}, w_{xg1}, w_{xg2}) = w_{xg0} + w_{xg1}x_1 + w_{xg2}x_2$$

- Função de ativação do neurônio G:

$$g(z_g) = tanh(z_g)$$

- Entrada da função de ativação do neurônio F:

$$z_f(w_{gf0}, w_{gf1}, g) = w_{gf0} + w_{gf1}g$$

- Função de ativação do neurônio F:

$$f(z_f) = S(z_f)$$

- Função erro:

$$e(f) = \frac{(y - f)^2}{2}$$

Backpropagation: exemplo

- Queremos minimizar:

$$Error(w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}) = e \circ f \circ z_f \circ g \circ z_g$$

Backpropagation: exemplo

- Queremos minimizar:

$$Error(w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}) = e \circ f \circ z_f \circ g \circ z_g$$

- Precisamos calcular as derivadas parciais da função e:

Backpropagation: exemplo

- Queremos minimizar:

$$Error(w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}) = e \circ f \circ z_f \circ g \circ z_g$$

- Precisamos calcular as derivadas parciais da função e:
- $\frac{\partial e}{\partial w_{gf0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf0}}$

Backpropagation: exemplo

- Queremos minimizar:

$$Error(w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}) = e \circ f \circ z_f \circ g \circ z_g$$

- Precisamos calcular as derivadas parciais da função e:
- $\frac{\partial e}{\partial w_{gf0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf0}}$
- $\frac{\partial e}{\partial w_{gf1}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf1}}$

Backpropagation: exemplo

- Queremos minimizar:

$$Error(w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}) = e \circ f \circ z_f \circ g \circ z_g$$

- Precisamos calcular as derivadas parciais da função e:
- $\frac{\partial e}{\partial w_{gf0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf0}}$
- $\frac{\partial e}{\partial w_{gf1}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf1}}$
- $\frac{\partial e}{\partial w_{xg0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial g} \frac{\partial g}{\partial z_g} \frac{\partial z_g}{\partial w_{xg0}}$

Backpropagation: exemplo

- Queremos minimizar:

$$Error(w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}) = e \circ f \circ z_f \circ g \circ z_g$$

- Precisamos calcular as derivadas parciais da função e:
- $\frac{\partial e}{\partial w_{gf0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf0}}$
- $\frac{\partial e}{\partial w_{gf1}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf1}}$
- $\frac{\partial e}{\partial w_{xg0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial g} \frac{\partial g}{\partial z_g} \frac{\partial z_g}{\partial w_{xg0}}$
- $\frac{\partial e}{\partial w_{xg1}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial g} \frac{\partial g}{\partial z_g} \frac{\partial z_g}{\partial w_{xg1}}$

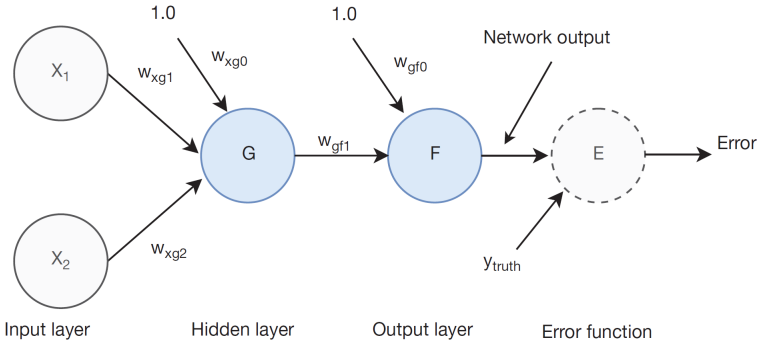
Backpropagation: exemplo

- Queremos minimizar:

$$Error(w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}, w_{gf0}) = e \circ f \circ z_f \circ g \circ z_g$$

- Precisamos calcular as derivadas parciais da função e:
- $\frac{\partial e}{\partial w_{gf0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf0}}$
- $\frac{\partial e}{\partial w_{gf1}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial w_{gf1}}$
- $\frac{\partial e}{\partial w_{xg0}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial g} \frac{\partial g}{\partial z_g} \frac{\partial z_g}{\partial w_{xg0}}$
- $\frac{\partial e}{\partial w_{xg1}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial g} \frac{\partial g}{\partial z_g} \frac{\partial z_g}{\partial w_{xg1}}$
- $\frac{\partial e}{\partial w_{xg2}} = \frac{\partial e}{\partial f} \frac{\partial f}{\partial z_f} \frac{\partial z_f}{\partial g} \frac{\partial g}{\partial z_g} \frac{\partial z_g}{\partial w_{xg2}}$

Backpropagation: exemplo



- $\frac{\partial e}{\partial w_{xg0}} = -(y-f)S'(z_f)w_{gf1}tanh'(z_g)$
- $\frac{\partial e}{\partial w_{xg1}} = -(y-f)S'(z_f)w_{gf1}tanh'(z_g)x_1$
- $\frac{\partial e}{\partial w_{xg2}} = -(y-f)S'(z_f)w_{gf1}tanh'(z_g)x_2$
- $\frac{\partial e}{\partial w_{gf0}} = -(y-f)S'(z_f)$
- $\frac{\partial e}{\partial w_{gf1}} = -(y-f)S'(z_f)g$

Backpropagation

1. **Passo Forward:**
   Calcular as saídas das funções de ativação de cada neurônio e o erro;

2. **Passo Backward:**
   Calcular a derivada $e'(y_f)$ da função erro;
   Calcular o 'erro' de cada neurônio:

$$error_f = -(y - f)S'(z_f)$$
$$error_g = error_f * w_{gf1} tanh'(z_g)$$

3. **Atualização dos pesos:**
   Para cada peso (ex: $w_{gf1} : -(\eta * y_g * error_f)$)