

# Redes Neuronales

Aprendizaje supervisado I

[carlos.andres.delgado@correounivalle.edu.co](mailto:carlos.andres.delgado@correounivalle.edu.co)

Septiembre de 2022

# Contenido

- 1 Preceptrón multicapa (MLP)
- 2 Algoritmo de propagación hacia atrás (BP)
- 3 Métricas de MLP

# Contenido

- 1 Preceptrón multicapa (MLP)
- 2 Algoritmo de propagación hacia atrás (BP)
- 3 Métricas de MLP

# Perceptrón multicapa

## Definición

- Está compuesta por capas de entrada, capas ocultas y capas de salida
- La señal de entrada se propaga hacia adelante entre las distintas capas
- Es una generalización del perceptrón de una capa
- Pueden solucionar problemas más complejos
- El algoritmo más común de entrenamiento es el algoritmo de propagación hacia atrás (*back-propagation*) que se basa en la regla de entrenamiento de corrección del error

# Perceptrón multicapa

## Entrenamiento

- 1 **Paso hacia adelante:** La señal de entrada es aplicada y se propaga capa a capa
- 2 **Paso hacia atrás:** Se ajustan los pesos de cada capa utilizando la regla de corrección de error.

# Perceptrón multicapa



## Características

- 1 **Señal de activación:** Debe ser derivable, ya que en el calculo del error, debemos trabajar con la derivada de la función de activación. Las que se utilizan son función **lineal** y **sigmoide**.
- 2 **Capas ocultas** Pueden ser un o más capas ocultas, las cuales no están conectadas a las entradas y salidas directamente
- 3 **Conectividad** Está determinada por los pesos de las conexiones entre cada capa

# Perceptrón multicapa

## Características

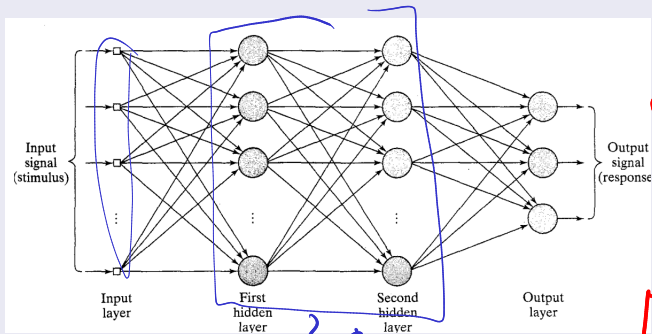


Figura: Arquitectura de MLP [Haykin, 1998]

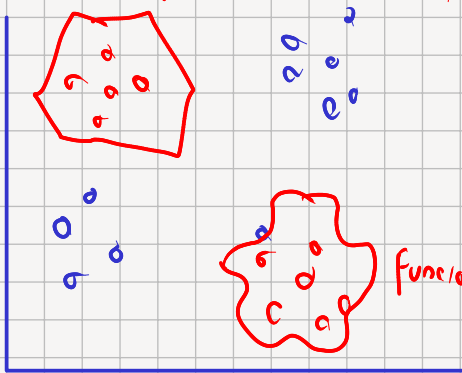
# Perceptrón multicapa

## Características

- 1 La computación de las entradas se puede expresar como una señal continua no lineal
- 2 La computación de un gradiente, es necesario para propagar el error a través de toda la red (regla de aprendizaje) y así ajustar los pesos



Function  $\sigma(x) = \ln(x)$



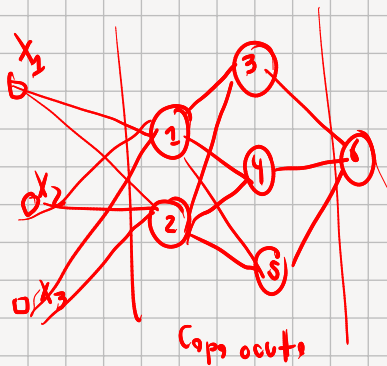
# Contenido

- 1 Preceptrón multicapa (MLP)
- 2 Algoritmo de propagación hacia atrás (BP)
- 3 Métricas de MLP

# Algoritmo BP

## Descripción

- Debe calcularse inicialmente la salida de la red neuronal y. Forward step.
- Para iniciar el proceso de propagación hacia atrás, en el que vamos a tomar el error como entrada de la red desde la capa de salida hacia la de entrada.
- Este proceso requiere hacer derivadas parciales en términos del error (buscando minimizarlo), por lo que la función de activación debe ser derivable.



①①

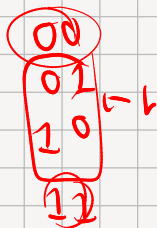
$\vec{v} = 0.000$

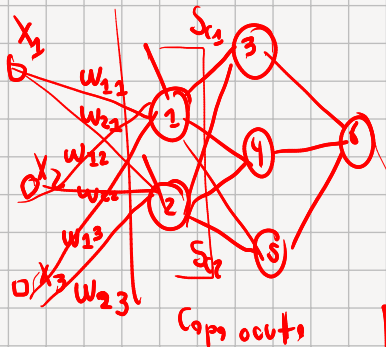
$\vec{A} = 1000$

$\vec{R} = 0100$

$\vec{M} = 0010$

$0001$





$$Net_{0-1} = X_1 w_{11} + X_2 w_{12} + X_3 w_{13}$$

$$Net_{2-2} = X_1 w_{21} + X_2 w_{22} + X_3 w_{23}$$

$$\begin{bmatrix} w_{21} & w_{22} & w_{23} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

$2 \times 3$        $3 \times 1$

$$\begin{bmatrix} w_{11}x_1 + w_{12}x_2 + w_{13}x_3 \\ w_{21}x_1 + w_{22}x_2 + w_{23}x_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \text{Neto}_1 \\ \text{Neto}_2 \end{bmatrix}$$

$$\begin{matrix} \text{bias} \\ [1 \quad 3] \end{matrix}$$

$$Sc_1 = f\left(\begin{bmatrix} \text{Neto}_1 \\ \text{Neto}_2 \end{bmatrix}\right)$$

# Algoritmo BP

## Descripción

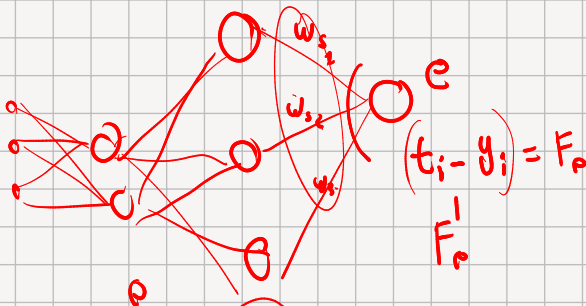
- La entrada neta que recibe una neurona en una capa oculta

$$e_j(n) = t_j(n) - y_j(n)$$

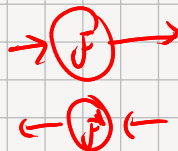
- Se toma como error de una capa  $c$  como el error cuadrático medio

$$\eta(n) = \frac{1}{2} \sum_{j \in c} e_j^2(n)$$

$$|t_i - y_i|$$



$$\frac{(t_i - y_i)^2}{2} = 2(t_i - y_i) \times \frac{\partial y_i}{\partial w_{ji}}$$





# Algoritmo BackPropagation

## Descripción

- La señal de error de una neurona  $j$  en una iteración  $n$  es definida por

$$e_j(n) = t_j(n) - y_j(n)$$

- Se toma como error de una capa  $c$  como el error cuadrático medio

$$\eta(n) = \frac{1}{2} \sum_{j \in c} e_j^2(n)$$

# Algoritmo BackPropagation

## Capa de salida

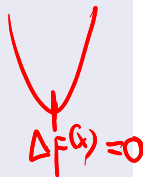
- Se busca el error mínimo, mediante el **gradiente descendiente**

$$\frac{\partial E_j}{\partial w_{ij}}$$

Realizamos los cálculos respectivos y obtenemos:

$$\frac{\partial E_j}{\partial w_{ij}} = -\overset{e}{(t - y)} f'(\overset{e}{Neta}) * \underset{-}{O_j}$$

Donde  $f'$  es la derivada de la función de activación,  $Neta$  es la entrada de la neurona y  $O_j$  es la salida de la neurona de la capa anterior ligada al peso que se está derivando.



# Algoritmo BackPropagation

## Capa de salida

- El proceso de entrenamiento busca modificar el peso  $w_{ij}$  de acuerdo al error calculado de la siguiente forma:

$$w_{ij}(n+1) = w_{ij}(n) + \eta \left( -\frac{\partial E_j}{\partial w_{ij}} \right)$$

De aquí se obtiene

$$w_{ij}(n+1) = w_{ij}(n) + \eta (t - y) f'(Neta) * O_j$$

$w_1, w_2, w_3$

$$\begin{bmatrix} \partial & \partial & \partial \end{bmatrix}$$

$$f(x) = \frac{1}{1 + e^{-x}} \quad F(x) = (1 + e^{-x})^{-1}$$

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

$$f(x)(1 - f(x))$$

tanh

$$\frac{1}{1 + e^{-x}} \left( 1 - \frac{1}{1 + e^{-x}} \right) = \frac{1}{1 + e^{-x}} - \frac{1}{(1 + e^{-x})^2}$$

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

# Algoritmo BackPropagation

## Capa de salida

- Si la función de activación es lineal, se obtiene que la derivada es 1, por lo que la variación del peso será:

$$w_{ij}(n+1) = w_{ij}(n) + \eta(t - y) * O_j$$

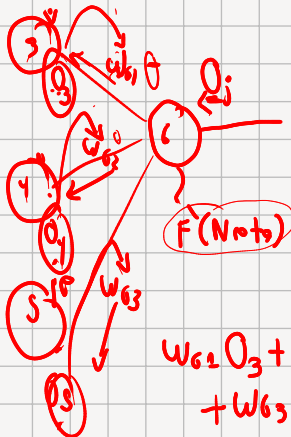
- Si es la función sigmoide  $s = \frac{1}{1+e^{-\eta t a}}$

$$w_{ij}(n+1) = w_{ij}(n) + \eta(t - y) \underline{s(1 - s)} * O_j$$



1

2



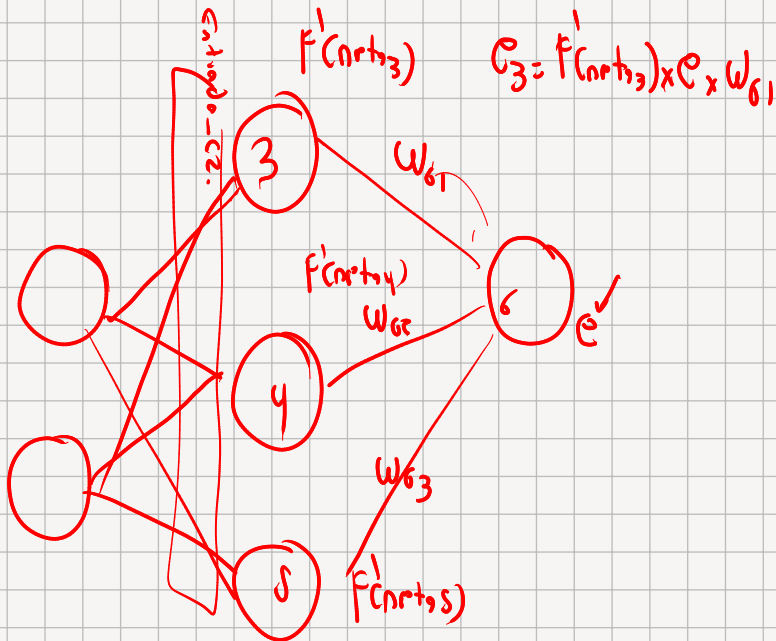
$$w_{61}O_3 + w_{62}O_4 + w_{63}O_5$$

# Algoritmo BackPropagation

## Capa oculta

- La actualización de los pesos depende del error de las capas ocultas siguientes y de salida
- El error de la capa oculta  $h$  y se tiene el conjunto  $C$  neuronas en la siguiente capa.

$$E_h = f'(Net_h)E_{(h+1)}w_{(h+1)i}$$





# Algoritmo BackPropagation

## Descripción

- Se utiliza un conjunto de patrones para entrenar la red
- Se aplica la entrada a la red y se calcula la salida total
- Se calcula el error entre el valor deseado y la salida
- Se propaga el error hacia atrás, es decir que el error de la capa  $n$  se basa en el error de la capa  $n + 1$
- Se modifican los pesos de las capas  $\Delta W$ . Este calculo depende de la capa siguiente.
- Se verifica la condición de parada

# Algoritmo BackPropagation

## Algoritmo

- 1 Se inicializan los pesos del MLP entre  $[-1,1]$
- 2 Mientras la condición de parada sea falsa se repiten los pasos 3 a 12
- 3 Se aplica la entrada
- 4 Se calculan los valores de entrada netos para la capa oculta  $h$

$$Neta^h = \sum_{i=1}^N w_{hj}y_h + \Theta_k$$

Se supone que la capa  $h$  tiene  $N$  neuronas

# Algoritmo BackPropagation

## Algoritmo

- 5 Se calcula la salida de la capa oculta

$$y_h = f_h(Neta_h)$$

- 6 Calculamos los valores netos de entrada para la capa de salida

$$Neta = \sum_{j=1}^L w_{kj} y_h + \Theta_j$$

- 7 Calculamos la salida de la red

# Algoritmo BackPropagation

## Algoritmo

- 8 Calculamos la salida de la red
- 9 Calculamos los términos de error para la capa de salida

$$E^o = (t_u - y_u)f'(Neta)$$

- 10 Estimamos el error para las capas ocultas

$$E^h = f'(Neta) \sum_{k=1}^M E_i^o w_{kj}$$

Como se puede observar, el error de la capa oculta depende de la siguiente capa

# Algoritmo BackPropagation

## Algoritmo

- 10 Actualizamos los pesos en la capa de salida

$$w^o(n+1) = \eta E^o * O_j$$

- 11 Actualizamos los pesos en la capa(s) oculta(s)

$$w^h(n+1) = \eta E^h * O_j$$

- 12 Verificamos si el error global cumple la condición de finalizar (un error mínimo) o un número de iteraciones

$$E_p = \frac{1}{2P} \sum_{p=1}^P \sum_{k=1}^M (t - y)^2$$

# Contenido

- 1 Preceptrón multicapa (MLP)
- 2 Algoritmo de propagación hacia atrás (BP)
- 3 Métricas de MLP

## Métricas durante el entrenamiento

- 1 Función de pérdida (loss function): Es la diferencia entre el valor esperado y el valor obtenido, es una medida local patrón por patrón.
- 2 El error cuadrático medio, es una medida global considerando todos los patrones de entrenamiento

## Métricas después del entrenamiento

Para analizar el rendimiento del MLP contamos con varias métricas, las cuales las analizamos a partir de su pertenencia a la clase 0 o 1.

- Verdaderos positivos ( $V_p$ ): Datos clasificados correctamente como clase 1
- Verdaderos negativos ( $V_n$ ): Datos clasificados correctamente como clase 0
- Falsos positivos ( $F_p$ ): Datos clasificados incorrectamente como clase 1
- Falsos negativos ( $F_n$ ): Datos clasificados incorrectamente como clase 0

Esto se puede expandir a problemas de clasificación m-aria.






## Métricas después del entrenamiento

- 1 Precisión: Porcentaje de los datos de prueba que son correctamente predichos
- 2 Recall: Es la relación entre los verdaderos positivos y los falsos negativos, está dado por:

$$\frac{V_p}{V_p + F_n}$$

- 3 Matriz de confusión: Nos permite observar como se predicen las clases.

# Referencias I

-  Eduardo, C. and Jesus Alfonso, L. (2009).  
*Una aproximación práctica a las redes neuronales artificiales.*  
Colección Libros de Texto. Programa Editorial Universidad del Valle.
-  Haykin, S. (1998).  
*Neural Networks: A Comprehensive Foundation (2nd Edition).*  
Prentice Hall.
-  Widrow, B. and Winter, R. (1988).  
Neural nets for adaptive filtering and adaptive pattern recognition.  
*Computer*, 21(3):25–39.

# ¿Preguntas?

Próximo tema:  
Perceptrón multicapa II